Inhaltsverzeichnis

Kapitel 2

Software-Systemdokumentation

2.1 Einleitung

Dieses Dokument beschreibt die Systemarchitektur des Klimasteuerungssystems. Als Grundlage dient die Anforderungsbeschreibung aus Kapitel 1.3

Der Entwurf wird analog zur Analyse objektorientiert sein und unter Verwendung der OMT-Methode erstellt. Als Notation wird wieder die UML (Unified Modeling Language) verwendet.

2.1.1 Entwurfsentscheidungen

EE	Beschreibung der Entwurfsentscheidung
EE1	Das Zielsystem ist ein zentraler Steuerungsrechner mit einem Portbaustein, über den • alle Sensoren abgefragt • alle Aktuatoren angesteuert werden können.
EE2	Parallelität wird durch Verwendung eines einfachen Schedulers simuliert.
EE3	Die Zeit wird durch eine wiederverwendbare Komponente Timer realisiert.
EE4	Die Anbindung des Portbausteins an den Simulator erfolgt mit Hilfe von Interprozeß-kommunikation.
EE5	Die Benutzungsschnittstelle wird als autonomes Teilsystem realisiert.

Tabelle 1: Entwurfsentscheidungen

2.1.2 Portbaustein

Alle Sensoren und Aktuatoren werden über einen Portbaustein angesteuert. Jeder Typ von Sensoren und Aktuatoren besitzt eine eigene Methode zur Eingabe und Ausgabe von Werten.

Es gibt folgende Arten von Eingabe- und Ausgabeports:

1. *Bit-Eingabeports* für Kontakte.
   Sie werden über int CallBitInputPort(int SectionNr , int RoomNr, char* SensorName) abgefragt.
2. *Integer-Eingabeports* für Temperaturen.
   Sie werden über int CallIntInputPort(int SectionNr, int RoomNr, char* SensorName) abgefragt.
3. *Bit-Ausgabeports* für einfache Aktuatoren.
   Sie werden über void CallBitOutputPort(int SectionNr, int RoomNr, char* ActuatorName, int value) angesteuert.
4. *Integer-Ausgabeports* für komplexere Aktuatoren.
   Sie werden über void CallIntOutputPort(int SectionNr, int RoomNr, char* ActuatorName, int value) angesteuert.

Die im Haus vorhandenen Aktuatoren und Sensoren werden dabei folgendermaßen zugeordnet:

- Bit-Eingabeports für Kontakte: motion detector, pushbutton, status line
- Integer-Eingabeports für Temperaturen und Helligkeitssensoren: room temperature sensor, water temperature sensor, outdoor temperature sensor, outdoor light sensor
- Bit-Ausgabeports für einfache Aktuatoren: pulse
- Integer-Ausgabeports für komplexere Aktoren: blind, radiator valve, dimmable light

Es werden folgende Portzuordnungen festgelegt:

Port	**Typ des Ports**	**SectionNr**	**RoomNr**	**SensorName ActuatorName**
room temperature sensor	IntInput	Section-Nr	Zimmer-Nr	rts1 - rts3
water temperature sensor	IntInput	Section-Nr	Zimmer-Nr	wts1 - wts3
motion detector	BitInput	Section-Nr	Zimmer-Nr	imd1 - imd3
status line	BitInput	Section-Nr	Zimmer-Nr	sl1 - sl2
outdoor light sensor	IntInput	0	0	ols1 - ols6
outdoor temperature sensor	IntInput	0	0	ots
light	BitOutput	Section-Nr	0	pulse1 - pulse2
dimmable light	IntOutput	Section-Nr	Zimmer-Nr	dll1 - dll2
blind	IntOutput	Section-Nr	Zimmer-Nr	blind
radiator valve	IntOutput	Section-Nr	Zimmer-Nr	rad. valve

Tabelle 2: Portzuordnung

2.1.3 Parallelität

Die bisherigen Beschreibungen der Klassen gehen von einer inhärenten Parallelität von Klassen aus. Eine parallele Implementierung der Klassen wäre auf dem Zielrechner (bzw. auf dessen Betriebssystem) zwar möglich, dies würde die Behandlung von Kommunikations- und Synchronisationsproblemen erforderlich machen. Um die dadurch hinzukommende Komplexität zu vermeiden, wird die parallele Abarbeitung simuliert. Dies wird durch einen einfachen Scheduler realisiert, der bestimmte Methoden der Objekte nacheinander periodisch aktiviert.

2.1.3.1 Scheduler

Die Klasse Scheduler wurde im Rahmen des CoDEx-Projektes entworfen. Die nachfolgenden Anforderungen an den Scheduler und der Entwurf des Schedulers sind wörtlich aus dem Systementwurf des zum CoDEx-Projekt gehörenden Systems BAuOS entnommen.

2.1.3.1.1 Anforderungen

Der Entwurf enthält eine Reihe von Automaten, die parallel und unabhängig voneinander arbeiten. Diese Automaten sollen in der Implementierung quasi-parallel arbeiten. Die Realisierung soll so aussehen, daß eine bestimmte Methode eines Objekts in regelmäßigen Abständen ausgeführt wird. In dieser Methode wird das Verhalten des zu dem Objekt gehörigen Automaten nachgebildet. Für die periodische Aktivierung wird ein primitiver Scheduler benötigt, bei dem die auszuführenden Methoden registriert werden müssen.

2.1.3.1.2 Entwurf

Zur Realisierung der Anforderungen sieht der Entwurf zwei Klassen vor. Zum einen eine Basisklasse, von der alle Objekte abgeleitet werden müssen. Diese Basisklasse stellt eine virtuelle Funktion zur Verfügung, die in den abgeleiteten Klassen überladen werden muß und das Verhalten des zugehörigen Automaten nachbildet. Die zweite Klasse realisiert den Scheduler. Die periodische Aktivierung wird mit Hilfe des UNIX Signal-Mechanismus implementiert.

2.1.3.1.2.1 Die Basisklasse ScheduledFct

Jede Klasse, die eine Funktion hat, die periodisch aktiviert werden soll, muß von der Klasse `ScheduledFct` abgeleitet werden (der Name steht für 'Scheduled Function').

```
class ScheduledFct {

public:
  virtual void callBack (void);
  void registerCallBack (void);
  void unregisterCallBack(void);
} // class ScheduledFct
```

Diese Klasse verfügt über zwei Methoden. Die virtuell Methode `callBack` muß in abgeleiteten Klassen überdefiniert werden und realisiert diejenige Funktion, die periodisch aufgerufen werden kann. Wenn mehrere Funktionen periodisch ausgeführt werden sollen, dann müssen diese durch die Funktion `callBack` aufgerufen werden.

Die Methode `registerCallBack` sorgt dafür, daß das Objekt beim Scheduler für die periodische Aktivierung registriert wird. Diese Methode kann vom Objekt selber oder (wenn sie public vererbt wird) von außen aufgerufen werden. Ein Annullieren der Registrierung geschieht durch den Aufruf der Funktion `unregisterCallBack`, durch die die periodische Aktivierung wieder gestoppt werden kann. Die Implementierung der Methoden `registerCallBack` und `unregisterCallBack` wird im Abschnitt 2.1.3.1.2 beschrieben.

2.1.3.1.2.2 Die Klasse Scheduler

Der Scheduler wird durch die Klasse `Scheduler` realisiert:

```
class Scheduler {
  friend class ScheduledFct;

  static Scheduler *myPtr;

  int intervall;
  CallBackList callBacks;

  static void signalCallBack (void);
  void schedule (void);
  void registerCallBack (ScheduledFct *fct);

public:
  Scheduler (int interv);
  ~Scheduler (void);

  void start (void);
  void stop (void);
};
```

Instanzen

Im Kontrollsystem muß eine Instanz dieser Klasse existieren. Es darf nicht mehr als eine Instanz angelegt werden.

Benutzung

Der Konstruktor der Klasse `Scheduler` erwartet einen Parameter, der angibt in welchen Abständen die periodische Aktivierung stattfinden soll. Dieses Intervall wird in Mikrosekunden spezifiziert. Nach dem Erzeugen einer Instanz der Klasse `Scheduler` können sich Objekte, die von der Klasse `ScheduledFct` abgeleitet worden sind, bei dieser Instanz registrieren (Abschnitt 2.1.3.1.2.1).

Durch den Aufruf der Methode `start` beginnt der Scheduler mit der periodischen Aktivierung aller registrierten Objekte. Die Registrierung weiterer Objekte ist danach immer noch möglich. Durch den Aufruf der Methode `stop` kann die Arbeit des Schedulers unterbrochen und ggf. mit `start` wieder aufgenommen werden. Einen Einfluß auf die registrierten Objekte hat das nicht.

Implementierung

Die periodische Aktivierung von Methoden wird dadurch realisiert, daß regelmäßig ein UNIX Signal (`SIGALRM`) erzeugt und abgefangen wird. Im Zuge der Behandlung des Signals werden dann die Methoden ausgeführt.

UNIX ruft beim Eintreffen eines Signals eine Funktion auf, die zuvor als Signal-Handler eingetragen worden sein muß. Für diese Aufgabe kann nicht einfach eine Methode einer Klasse verwendet werden. Methoden erwarten beim Aufruf nämlich immer einen Zeiger auf das zugehörige Objekt (`this`-Pointer). Dieser Pointer wird beim Aufruf als Signal-Handler durch das Betriebssystem nicht übergeben. Das Problem kann dadurch umgangen werden, daß man für diesem Zweck eine statische Methode der Klasse heranzieht. Daher wird eine statische Funktion (`signalCallBack`) in die Klasse aufgenommen, die beim Eintreffen des Signals ausgeführt werden kann. Des weiteren verfügt die Klasse über ein statisches Mitglied (`myPtr`), das einen Pointer auf die Instanz des Scheduler's darstellt (wird durch den Konstruktor der Klasse gesetzt). Über diesen Pointer greift die statische Methode auf die Instanz des Schedulers zu und ruft die Funktion `schedule` auf, die alle registrierten Callbacks nacheinander ausführt.

Die Registrierung von Objekten geschieht ebenfalls über das statische Mitglied, das auf die Instanz des Schedulers zeigt. Die Methode `registerCallBack` der Klasse `ScheduledFct` ruft über diesen Zeiger die Methode `registerCallBack` des Schedulers auf. Damit `ScheduledFct::registerCallBack` auf das dieses Mitglied und die Methode der Klasse `Scheduler` zugreifen darf, muß die Klasse `ScheduledFct` als Friend der Klasse `Scheduler` deklariert werden.

2.1.4 Modellierung der Zeit

Die Modellierung der Zeit wird mit Hilfe einer Timer-Klasse vorgenommen.

2.1.4.1 Timer

Die Klasse Timer wurde im Rahmen des CoDEx-Projektes entworfen. Die nachfolgenden Anforderungen an den Timer und der Entwurf des Timers sind wörtlich aus dem Systementwurf des zum CoDEx-Projekt gehörenden Systems BAuOS entnommen.

2.1.4.1.1 Anforderungen

Die Klasse Timer stellt die aktuelle Zeit für das System zur Verfügung. Es wird nicht direkt auf einen Betriebssystemdienst zurückgegriffen, da zu Testzwecken bestimmte Manipulationen an der Zeit des Kontrollsystems vorgenommen werden sollen. Dazu gehören das Starten zu einem beliebigen Zeitpunkt, der nicht der realen Zeit entspricht und das Raffen der Zeit.

2.1.4.1.2 Entwurf

2.1.4.1.2.1 Die Klasse Timer

Der Zeitdienst wird durch die Klasse `Timer` realisiert.

```
class Timer {
  TimerInternalTimeType t0;
  // user given starting time of the control system
  TimerInternalTimeType t0real;
  // real world starting time of the control system
  int speedup;
  // speedup of control system in comparison with the real world
  TimerInternalTimeType getSystemTime ();
```

```
  // gets real world time from the operating system an converts it
  // to the requested type

public:
  Timer (void)
  Timer (Time t
  Timer (int speed
  Timer (Time t, int speed)

 BAuSysTimeType getActualTime ();
};
```

Instanzen

Im System gibt es eine Instanz dieser Klasse.

Benutzung

Das Ergebnis, das die Funktion `getActualTime` liefert wird nach der Formel

```
t0 + (getSystemTime () - t0real) * speedup
```

berechnet. D.h. die seit dem Start des Kontrollsystems abgelaufene Realzeit (`getSystemTime () - t0real`) wird mit den Beschleunigungsfaktor multipliziert. Dadurch erhält man die Zeitspanne, die für das Kontrollsystem währenddessen abgelaufen sein soll. Schließlich wird dies auf den vorgegebenen Startzeitpunkt des Systems aufaddiert. Damit erhält man die aktuelle Zeit für das Kontrollsystem.

Die verschiedenen Konstruktoren der Klasse sind vorhanden, um ggf. default-Werte automatisch zu setzen.

2.1.5 Benutzungsschnittstelle

Das Setzen von Werten durch den Benutzer oder den Hausmeister erfolgt über Benutzungsschnittstellen (Control Panel). Diese wird durch ein autonomes Subsystem realisiert. Die Kommunikation zwischen Benutzungsschnittstelle und System erfolgt über die Strukturklasse, in der das ControlPanel integriert ist, z.B. beim ControlPanel des Raumes über die Methoden der Klasse *Room*.

2.2 UML-Entwurf

Nachfolgend werden für die Klassen des Entwurfs die Änderungen gegenüber der Analyse kurz beschrieben. Im Anschluß werden die Attribute und Methoden der Klassen mit Hilfe des Klassendiagramms und den Classtables erläutert. Abschließend werden die Sequenzdiagramme, die Kollaborationsdiagrammen, sowie die Statediagramme aufgeführt.

2.2.1 Kurzbeschreibung der Unterschiede zur Analyse

In den folgenden Abschnitten werden für jede Klasse die Änderungen beschrieben, die sich gegenüber der Analyse ergeben haben.

2.2.1.1 Klasse ScheduledFct

Neu hinzugekommen ist die Basisklasse *ScheduledFct*. Die repräsentiert die in Abschnitt 2.1.3.1.2.1 erläuterte Funktionalität.

2.2.1.2 Klasse Scheduler

Die Klasse *Scheduler* ist verantwortlich für die periodische Aktivierung verschiedener Methoden. Seine Funktionalität wurde in Abschnitt 2.1.3.1.2.2 beschrieben.

2.2.1.3 Klasse Timer

Die Klasse *Timer* repräsentiert die in Abschnitt 2.1.4 beschriebene Klasse.

2.2.1.4 Klasse BlindController

Zur Abfrage des Lockstatus wurden die Methoden blindEinstellungTempEnabled und blindEinstellungLichtEnabled hinzugefügt.

2.2.1.5 Klasse PRBlindController

Die Methode PRTempController.SetValues wurde durch setzeZeitspanneVerlassen ersetzt.

2.2.1.6 Klasse PRTempController

Die Methode PRTempController.SetValues wurde durch ersetzt durch die drei Methoden setzeZeitspanneVerlassen, setzeZeitspanneEintritt und setzeDT2.

2.2.1.7 Klasse RadiatorController

Es wurden die Methoden VentilOeffnen und VentilSchliessen zur Regelung des RadiatorValve hinzugefügt.

2.2.1.8 Klasse RoomTemperatureController

Die Funktionalität des RoomTemperatureController wurde auf das Berechnen und Durchreichen der Temperaturen an den Temperatureregulator beschränkt. Dafür wurden die privaten Variablen MinTemp, MaxTemp, ZielTemp in die Klasse Temperatureregulator verlegt.

2.2.1.9 Klasse Temperatureregulator

Die Klasse Temperatureregualtor wurde durch Vererbung in TemperatureregulatorWithBlinds und TemperatureregulatorWithoutBlinds unterteilt.

Die Methoden ZimmerTempGut, ZimmerTempZuHoch und ZimmerTempZuTief wurden ersatzlos gestrichen. Die Methode SetValues wurde durch setzeTemperaturenUndModus ersetzt.

2.2.1.10 Klasse TemperatureregulatorWithBlinds

Zusätzlich zu der Regelung des Temperatureregulator verwendet TemperatureregulatorWithBlinds die Blindssteuerung zur Regelung der Temperatur. Die Methode istBlindEinstellungRelevant wurde aus Temperatureregulator zu dieser Klasse verschoben.

2.2.1.11 Klasse TemperatureregulatorWithoutBlinds

TemperatureregulatorWithoutBlinds ist eine Unterklasse von Temperatureregulator und wird in Rooms ohne Blindsteuerung eingesetzt. Es implementiert keine zusätzliche Funktionalität.

2.2.1.12 Klasse ValueInterface

Die Klasse ValueInterface wurde durch Vererbung in ValueInterfaceWithBlinds und ValueInterfaceWithoutBlinds unterteilt.

2.2.1.13 Klasse ValueInterfaceWithBlinds

Repräsentiert die Schnittstelle zum Teilsystem Systemmodusermittlung. Wird in Rooms verwendet, in denen Blinds vorhanden sind. ValueInterfaceWithBlinds erweitert die Methode SetValues, so dass die entsprechenden Werte auch an PRBlindController weitergeleitet werden.

2.2.1.14 Klasse ValueInterfaceWithoutBlinds

Repräsentiert die Schnittstelle zum Teilsystem Systemmodusermittlung. ValueInterfaceWithoutBlinds wird in Rooms ohne Blinds verwendet und implementiert keine weitere Funktionalität.

2.2.1.15 Klasse Heizperioden

In dieser Klasse kommt eine Methode IsHeatingPeriod() hinzu, die für einen gegebenen Tag prüft, ob dieser in einer Heizperiode liegt oder nicht.

2.2.1.16 Klasse Urlaubszeiten

In dieser Klasse kommt eine Methode IsHoliday() hinzu, die für einen gegebenen Tag prüft, ob dieser ein Urlaubstag ist oder nicht.

2.2.1.17 Klasse Wochenintervale

In dieser Klasse kommt eine Methode IsWorkingDay() hinzu, die für einen gegebenen Tag prüft, ob dieser ein Arbeitstag ist oder nicht.

2.2.1.18 Klasse Tagesintervalle

In dieser Klasse kommt eine Methode GetMinMode() hinzu, die für eine gegebene Minute prüft, welcher Modus (comfort, standby, off) gerade gültig ist.

2.2.1.19 Klasse Benutzerdaten

Diese Klasse wurde um drei Methoden erweitert
Check() ist eine abstrakte Methode die an alle Unterklassen vererbt wird. Sie prüft je nach Instanz, ob der entsprechende Benutzerwert gültig ist.
MinIntervall() und MaxIntervall() ermitteln das Minimum bzw. das Maximum eines gegebenen Intervalls.

2.2.1.20 Klasse TempOpModusErmittlung

Diese Klasse ist jetzt eine Unterklasse von ScheduledFct

2.2.1.21 Klasse Temperaturdaten

In dieser Klasse kommen zwei Methoden hinzu, Check() und CheckIntervall().

2.2.1.22 Klasse Spezialzeiten und Zeitspannen

Diese beiden Klassen wurden verfeinert in die Klassen ClassZeitspanneEintrittNormal, Class ZeitspanneEintrittStandbyNormal, ClassZeitspanneAufheizenNormal, ClassZeitspanneAufheizenStandbyNormal, ClassDT2, ClassDT2Standby und ZeitspannenVerlassen.
Jede Klasse hat die Methoden Check() und CheckIntervall(), die je nach Klasse verschieden realisiert sind.

2.2.1.23 Generelles

Es wurde festgestellt, daß in Räume mit und ohne Blinds unterschieden werden muß. Daher mußte die Klasse Room in RoomWithoutBlinds und RoomWithBlinds unterteilt werden. Entsprechend mußten der RoomLightRegulator und der RoomLightController ebenso unterteilt werden um die Lichtregelung für die unterschiedlichen Raumartern durchführen zu können.

Weiterhin wurden alle Funktionen und Attribute kleingeschrieben.

2.2.1.24 Klasse Room

Die Klasse Room wurde durch Vererbung in RoomWithBlinds und RoomWithoutBlinds unterteilt. Jeder Raum, egal ob mit oder ohne Blinds, besitzt zwei DimmableLightController (einen pro DimmableLight)

2.2.1.24.1 Klasse RoomWithBlinds

Die Klasse RoomWithBlinds repräsentiert einen Raum der mit Blinds ausgestattet ist. Jeder RoomWithBlinds besitzt einen BlindController, einen RoomLightControllerWithBlinds und einen RoomLightRegulatorWithBlinds. Die Vaterklasse zu RoomWithBlinds ist die Klasse Room.

2.2.1.24.2 Klasse RoomWithoutBlinds

Die Klasse RoomWithoutBlinds repräsentiert einen Raum der nicht mit Blinds ausgestattet ist. Jeder RoomWithoutBlinds besitzt einen RoomLightControllerWithoutBlinds und einen RoomLightRegulatorWithoutBlinds. Die Vaterklasse zu RoomWithoutBlinds ist die Klasse Room.

2.2.1.25 Klasse RoomLightRegulator

Die Klasse RoomLightRegulator wurde durch Vererbung in RoomLightRegulatorWithBlinds und RoomLightRegulatorWithoutBlinds unterteilt. Dei beiden abgeleiteten Klassen unterscheiden sich nur in der Implementierung der Funktion updateLight() die jeweils die virtuelle Funktion die der Basisklasse RoomLightRegulator überladen. Die Funktion updateLight() wird ab sofort durch einen Sceduler in gewissen Abständen aufgerufen, daher wird diese Klasse aktiv und RoomLightController wird passiv (siehe auch Anmerkungen zur Klasse RoomLightController).

Rückgabewerte und Parameter angegeben und deren Typen festgelegt.

2.2.1.25.1 Klasse RoomLightRegulatorWithBlinds

Wird von der Klasse RoomLightRegulator abgeleitet und repräsentiert einen RoomLightRegulator für einen Room mit Blinds. Die Funktion updateLight() wird unter Beachtung des Einflusses der Blinds auf die Helligkeit im Raum implementiert.

2.2.1.25.2 Klasse RoomLightRegulatorWithoutBlinds

Wird von der Klasse RoomLightRegulator abgeleitet und repräsentiert einen RoomLightRegulator für einen Room ohne Blinds. Die Funktion updateLight() wird ohne Beachtung des Einflusses der Blinds auf die Helligkeit im Raum implementiert.

2.2.1.26 Klasse RoomLightController

Die Klasse RoomLightController wird durch Ableitung in RoomLightControllerWithBlinds und RoomLightControllerWithoutBlinds unterteilt. RoomLightController wird passiv. Das heisst die Funktion updateLight() wird nicht mehr von RoomLightController aufgerufen sondern die LightIntensity wird von RoomLightRegulator aktiv abgefragt. Die Datenkomponente indoorSunLightLevel wurde nicht mehr benötigt und daher gelöscht.

2.2.1.26.1 Klasse RoomLightControllerWithBlinds

Die Klasse RoomLightControllerWithBlinds repräsentiert einen RoomLightController für einen Raum mit Blinds. Die Klasse besitzt eine Methode zur Berechnungn der optimalen Blind Position (calculateOptimalBlindPos()). Weiterhin wurden die Funktionen getIntensityLevelBlindsUp() und getIntensityLevelBlindsDown() von der Vaterklasse RoomLightController in die Klasse RoomLightControllerWithBlinds verlagert.

2.2.1.26.2 Klasse RoomLightControllerWithoutBlinds

Die Klasse RoomLightControllerWithoutBlinds repräsentiert einen RoomLightController für einen Raum ohne Blinds. Der Konsistenz halber existiert diese Klasse, obwohl sie nichts anderes tut, als den Wert der OutdoorLightSensors zurückzuliefern.

2.2.1.27 Klasse DimmableLightController

Die Klasse DimmableLightController repräsentiert eine Lampe im Raum. Sie wurde komplett neu hinzugefügt. Sie kapselt das Verhalten der DimmableLights und enthält daher eine Methode zum ein/ausschalten des Lichts (‘switchLight(..)’), sowie eine Methode zum dimmen (‘dim(..)’). Des weiteren ist sie eine aktive Klasse, d.h. vom Scheduler abgeleitet und prüft in periodischen Abständen, ob der Benutzer von Hand (PushButton) die Lampen im Raum ein-/bzw. ausgeschaltet hat.

2.2.1.28 Klasse ControlPanelFM

In der Klasse ControlPanelFM wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt. Die Methode SetZeitspanneVerlassen() wird zum Durchreichen eines Wertes ZeitspanneVerlassen vom Typ durationType vom ControlPanelFM-Objekt zu einem PRHallwayLightController-Objekt benötigt. Die Methode EnterFMLichtAnsteuern() wird zum Durchreichen eines Wertes LichtStatus vom Typ zLichtStatusType vom ControlPanelFM-Objekt zu einem LightControlUnit-Objekt benötigt.

2.2.1.29 Klasse House

In der Klasse House wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt. Die Methode SetZeitspanneVerlassen() wird zum Durchreichen eines Wertes ZeitspanneVerlassen vom Typ durationType vom ControlPanelFM-Objekt zu einem PRHallwayLightController-Objekt benötigt. Die Methode EnterFMLichtAnsteuern() wird zum Durchreichen eines Wertes LichtStatus vom Typ zLichtStatusType vom ControlPanelFM-Objekt zu einem LightControlUnit-Objekt benötigt.

2.2.1.30 Klasse Section

In der Klasse Section wurden das Attribut SectionNr und die Methoden SetZeitspanneVerlassen(), EnterFMLichtAnsteuern(), GetSectionNr() eingefügt. Die Methode SetZeitspanneVerlassen() wird zum Durchreichen eines Wertes ZeitspanneVerlassen vom Typ durationType vom ControlPanelFM-Objekt zu einem PRHallwayLightController-Objekt benötigt. Die Methode EnterFMLichtAnsteuern() wird zum Durchreichen eines Wertes LichtStatus vom Typ zLichtStatusType vom ControlPanelFM-Objekt zu einem LightControlUnit-Objekt benötigt.

2.2.1.31 Klasse HallwaySection

In der Klasse HallwaySection wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt. Die Methode SetZeitspanneVerlassen() wird zum Durchreichen eines Wertes ZeitspanneVerlassen vom Typ durationType vom ControlPanelFM-Objekt zu einem PRHallwayLightController-Objekt benötigt. Die Methode EnterFMLichtAnsteuern() wird zum Durchreichen eines Wertes LichtStatus vom Typ zLichtStatusType vom ControlPanelFM-Objekt zu einem LightControlUnit-Objekt benötigt. Die Attribute SectionNr und RoomNr wurden eingefügt, um Anwesenheitsdetektoren und Lichter auf der Hardware-Ebene eindeutig initialisieren bzw. ansprechen zu können. Weiterhin wurde die an einigen Stellen des Dokumentes falsch eingegebene Bezeichnung “HallwaySector” in “HallwaySection” korrigiert.

2.2.1.32 Klasse PresenceController

Die Klasse PresenceController wird aufgrund ihres aktiven Verhaltens von der Klasse ScheduledFct abgeleitet. Wegen dem unterschiedlichen Verhalten der Klasse PresenceController in den Räumen und in den Flurabschnitten, wurde sie als abstrakte Klasse festgelegt, von der die Klassen PresenceControllerRoom und PresenceControllerHallway abgeleitet werden. Desweiteren wurde die private Methode ErmittleAnwesenheit() hinzufügt, die die Abfrage der Zustandskontakte kapselt. Von der Klasse PresenceController abgeleitete Klassen implementieren diese Methode entsprechend ihrer Anforderungen.

2.2.1.33 Klasse PresenceControllerHallway

Die Klasse PresenceControllerHallway wurde neu eingefügt. Sie wird von der Klasse PresenceController abgeleitet. Das ist notwendig, um die vier Anwesenheitsdetektoren (repräsentiert durch Instanzen der Klasse DoorContact) an den Türen zuzüglich dem Anwesenheitsdetektor für den gesamten Flurabschnitt (repräsentiert durch eine Instanz der Klasse Contact) eines Flurabschnittes anzusprechen. Die Abfrage der insgesamt fünf Anwesenheitskontakte erfolgt absofort über die von der Klasse PresenceController abgeleiteten Methode ErmittleAnwesenheit(). Diese Methode wird überschrieben. Sie kapselt die Aufrufe der Methode AbfrageZustand() der jeweiligen Anwesenheitsdetektoren. Weiterhin wurden die Attribute SectionNr und RoomNr eingefügt, um Anwesenheitsdetektoren und Lichter auf der Hardware-Ebene eindeutig initialisieren bzw. ansteuern zu können.

2.2.1.34 Klasse PRHallwayLightController

Die Klasse PRHallwayLightController wird aufgrund ihres aktiven Verhaltens von der Klasse ScheduledFct abgeleitet. Der Name PRHallwayLightControler wurde in PRHallwayLightController korrigiert.

2.2.1.35 Klasse LightControlUnit

Die Klasse LightControlUnit wird aufgrund ihres aktiven Verhaltens von der Klasse ScheduledFct abgeleitet. Folgende Änderungen wurden vorgenommen, um die Kohäsion innerhalb der Klasse zu maximieren: Die Methoden JemandDa(), NiemandDa() wurden zur Methode AnwesenheitMelden(zAnwesenheitType) zusammengefaßt. Die Methoden FM-SchaltetEin(), FM-SchaltetAus() wurden zur Methode FMLichtAnsteuern(zLichtStatusType) zusammengefaßt.

2.2.1.36 Klasse LightController

Die Klasse LightController wird aufgrund ihres aktiven Verhaltens von der Klasse ScheduledFct abgeleitet. Der Name LightControler wurde in LightController korrigiert. Die Attribute SectionNr und RoomNr wurden eingefügt, um Anwesenheitsdetektoren und Lichter auf der Hardware-Ebene eindeutig initialisieren bzw. ansprechen zu können. Die Methoden SchalteLichtEin(), SchalteLichtAus() wurden in der Methode LichtAnsteuern(zLichtStatusType) zusammengefaßt, um die Kohäsion innerhalb der Klasse zu maximieren.

2.2.1.37 Klasse Sensor

2.2.1.37.1 Neue Attribute der Klasse Sensor

Es wurden die Attribute Name, RoomNr, SectionNr, PortNr zugefügt, um eine eindeutige Identifikation des Sensors zu ermöglichen und die notwendigen Daten zum Aufruf der Methoden des Portbausteins innerhalb der Klasse zur Verfügung zu haben.

2.2.1.38 Klasse RoomTemperatureSensor

2.2.1.38.1 Änderungen an den Attributen der Klasse RoomTemperatureSensor

Das Attribut Temperature wurde entfernt, da der Temperaturwert nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.1.38.2 Neue Attribute der Klasse RoomTemperatureSensor

Es wurden die Attribute LowerTempLimit und UpperTempLimit zugefügt. Diese Attribute legen den gültigen Wertebereich des Sensors fest.

2.2.1.39 Klasse WaterTemperatureSensor

2.2.1.39.1 Änderungen an den Attributen der Klasse WaterTemperatureSensor

Das Attribut Temperature wurde entfernt, da der Temperaturwert nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.1.39.2 Neue Attribute der Klasse WaterTemperatureSensor

Es wurden die Attribute LowerTempLimit und UpperTempLimit zugefügt. Diese Attribute legen den gültigen Wertebereich des Sensors fest.

2.2.1.40 Klasse OutdoorTemperatureSensor

2.2.1.40.1 Änderungen an den Attributen der Klasse OutdoorTemperatureSensor

Das Attribut Temperature wurde entfernt, da der Temperaturwert nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.1.40.2 Neue Attribute der Klasse OutdoorTemperatureSensor

Es wurden die Attribute LowerTempLimit und UpperTempLimit zugefügt. Diese Attribute legen den gültigen Wertebereich des Sensors fest.

2.2.1.41 Klasse OutdoorLightSensor

2.2.1.41.1 Änderungen an den Attributen der Klasse OutdoorLightSensor

Das Attribut Intensity wurde entfernt, da die Intensität nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.1.41.2 Neue Attribute der Klasse OutdoorLightSensor

Es wurden die Attribute LowerIntensityLimit und UpperIntensityLimit zugefügt. Diese Attribute legen den gültigen Wertebereich des Sensors fest.

2.2.1.42 Klasse Contact

2.2.1.42.1 Änderungen an den Attributen der Klasse Contact

Das Attribut Zustand wurde entfernt, da der Zustandswert nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.1.43 Klasse Actuator

2.2.1.43.1 Neue Attribute der Klasse Actuator

Es wurden die Attribute Name, RoomNr, SectionNr, PortNr zugefügt, um eine eindeutige Identifikation des Aktuators zu ermöglichen und die notwendigen Daten zum Aufruf der Methoden des Portbausteins innerhalb der Klasse zur Verfügung zu haben.

2.2.1.44 Klasse StatusLine

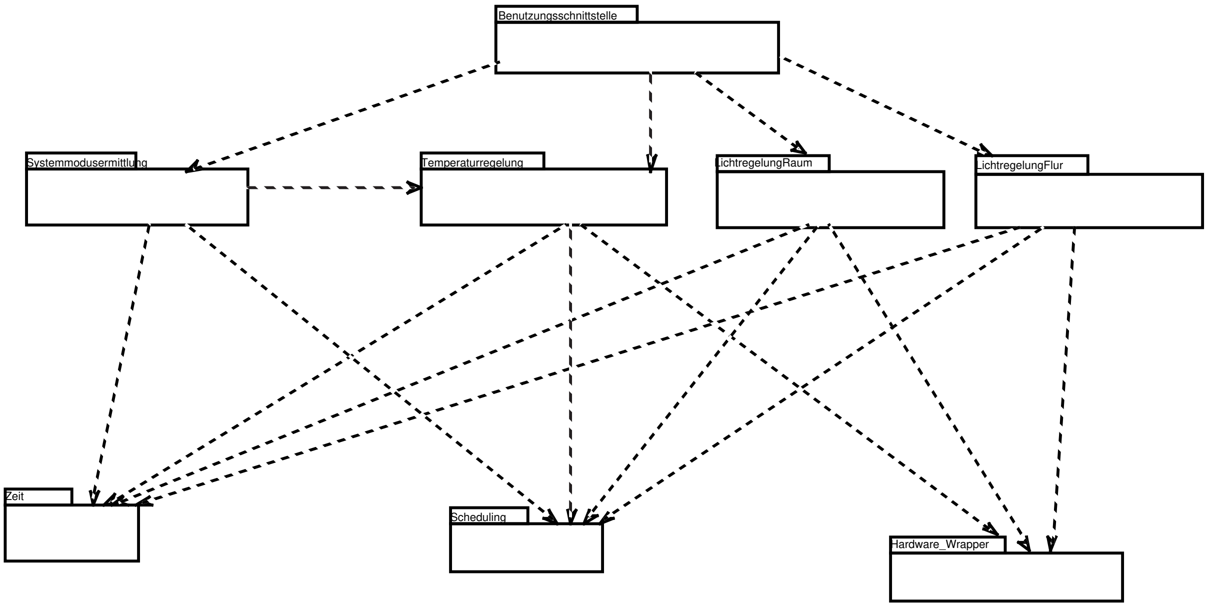
2.2.1.44.1 Änderungen an den Attributen der Klasse StatusLine

Das Attribut State wurde entfernt, da der Status des betroffenen Lichts nicht innerhalb der Klasse zwischengespeichert wird, sondern direkt der aktuelle Wert aus dem Portbaustein "ausgelesen" wird.

2.2.2 UML-Klassendiagramme

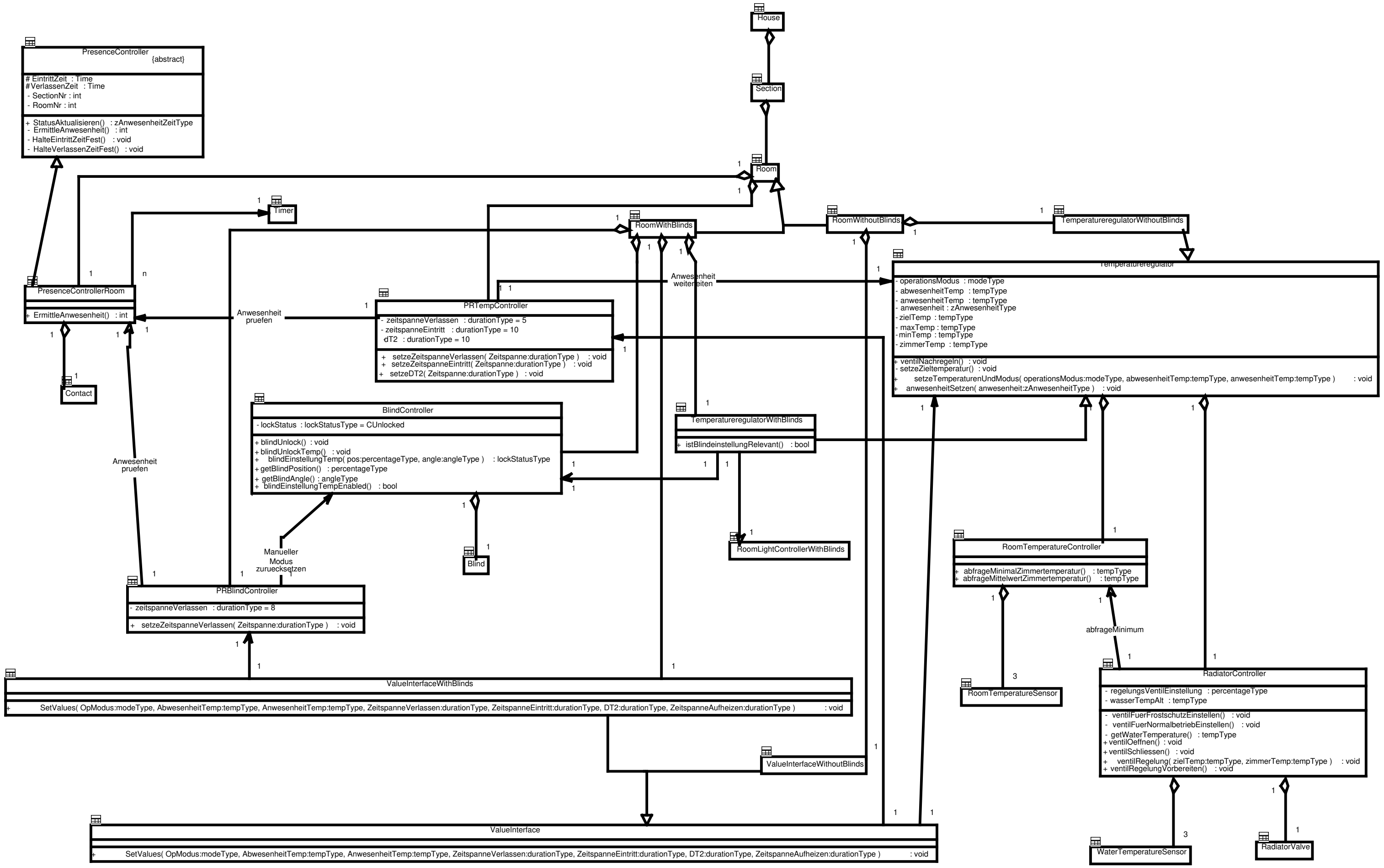
2.2.2.1 Packagestruktur

Dieses Diagramm zeigt die Aufteilung des Systems in einzelne Subsysteme (Packages). Die einzelnen Packages zeigen Sichen auf die einzelnen Klassen. Deshalb sind in einem Package nicht unbedingt alle Methoden und Attribute einer Klasse definiert.



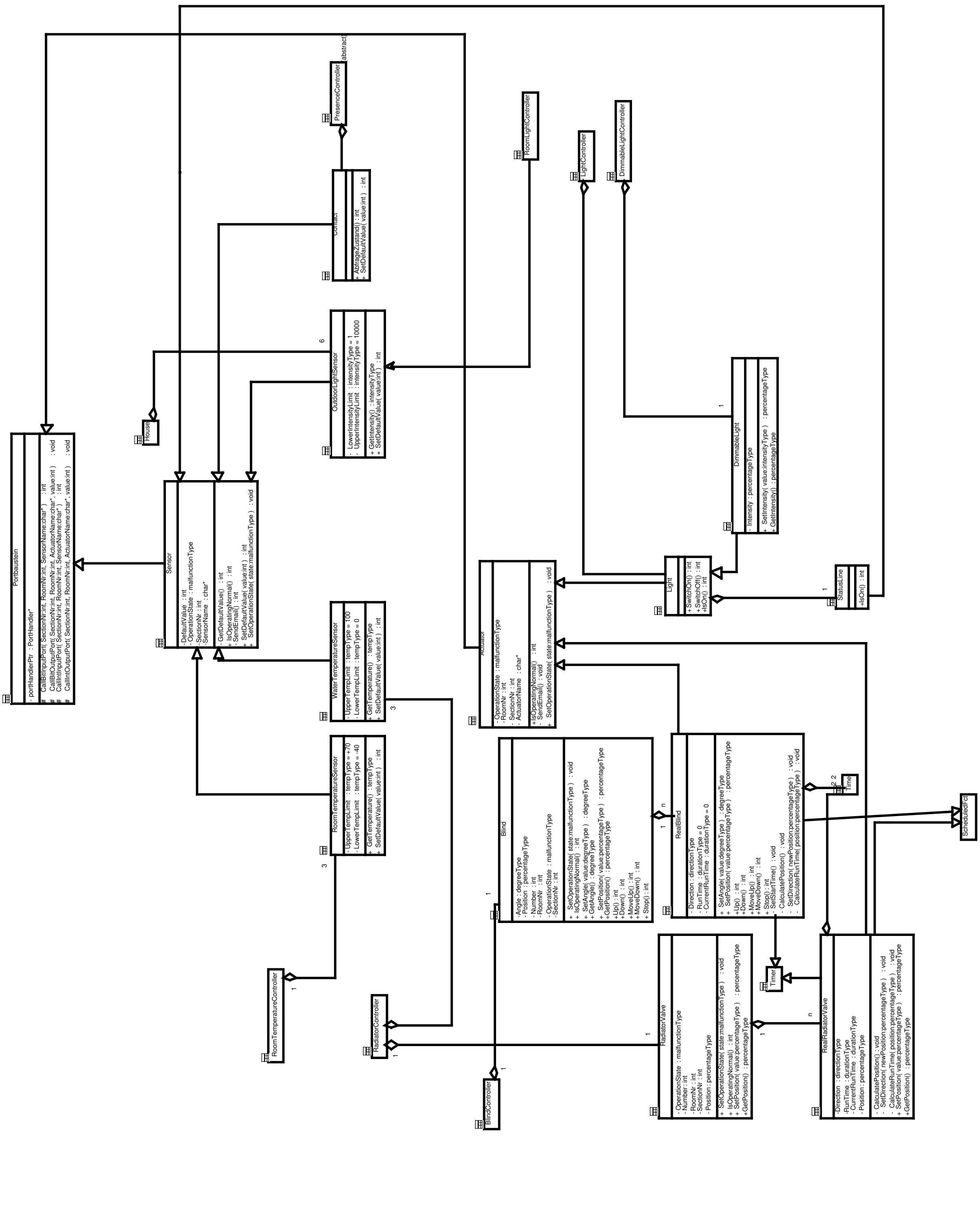
2.2.2.2 Temperaturregelung

Dieses Diagramm zeigt alle Klassen, die an der Temperaturregelung beteiligt sind.


House
Section
Room
PresenceController
{abstract}
EintrittZeit : Time
VerlassenZeit : Time
- SectionNr : int
- RoomNr : int
+ StatusAktualisieren() : zAnwesenheitZeitType
- ErmittleAnwesenheit() : int
- HalteEintrittZeitFest() : void
- HalteVerlassenZeitFest() : void
Timer
RoomWithBlinds
RoomWithoutBlinds
TemperatureregulatorWithoutBlinds
Temperatureregulator
- operationsModus : modeType
- abwesenheitTemp : tempType
- anwesenheitTemp : tempType
- anwesenheit : zAnwesenheitType
- zielTemp : tempType
- maxTemp : tempType
- minTemp : tempType
- zimmerTemp : tempType
+ ventilNachregeln() : void
- setzeZieltemperatur() : void
+ setzeTemperaturenUndModus(operationsModus:modeType, abwesenheitTemp:tempType, anwesenheitTemp:tempType) : void
+ anwesenheitSetzen(anwesenheit:zAnwesenheitType) : void
PresenceControllerRoom
+ ErmittleAnwesenheit() : int
Anwesenheit pruefen
Anwesenheit weiterleiten
PRTempController
- zeitspanneVerlassen : durationType = 5
- zeitspanneEintritt : durationType = 10
dT2 : durationType = 10
+ setzeZeitspanneVerlassen(Zeitspanne:durationType) : void
+ setzeZeitspanneEintritt(Zeitspanne:durationType) : void
+ setzeDT2(Zeitspanne:durationType) : void
Contact
BlindController
- lockStatus : lockStatusType = CUnlocked
+ blindUnlock() : void
+ blindUnlockTemp() : void
+ blindEinstellungTemp(pos:percentageType, angle:angleType) : lockStatusType
+ getBlindPosition() : percentageType
+ getBlindAngle() : angleType
+ blindEinstellungTempEnabled() : bool
TemperatureregulatorWithBlinds
+ istBlindeinstellungRelevant() : bool
RoomLightControllerWithBlinds
Blind
Manueller Modus zuruecksetzen
PRBlindController
- zeitspanneVerlassen : durationType = 8
+ setzeZeitspanneVerlassen(Zeitspanne:durationType) : void
RoomTemperatureController
+ abfrageMinimalZimmertemperatur() : tempType
+ abfrageMittelwertZimmertemperatur() : tempType
abfrageMinimum
RoomTemperatureSensor
RadiatorController
- regelungsVentilEinstellung : percentageType
- wasserTempAlt : tempType
- ventilFuerFrostschutzEinstellen() : void
- ventilFuerNormalbetriebEinstellen() : void
- getWaterTemperature() : tempType
+ ventilOeffnen() : void
+ ventilSchliessen() : void
+ ventilRegelung(zielTemp:tempType, zimmerTemp:tempType) : void
+ ventilRegelungVorbereiten() : void
ValueInterfaceWithBlinds
+ SetValues(OpModus:modeType, AbwesenheitTemp:tempType, AnwesenheitTemp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType) : void
ValueInterfaceWithoutBlinds
ValueInterface
+ SetValues(OpModus:modeType, AbwesenheitTemp:tempType, AnwesenheitTemp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType) : void
WaterTemperatureSensor
RadiatorValve

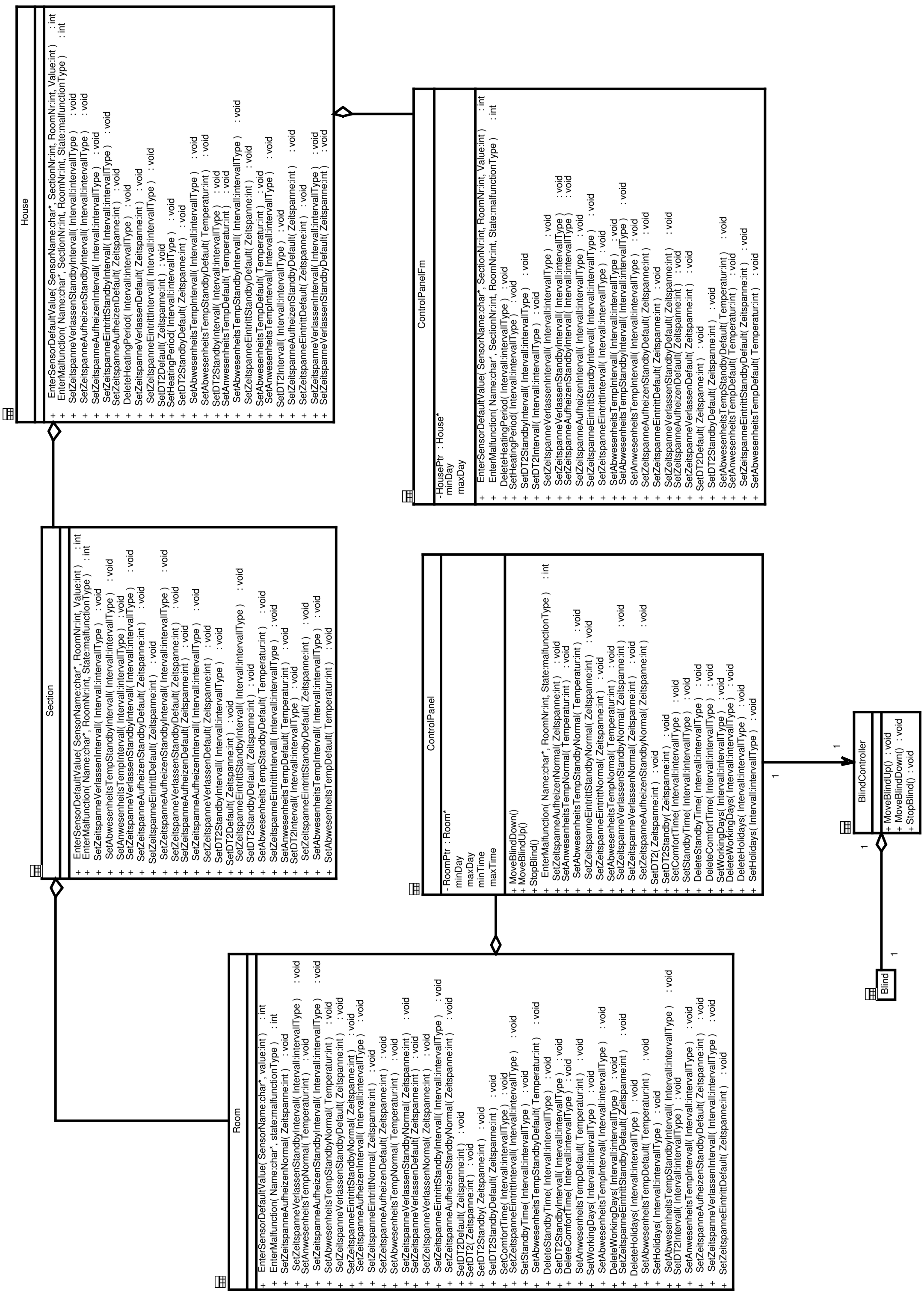
2.2.2.3 Hardware_Wrapper

Dieses Diagramm zeigt alle Klassen, die Sensoren oder Aktoren repräsentieren.



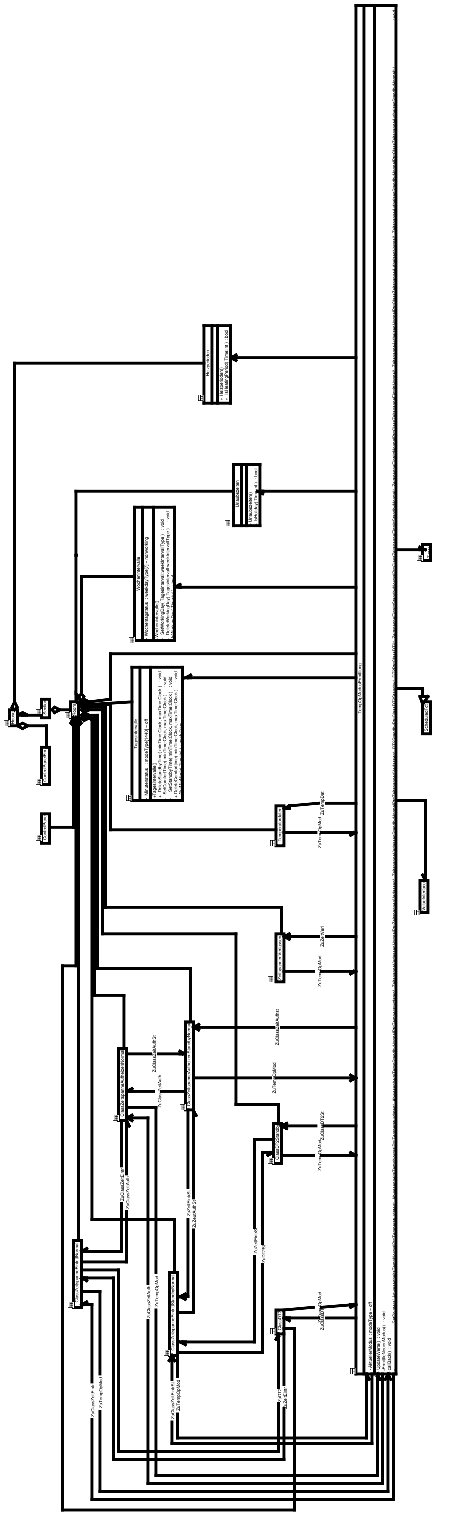
2.2.2.4 Benutzungsschnittstelle

Dieses Diagramm zeigt das Benutzungsschnittstellen-Subsystem.



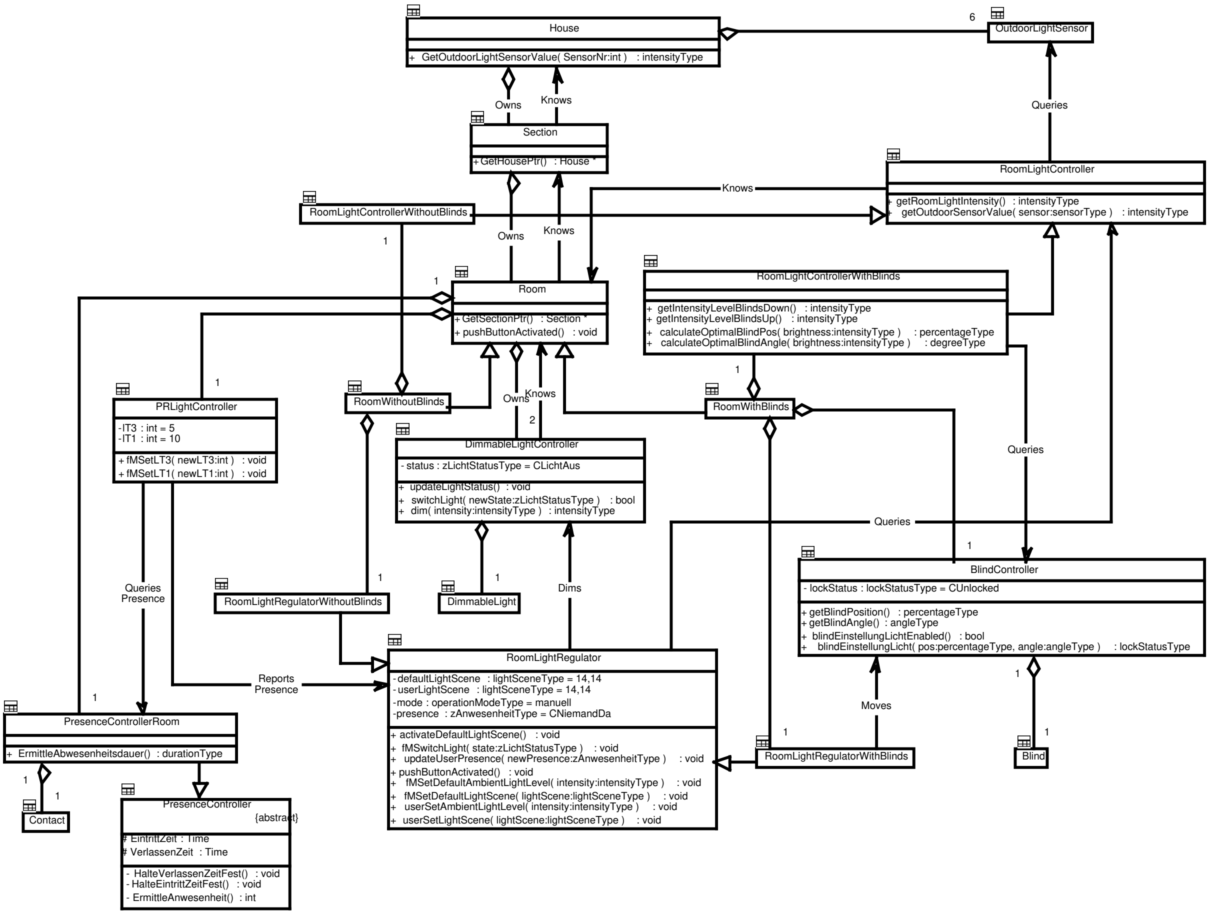
2.2.2.5 Systemmodusermittlung

Klassendiagramm des Teilsystems Systemmodusermittlung



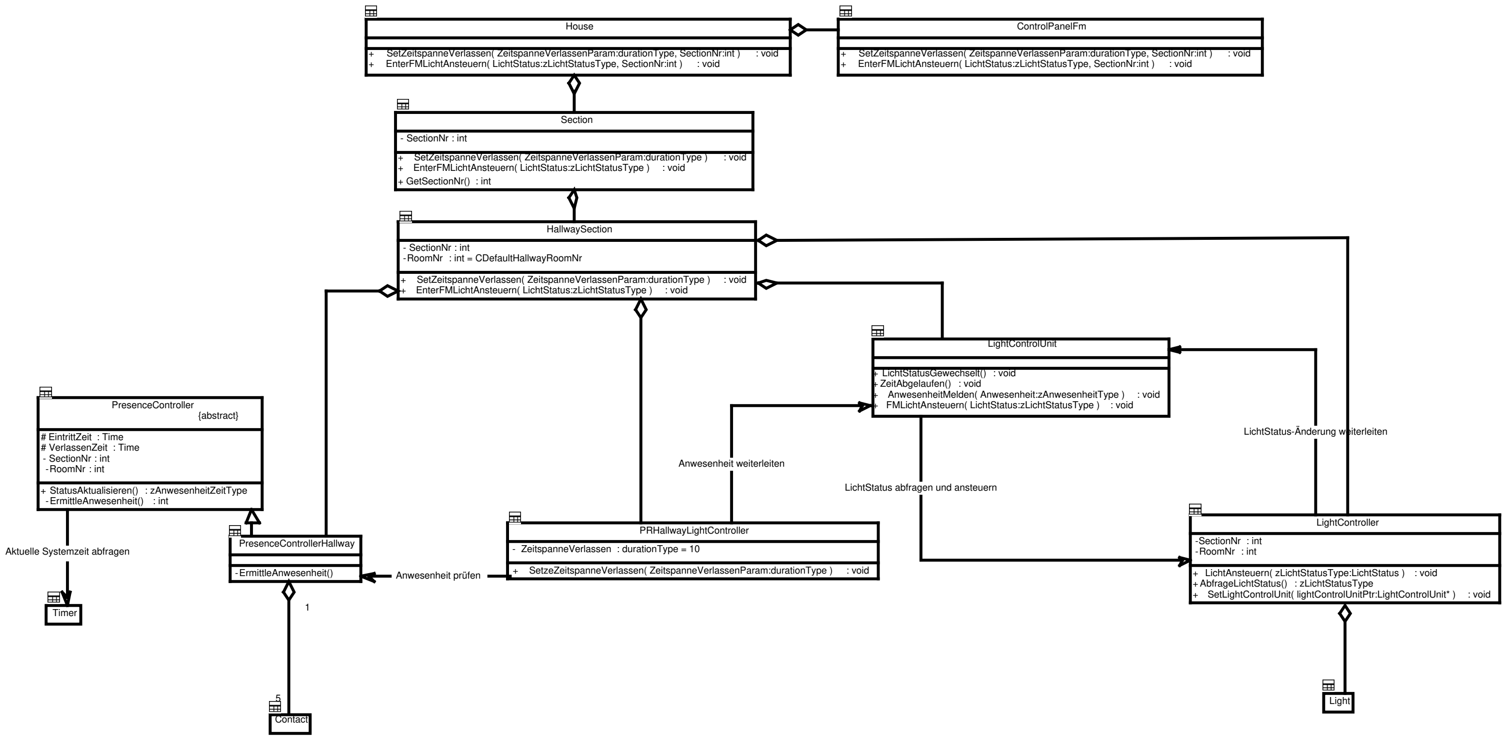
2.2.2.6 LichtregelungRaum

Dieses Diagramm zeigt alle Klassen, die an der Lichtregelung für der Raum beteiligt sind.

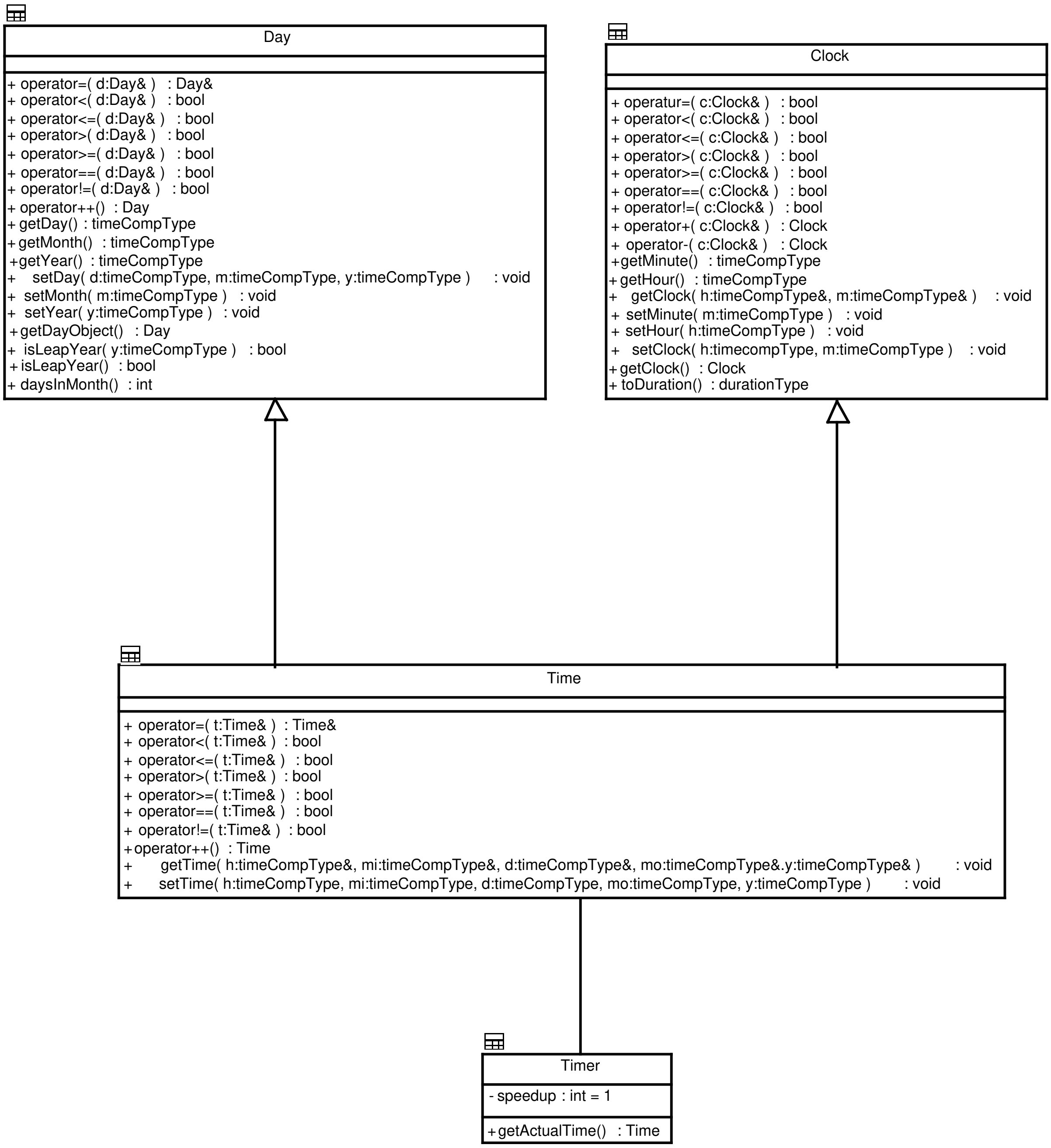


2.2.2.7 LichtregelungFlur

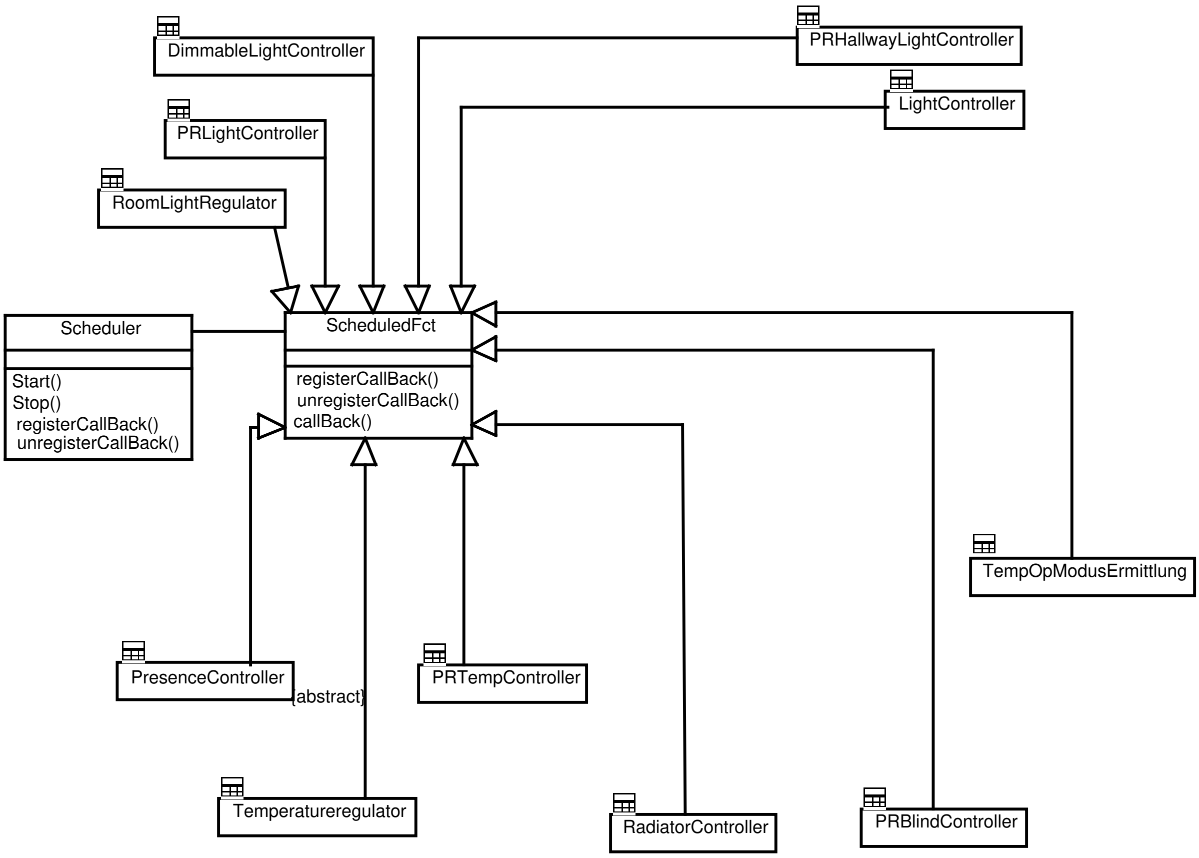
Dieses Diagramm beschreibt das Teilsystem Flurlichtregelung, in dem das Verhalten der Lichtanlagen der Flurabschnitte modelliert wird. Die Hauptklassen dieses Teilsystems sind die Klasse LightControlUnit (zur Verwaltung der Anwesenheitsinformationen und der Ansteuerung der zugeordneten Lichtanlage)j, die Klasse PresenceControllerHallway (zur Ermittlung und Verknüpfung der Zustände der fünf Anwesenheitskontakte im Flurabschnitt), die Klasse PRHallwayLightController (zur Vermittlung der Anwesenheitsinformation (Flurabschnitt leer oder besetzt) an die LightControlUnit) sowie die Klasse LightController (zur direkten Ansteuerung der Lichtanlage).


House
ControlPanelFm
Section
HallwaySection
LightControlUnit
PresenceController
{abstract}
PresenceControllerHallway
PRHallwayLightController
LightController
Timer
Contact
Light
Aktuelle Systemzeit abfragen
Anwesenheit prüfen
Anwesenheit weiterleiten
LichtStatus abfragen und ansteuern
LichtStatus-Änderung weiterleiten

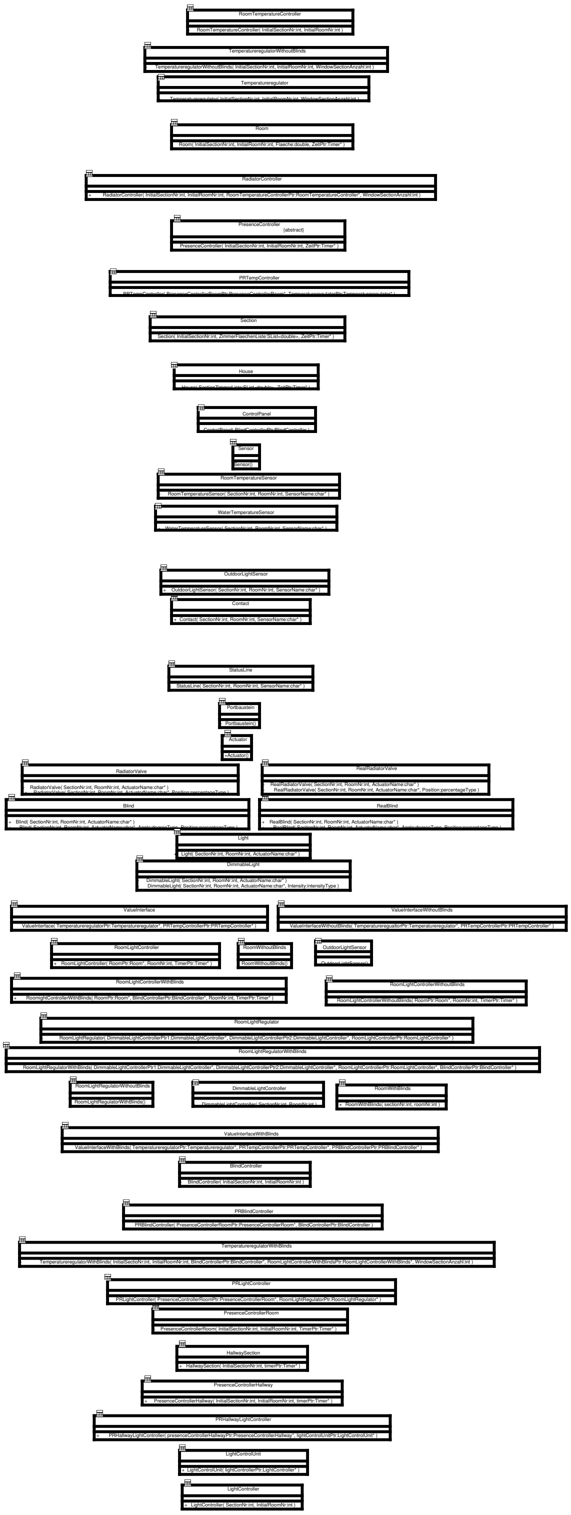
2.2.2.8 Zeit


Day
+ operator=(d:Day&) : Day&
+ operator<(d:Day&) : bool
+ operator<=(d:Day&) : bool
+ operator>(d:Day&) : bool
+ operator>=(d:Day&) : bool
+ operator==(d:Day&) : bool
+ operator!=(d:Day&) : bool
+ operator++() : Day
+ getDay() : timeCompType
+ getMonth() : timeCompType
+ getYear() : timeCompType
+ setDay(d:timeCompType, m:timeCompType, y:timeCompType) : void
+ setMonth(m:timeCompType) : void
+ setYear(y:timeCompType) : void
+ getDayObject() : Day
+ isLeapYear(y:timeCompType) : bool
+ isLeapYear() : bool
+ daysInMonth() : int
Clock
+ operatur=(c:Clock&) : bool
+ operator<(c:Clock&) : bool
+ operator<=(c:Clock&) : bool
+ operator>(c:Clock&) : bool
+ operator>=(c:Clock&) : bool
+ operator==(c:Clock&) : bool
+ operator!=(c:Clock&) : bool
+ operator+(c:Clock&) : Clock
+ operator-(c:Clock&) : Clock
+ getMinute() : timeCompType
+ getHour() : timeCompType
+ getClock(h:timeCompType&, m:timeCompType&) : void
+ setMinute(m:timeCompType) : void
+ setHour(h:timeCompType) : void
+ setClock(h:timecompType, m:timeCompType) : void
+ getClock() : Clock
+ toDuration() : durationType
Time
+ operator=(t:Time&) : Time&
+ operator<(t:Time&) : bool
+ operator<=(t:Time&) : bool
+ operator>(t:Time&) : bool
+ operator>=(t:Time&) : bool
+ operator==(t:Time&) : bool
+ operator!=(t:Time&) : bool
+ operator++() : Time
+ getTime(h:timeCompType&, mi:timeCompType&, d:timeCompType&, mo:timeCompType&.y:timeCompType&) : void
+ setTime(h:timeCompType, mi:timeCompType, d:timeCompType, mo:timeCompType, y:timeCompType) : void
Timer
- speedup : int = 1
+ getActualTime() : Time

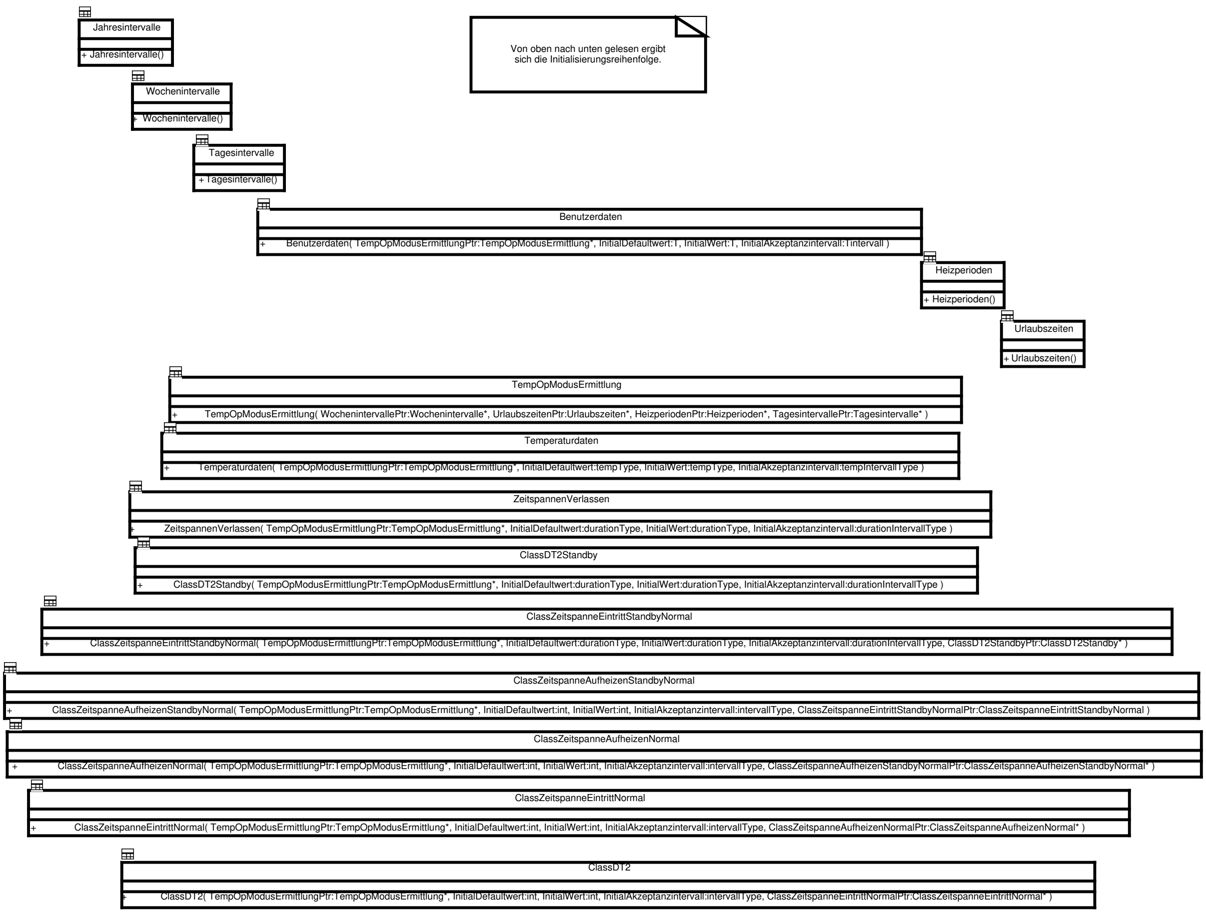
2.2.2.9 Scheduler



2.2.2.10 Konstruktoren

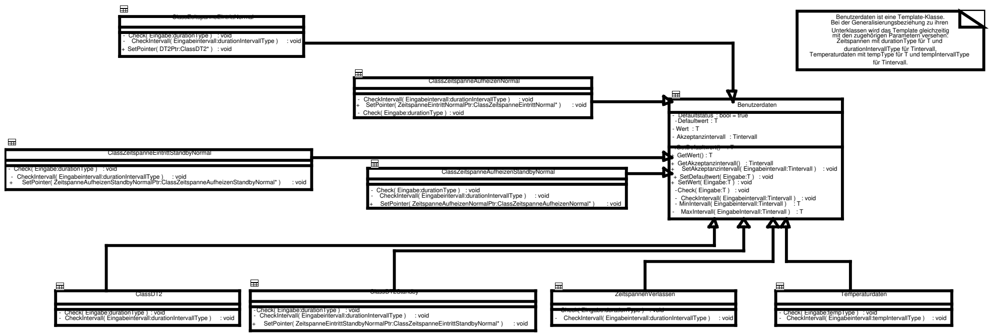


2.2.2.11 SystemmodusermittlungKonstruktoren

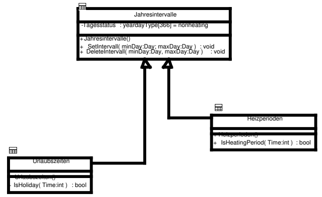


2.2.2.12 SystemmodusermittlungVererbung

Klassendiagramm des Teilsystems Systemmodusermittlung



Check(Eingabe:durationType) : void
CheckIntervall(Eingabeintervall:durationIntervallType) : void
SetPointer(DT2Ptr:ClassDT2*) : void
ClassZeitspanneAufheizenNormal
CheckIntervall(Eingabeintervall:durationIntervallType) : void
SetPointer(ZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*) : void
Check(Eingabe:durationType) : void
ClassZeitspanneEintrittStandbyNormal
Check(Eingabe:durationType) : void
CheckIntervall(Eingabeintervall:durationIntervallType) : void
SetPointer(ZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*) : void
ClassZeitspanneAufheizenStandbyNormal
Check(Eingabe:durationType) : void
CheckIntervall(Eingabeintervall:durationIntervallType) : void
SetPointer(ZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*) : void
Benutzerdaten
Defaultstatus : bool = true
Defaultwert : T
Wert : T
Akzeptanzintervall : Tintervall
GetWert() : T
GetAkzeptanzintervall() : Tintervall
SetAkzeptanzintervall(Eingabeintervall:Tintervall) : void
SetDefaultwert(Eingabe:T) : void
SetWert(Eingabe:T) : void
Check(Eingabe:T) : void
CheckIntervall(Eingabeintervall:Tintervall) : void
MinIntervall(Eingabeintervall:Tintervall) : T
MaxIntervall(EingabeIntervall:Tintervall) : T
Benutzerdaten ist eine Template-Klasse.
Bei der Generalisierungsbeziehung zu ihren
Unterklassen wird das Template gleichzeitig
mit den zugehörigen Parametern versehen:
Zeitspannen mit durationType für T und
durationIntervallType für Tintervall,
Temperaturdaten mit tempType für T und tempIntervallType
für Tintervall.
ClassDT2
Check(Eingabe:durationType) : void
CheckIntervall(Eingabeintervall:durationIntervallType) : void
Check(Eingabe:durationType) : void
CheckIntervall(Eingabeintervall:durationIntervallType) : void
SetPointer(ZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal*) : void
ZeitspannenVerlassen
CheckIntervall(Eingabeintervall:durationIntervallType) : void
Temperaturdaten
Check(Eingabe:tempType) : void
CheckIntervall(Eingabeintervall:tempIntervallType) : void



Jahresintervalle
-Tagesstatus : yeardayType[366] = nonheating
+Jahresintervalle()
+ SetIntervall(minDay:Day; maxDay:Day) : void
+ DeleteIntervall(minDay:Day, maxDay:Day) : void
Heizperioden
+ IsHeatingPeriod(Time:int) : bool
Urlaubszeiten
IsHoliday(Time:int) : bool

2.2.3 UML-Classtables

Nachfolgend werden die Classtables der einzelnen Klassen aufgeführt, die die Attribute und Methoden der jeweiligen Klasse darstellen. Zur Erläuterung dieser Atrribute und Methoden befindet sich direkt im Anschluß an jedes Classtable eine Beschreibung der Attribute und Methoden.

2.2.3.1 Classtable der Klasse Actuator

Actuator		
Attribute	**Type**	**Default Value**
OperationState	malfunctionType	
RoomNr	int	
SectionNr	int	
ActuatorName	char*	
Operation	**Arguments**	**Return Type**
IsOperatingNormal		int
SendEmail		void
SetOperationState	state:malfunctionType	void
Actuator		
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void

2.2.3.1.1 Attribute der Klasse Actuator

Attribut	Erläuterung
OperationState	Das Attribut OperationState gibt an, ob der Aktuator in Ordnung oder defekt ist.
RoomNr	Das Attribut RoomNr speichert die Nummer des Raumes, in dem sich der reale Aktuator befindet.
SectionNr	Das Attribut SectionNr speichert die Nummer der Section, in der sich der reale Aktuator befindet.
ActuatorName	Das Attribut Name speichert den Namen des Aktuators.

2.2.3.1.2 Methodenschnittstellen der Klasse Actuator

Signatur: **int IsOperatingNormal()**

Beschreibung: Die Methode IsOperatingNormal liefert den aktuellen OperationState des Aktuators zurueck.

Signatur: **void SetOperationState(state:malfunctionType)**

Beschreibung: Mit der Methode SetOperationState kann man den aktuellen OperationState des Aktuators setzen.

2.2.3.2 Classtable der Klasse Benutzerdaten

Benutzerdaten		
Attribute	**Type**	**Default Value**
Defaultstatus	bool	true
Defaultwert	T	
Wert	T	
Akzeptanzintervall	Tintervall	
Operation	**Arguments**	**Return Type**
SetAkzeptanzintervall	Eingabeintervall:Tintervall	void
SetDefaultwert	Eingabe:T	void
SetWert	Eingabe:T	void
GetDefaultwert		T

GetWert		T
GetAkzeptanzintervall		Tintervall
Check	Eingabe:T	void
CheckIntervall	Eingabeintervall:Tintervall	void
MinIntervall	Eingabeintervall:Tintervall	T
MaxIntervall	EingabeIntervall:Tintervall	T
Benutzerdaten	TempOpModusErmittlungPtr:TempOp-ModusErmittlung*, InitialDefault-wert:T, InitialWert:T, InitialAkzeptanzintervall:Tintervall	

2.2.3.2.1 Attribute der Klasse Benutzerdaten

Attribut	Erläuterung
Defaultstatus	Der Defaultstatus gibt an, ob der Benutzer für Wert schon eine eigene Einstellung gewählt (Defaultstatus 0) hat oder ob der Defaultwert noch gilt (Defaultstatus 1).
Defaultwert	Der vom Facility Manager eingestellte Defaultwert für Wert.
Wert	Der aktuelle eingestellte Wert.
Akzeptanzintervall	Das vom Facility Manager eingegebene Akzeptanzintervall für Wert.

2.2.3.2.2 Methodenschnittstellen der Klasse Benutzerdaten

Signatur: **T GetDefaultwert()**

Beschreibung: Liefert den Wert von Defaultwert.

Signatur: **T GetWert()**

Beschreibung: Liefert den Wert von Wert.

Signatur: **Tintervall GetAkzeptanzintervall()**

Beschreibung: Liefert die Größe von Akzeptanzintervall.

Signatur: **void SetAkzeptanzintervall(Eingabeintervall:Tintervall)**

Beschreibung: Setzt Akzeptanzintervall auf die vom Facility Manager eingegebene Größe.

Signatur: **void SetDefaultwert(Eingabe:T)**

Beschreibung: Setzt Defaultwert auf den vom Facility Manager eingegebenen Wert. Überprüft den Defaultstatus und setzt, falls dieser noch auf 1 steht (d.h. der Benutzer hat noch keinen eigenen Wert eingestellt) gleich noch den Wert neu.

Signatur: **void SetWert(Eingabe:T)**

Beschreibung: Setzt Wert auf den vom Benutzer eingebenen Wert falls dieser im Intervall Akzeptanzintervall liegt. Hier wird die Methode check() aufgerufen. Nach der erfolgreichen Überprüfung wird der Defaultstatus auf 0 gesetzt und ansonsten die Eingabe ignoriert.

2.2.3.3 Classtable der Klasse Blind

Blind		
Attribute	**Type**	**Default Value**
Angle	degreeType	
Position	percentageType	
Number	int	
RoomNr	int	
SectionNr	int	
OperationState	malfunctionType	
Operation	**Arguments**	**Return Type**
MoveUp		int
MoveDown		int
Stop		int
GetAngle		degreeType
GetPosition		percentageType
Up		int
Down		int
SetAngle	value:degreeType	degreeType
SetPosition	value:percentageType	percentageType
Blind	SectionNr:int, RoomNr:int, ActuatorName:char*	

Blind	SectionNr:int, RoomNr:int, ActuatorName:char*, Angle:degreeType, Position:percentageType	
IsOperatingNormal		int
SetOperationState	state:malfunctionType	void
Inherited Class::Operation	**Arguments**	**Return Type**

2.2.3.3.1 Attribute der Klasse Blind

Attribut	Erläuterung
Angle	Speichert den aktuellen Winkel der Sonnenblenden. (0 - 180 Grad)
Position	Speichert die aktuelle Position der Sonnenblende. (0 - 100%)
Number	Gibt die Anzahl der benötigten Blinds an.
RoomNr	Gibt die Zimmernummer an.
SectionNr	Gibt die Sektionsnummer an.
OperationState	Das Attribut OperationState gibt an, ob der Aktuator in Ordnung oder defekt ist.

2.2.3.3.2 Methodenschnittstellen der Klasse Blind

Signatur: **int MoveUp()**

Beschreibung: Nach Auruf dieser Methode werden die Blinds solange nach oben gefahren bis die Methode Stop() aufgerufen wird, oder die Blinds ganz oben sind.

Signatur: **int MoveDown()**

Beschreibung: Nach Aufruf dieser Methode werden die Blinds solange nach unten gefahren bis die Methode Stop() aufgerufen wird oder die Blinds ganz unten sind.

Signatur: **int Stop()**

Beschreibung: Wenn sich die Blinds bewegen können sie mit Hilfe der Methode Stop() gestoppt werden.

Signatur: **degreeType GetAngle()**

Beschreibung: Liefert den aktuellen Winkel der Sonnenblende zurueck. (0 - 180 Grad)

Signatur: **percentageType GetPosition()**

Beschreibung: Liefert die aktuelle Position der Sonnenblende zurueck. (0 - 100%)

Signatur: **int Up()**

Beschreibung: Fährt die Sonnenblende ganz nach oben. (0%)

Signatur: **int Down()**

Beschreibung: Fährt die Sonnenblende ganz nach unten. (100%)

Signatur: **degreeType SetAngle(value:degreeType)**

Beschreibung: Setzt den Winkel der Sonnenblende auf den angegebenen Wert.

Signatur: **percentageType SetPosition(value:percentageType)**

Beschreibung: Setzt die Sonnenblende auf die angegebene Position. (0 - 100%)

Signatur: **int IsOperatingNormal()**

Beschreibung: Die Methode IsOperatingNormal liefert den aktuellen OperationState des Aktuators zurueck.

Signatur: **void SetOperationState(state:malfunctionType)**

Beschreibung: Mit der Methode SetOperationState kann man den aktuellen OperationState des Aktuators setzen.

2.2.3.4 Classtable der Klasse BlindController

BlindController		
Attribute	**Type**	**Default Value**
lockStatus	lockStatusType	CUnlocked
Operation	**Arguments**	**Return Type**
blindEinstellungLicht	pos:percentage-Type, angle:angleType	lockStatusType

blindEinstellungTemp	pos:percentage-Type, angle:angleType	lockStatusType
blindUnlock		void
MoveBlindUp		void
MoveBlindDown		void
StopBlind		void
blindUnlockTemp		void
getBlindPosition		percentageType
getBlindAngle		angleType
blindEinstellungLichtEnabled		bool
blindEinstellungTempEnabled		bool
BlindController	InitialSectionNr:int, InitialRoomNr:int	
Inherited Class::Operation	**Arguments**	**Return Type**

2.2.3.4.1 Attribute der Klasse BlindController

Attribut	Erläuterung
lockStatus	Kann drei Werte haben, die festlegen, ob nur der User oder auch die Temperaturregelung und eventuell die Lichtregelung Zugriff auf die Blinds haben.

2.2.3.4.2 Methodenschnittstellen der Klasse BlindController

Signatur: **void MoveBlindUp()**

Beschreibung: Darueber stellt der Benutzer die Jalousien ein. Die Benutzung der Jalousien wird durch LockStatus fuer die Temperatur- und die Lichtregelung gesperrt.

Signatur: **void MoveBlindDown()**

Beschreibung: Darueber stellt der Benutzer die Jalousien ein. Die Benutzung der Jalousien wird durch LockStatus fuer die Temperatur- und die Lichtregelung gesperrt.

Signatur: **void StopBlind()**

Beschreibung: Diese Methode wird aufgeruden, wenn der Benutzer die sich bewegenden Blinds anhalten will.

Signatur: **void blindUnlock()**

Beschreibung: Gibt die Blinds fuer die Lichtregelung und und die Temperaturregelung frei (falls die Einstellung der Blinds durch eine Person verändert wurde), nachdem der Raum seit ZeitspanneVerlassen leer ist.

Signatur: **void blindUnlockTemp()**

Beschreibung: Gibt die Jalousien fuer die Lichtregelung frei, wenn die Temperaturregelung die Jalousien nicht mehr braucht.

Signatur: **lockStatusType blindEinstellungLicht(pos:percentageType, angle:angleType)**

Beschreibung: Stellt die Blinds im jeweiligen Raum auf eine vom RoomLightRegulator gewünschte Position. Hat niedrigste Priorität unter den Einstellungsfunktionen und wird durch jede andere Aktion blockiert.

Signatur: **lockStatusType blindEinstellungTemp(pos:percentageType, angle:angleType)**

Beschreibung: Die Temperaturregelung steuert die Blinds an, um zu heizen oder zu kuehlen (vorausgesetzt LockStatus erlaubt das).

Signatur: **percentageType getBlindPosition()**

Beschreibung: Liefert die aktuelle Position der Blinds im jeweiligen Raum.

Signatur: **angleType getBlindAngle()**

Beschreibung: Liefert den aktuell eingestellten Winkel der Blinds im jeweiligen Raum.

Signatur: **bool blindEinstellungLichtEnabled()**

Beschreibung: Liefert den aktuell eingestellten Winkel der Blinds im jeweiligen Raum.

Signatur: **bool blindEinstellungTempEnabled()**

Beschreibung: Liefert den aktuell eingestellten Winkel der Blinds im jeweiligen Raum.

2.2.3.5 Classtable der Klasse CentralHeatingSite

CentralHeatingSite		
Attribute	**Type**	**Default Value**
Kesseltemperatur		
Operation	**Arguments**	**Return Type**
KesseltemperaturSetzen	KesselTemp	

2.2.3.5.1 Attribute der Klasse CentralHeatingSite

Attribut	Erläuterung
Kesseltemperatur	Eingestellte Temperatur

2.2.3.5.2 Methodenschnittstellen der Klasse CentralHeatingSite

Die Klasse CentralHeatingSite besitzt keine Methodenschnittstellen.

2.2.3.6 Classtable der Klasse ClassDT2

ClassDT2		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void
ClassDT2	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*	

2.2.3.6.1 Attribute der Klasse ClassDT2

Keine.

2.2.3.6.2 Methodenschnittstellen der Klasse ClassDT2

Die Klasse ClassDT2 besitzt keine Methodenschnittstellen.

2.2.3.7 Classtable der Klasse ClassDT2Standby

ClassDT2Standby		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void
ClassDT2Standby	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:durationType, InitialWert:durationType, InitialAkzeptanzintervall:durationIntervallType	
SetPointer	ZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal*	void

2.2.3.7.1 Attribute der Klasse ClassDT2Standby

Keine.

2.2.3.7.2 Methodenschnittstellen der Klasse ClassDT2Standby

Signatur: **void SetPointer(ZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.8 Classtable der Klasse ClassZeitspanneAufheizenNormal

ClassZeitspanneAufheizenNormal		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void
ClassZeitspanneAufheizenNormal	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*	
SetPointer	ZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*	void

2.2.3.8.1 Attribute der Klasse ClassZeitspanneAufheizenNormal

Keine.

2.2.3.8.2 Methodenschnittstellen der Klasse ClassZeitspanneAufheizenNormal

Signatur: **void SetPointer(ZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.9 Classtable der Klasse ClassZeitspanneAufheizenStandbyNormal

ClassZeitspanneAufheizenStandbyNormal		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**

Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void
ClassZeitspanne-AufheizenStandby-Normal	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal	
SetPointer	ZeitspanneAufheizenNormalPtr:ClassZeitspanne-AufheizenNormal*	void

2.2.3.9.1 Attribute der Klasse ClassZeitspanneAufheizenStandbyNormal

Keine.

2.2.3.9.2 Methodenschnittstellen der Klasse ClassZeitspanneAufheizenStandbyNormal

Signatur: **void SetPointer(ZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.10 Classtable der Klasse ClassZeitspanneEintrittNormal

ClassZeitspanneEintrittNormal		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void

ClassZeitspanneEintrittNormal	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*	
SetPointer	DT2Ptr:ClassDT2*	void

2.2.3.10.1 Attribute der Klasse ClassZeitspanneEintrittNormal

Keine.

2.2.3.10.2 Methodenschnittstellen der Klasse ClassZeitspanneEintrittNormal

Signatur: **void SetPointer(DT2Ptr:ClassDT2*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.11 Classtable der Klasse ClassZeitspanneEintrittStandbyNormal

ClassZeitspanneEintrittStandbyNormal		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationIntervallType	void
ClassZeitspanneEintrittStandbyNormal	TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:durationType, InitialWert:durationType, InitialAkzeptanzintervall:durationIntervallType, ClassDT2StandbyPtr:ClassDT2Standby*	
SetPointer	ZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*	void

2.2.3.11.1 Attribute der Klasse ClassZeitspanneEintrittStandbyNormal

Keine.

2.2.3.11.2 Methodenschnittstellen der Klasse ClassZeitspanneEintrittStandbyNormal

Signatur: **void SetPointer(ZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.12 Classtable der Klasse Clock

Clock		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
toDuration		durationType
getClock		Clock
setClock	h:timecompType, m:timeCompType	void
setHour	h:timeCompType	void
setMinute	m:timeCompType	void
getClock	h:timeCompType&, m:timeCompType&	void
getHour		timeCompType
getMinute		timeCompType
operator-	c:Clock&	Clock
operator+	c:Clock&	Clock
operator!=	c:Clock&	bool
operator==	c:Clock&	bool
operator>=	c:Clock&	bool
operator>	c:Clock&	bool

operator<=	c:Clock&	bool
operator<	c:Clock&	bool
operatur=	c:Clock&	bool

2.2.3.12.1 Attribute der Klasse Clock

Keine.

2.2.3.13 Classtable der Klasse Contact

Contact		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
AbfrageZustand		int
SetDefaultValue	value:int	int
Contact	SectionNr:int, RoomNr:int, SensorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Sensor::GetDefaultValue		int
Sensor::IsOperatingNormal		int
Sensor::SendEmail		int
Sensor::SetDefaultValue	value:int	int

Sensor::SetOperationState	state:int	
Sensor::Sensor		

2.2.3.13.1 Attribute der Klasse Contact

Keine.

2.2.3.13.2 Methodenschnittstellen der Klasse Contact

Signatur: **int AbfrageZustand()**

Beschreibung: Liefert den aktuellen Zustand des realen Kontaktes zurück.

Signatur: **int SetDefaultValue(value:int)**

Beschreibung: Implementiert die abstrakte Methode der Klasse Sensor. Setzt den Defaultvalue des Sensors. Zulaessiger Bereich: 0,1.
Aus Verfolgbarkeitsgruenden zum vorherigen System wird an Stelle des Typs bool weiterhin der Typ int verwendet.

2.2.3.14 Classtable der Klasse ControlPanel

ControlPanel		
Attribute	**Type**	**Default Value**
RoomPtr	Room*	
Operation	**Arguments**	**Return Type**
EnterMalfunction	Name:char*, RoomNr:int, State:malfunctionType	int
MoveBlindDown		
MoveBlindUp		
StopBlind		
SetZeitspanneAufheizenNor-mal	Zeitspanne:int	void
SetAnwesenheitsTempNormal	Temperatur:int	void

SetAbwesenheitsTempStand-byNormal	Temperatur:int	void
SetZeitspanneEintrittStand-byNormal	Zeitspanne:int	void
SetZeitspanneEintrittNormal	Zeitspanne:int	void
SetAbwesenheitsTempNormal	Temperatur:int	void
SetZeitspanneVerlassenStand-byNormal	Zeitspanne:int	void
SetZeitspanneVerlassenNormal	Zeitspanne:int	void
SetZeitspanneAufheizenStand-byNormal	Zeitspanne:int	void
SetDT2	Zeitspanne:int	void
SetDT2Standby	Zeitspanne:int	void
SetComfortTime	Intervall:intervallType	void
SetStandbyTime	Intervall:intervallType	void
DeleteStandbyTime	Intervall:intervallType	void
DeleteComfortTime	Intervall:intervallType	void
SetWorkingDays	Intervall:intervallType	void
DeleteWorkingDays	Intervall:intervallType	void
DeleteHolidays	Intervall:intervallType	void
SetHolidays	Intervall:intervallType	void
SetRoomLightAmbientLight-Level	Level:intensityType	bool
SetRoomLightScene	LS:lightSceneType	bool

2.2.3.14.1 Attribute der Klasse ControlPanel

Attribut	Erläuterung
RoomPtr	
minDay	
maxDay	
minTime	
maxTime	

2.2.3.14.2 Methodenschnittstellen der Klasse ControlPanel

Signatur:

Beschreibung: Der Benutzer will eine (ausgewaehlte) Jalousie nach unten bewegen. Dies wird an den BlindController weitergegeben.

Signatur:

Beschreibung: Der Benutzer will eine Jalousie nach oben bewegen. Dieses wird an den BlindController weitergegeben.

Signatur:

Beschreibung: Diese Methode wird aufgerufen, wenn der Benutzer die sich bewegenden Blinds anhalten will.

Signatur: **int EnterMalfunction(Name:char*, RoomNr:int, State:malfunctionType)**

Beschreibung: Ruft die Methode SetOperationStatus() der entsprechenden Sensorklasse mit
true : zur Eingabe einer Malfunction und
false: zum Aufheben einer Malfunction
auf.

Signatur: **void SetZeitspanneAufheizenNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAnwesenheitsTempNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAbwesenheitsTempNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneVerlassenStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneAufheizenStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetDT2(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Standby(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetComfortTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void SetStandbyTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void DeleteStandbyTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void DeleteComfortTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void SetWorkingDays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Wochenintervalle auf.

Signatur: **void DeleteWorkingDays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Wochenintervalle auf.

Signatur: **void DeleteHolidays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Urlaubszeiten auf.

Signatur: **void SetHolidays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Urlaubszeiten auf.

Signatur: **bool SetRoomLightAmbientLightLevel(Level:intensityType)**

Beschreibung: Dient zum Setzen des AmbientLightLevel im jeweiligen Raum. Gibt zurück, ob erfolgreich.

Signatur: **bool SetRoomLightScene(LS:lightSceneType)**

Beschreibung: Dient zum Setzen der LightScene im jeweiligen Raum. Gibt zurück, ob erfolgreich.

2.2.3.15 Classtable der Klasse ControlPanelFm

ControlPanelFm		
Attribute	**Type**	**Default Value**
HousePtr	House*	
Operation	**Arguments**	**Return Type**
EnterMalfunction	Name:char*, SectionNr:int, RoomNr:int, State:malfunctionType	int
EnterSensorDefaultValue	SensorName:char*, SectionNr:int, RoomNr:int, Value:int	int
EnterFMLichtAnsteuern	LichtStatus:zLichtStatusType, SectionNr:int	void
SetAbwesenheitsTempStandbyDefault	Temperatur:int	void
SetAnwesenheitsTempDefault	Temperatur:int	void
SetZeitspanneEintrittStandbyDefault	Zeitspanne:int	void
SetAbwesenheitsTempDefault	Temperatur:int	void
SetZeitspanneAufheizenStandbyDefault	Zeitspanne:int	void
SetZeitspanneEintrittDefault	Zeitspanne:int	void
SetZeitspanneVerlassenStandbyDefault	Zeitspanne:int	void
SetZeitspanneAufheizenDefault	Zeitspanne:int	void
SetZeitspanneVerlassenDefault	Zeitspanne:int	void

SetAnwesenheitsTempIntervall	Intervall:intervallType	void
SetAbwesenheitsTempStandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempIntervall	Intervall:intervallType	void
SetZeitspanneEintrittIntervall	Intervall:intervallType	void
SetZeitspanneEintrittStandbyIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenStandbyIntervall	Intervall:intervallType	void
SetZeitspanneVerlassenStandbyIntervall	Intervall:intervallType	void
SetZeitspanneVerlassenIntervall	Intervall:intervallType	void
SetZeitspanneVerlassen	ZeitspanneVerlassenParam:durationType, SectionNr:int	void
SetDT2Default	Zeitspanne:int	void
SetDT2Intervall	Intervall:intervallType	void
SetDT2StandbyDefault	Zeitspanne:int	void
SetDT2StandbyIntervall	Intervall:intervallType	void
SetHeatingPeriod	Intervall:intervallType	void
DeleteHeatingPeriod	Intervall:intervallType	void
SetRoomLightLT3FM	Zeitspanne:int, RoomNr:int, SectionNr:int	bool
SetRoomLightLT1FM	Zeitspanne:int, RoomNr:int, SectionNr:int	bool
SetRoomLightDefaultAmbientLightLevelFM	Level:intensityType, RoomNr:int, SectionNr:int	bool
SetRoomLightDefaultLightSceneFM	LS:lightSceneType, RoomNr:int, SectionNr:int	bool
SwitchRoomLightFM	SwitchOn:bool, RoomNr:int, SectionNr:int	bool

2.2.3.15.1 Attribute der Klasse ControlPanelFm

Attribut	Erläuterung
HousePtr	

Attribut	Erläuterung
minDay	
maxDay	

2.2.3.15.2 Methodenschnittstellen der Klasse ControlPanelFm

Signatur: **int EnterSensorDefaultValue(SensorName:char*, SectionNr:int, RoomNr:int, Value:int)**

Beschreibung: Ruft die Methode SetDefaultValue() der jeweiligen Sensorklasse auf um den Default Value zu setzen.

Signatur: **int EnterMalfunction(Name:char*, SectionNr:int, RoomNr:int, State:malfunctionType)**

Beschreibung: Ruft die Methode SetOperationStatus() der entsprechenden Sensorklasse mit
true : zum setzen einer Malfunction und
false: zum Aufheben einer Malfunction
auf.

Signatur: **void DeleteHeatingPeriod(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Heizperioden auf.

Signatur: **void SetHeatingPeriod(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Heizperioden auf.

Signatur: **void SetDT2StandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Intervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetZeitspanneAufheizenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAbwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAnwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneAufheizenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetZeitspanneAufheizenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Default(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2StandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetAbwesenheitsTempStandbyDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAnwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAbwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **bool SetRoomLightLT3FM(Zeitspanne:int, RoomNr:int, SectionNr:int)**

Beschreibung: Dient zum Setzen des LT3-Wertes im PRLightController. Gibt zurück, ob erfolgreich.

Signatur: **bool SetRoomLightLT1FM(Zeitspanne:int, RoomNr:int, SectionNr:int)**

Beschreibung: Dient zum Setzen des LT1-Wertes im PRLightController. Gibt zurück, ob erfolgreich.

Signatur: **bool SetRoomLightDefaultAmbientLightLevelFM(Level:intensityType, RoomNr:int, SectionNr:int)**

Beschreibung: Dient zum Setzen des DefaultAmbientLightLevel im RoomLightRegulator. Gibt zurück, ob erfolgreich.

Signatur: **bool SetRoomLightDefaultLightSceneFM(LS:lightSceneType, RoomNr:int, SectionNr:int)**

Beschreibung: Dient zum Setzen der DefaultLightScene im RoomLightRegulator. Gibt zurück, ob erfolgreich.

Signatur: **bool SwitchRoomLightFM(SwitchOn:bool, RoomNr:int, SectionNr:int)**

Beschreibung: Ermöglicht dem Hausmeister, Licht in einem Raum ein- bzw. auszuschalten. Gibt zurück, ob erfolgreich.

Signatur: **void EnterFMLichtAnsteuern(LichtStatus:zLichtStatusType, SectionNr:int)**

Beschreibung: Die Methode EnterFMLichtAnsteuern() ruft die Methode House.EnterFMLichtAnsteuern() auf und übergibt ihr einen Wert LichtStatus vom Typ zLichtStatusType und die SectionNr des gewählten Flurabschnitts.
Diese Methode wird zum Durchreichen des Wertes LichtStatus vom ControlPanelFM-Objekt zum LightControlUnit-Objekt des entsprechenden Flurabschnitts benötigt.

Signatur: **void SetZeitspanneVerlassen(ZeitspanneVerlassenParam:durationType, SectionNr:int)**

Beschreibung: Die Methode SetZeitspanneVerlassen() ruft die Methode House.SetZeitspanneVerlassen() auf und übergibt ihr einen Wert ZeitspanneVerlassen vom Typ durationType und die SectionNr des gewählten Flurabschnitts.
Diese Methode wird zum Durchreichen des Wertes ZeitspanneVerlassen vom ControlPanelFM-Objekt zum PRHallwayLightController-Objekt des entsprechenden Flurabschnitts benötigt.

2.2.3.16 Classtable der Klasse Day

Day		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
daysInMonth		int
isLeapYear		bool
isLeapYear	y:timeCompType	bool
getDayObject		Day
setYear	y:timeCompType	void
setMonth	m:timeCompType	void
setDay	d:timeCompType, m:timeCompType, y:timeCompType	void
getYear		timeCompType

getMonth		timeCompType
getDay		timeCompType
operator++		Day
operator!=	d:Day&	bool
operator==	d:Day&	bool
operator>=	d:Day&	bool
operator>	d:Day&	bool
operator<=	d:Day&	bool
operator<	d:Day&	bool
operator=	d:Day&	Day&

2.2.3.16.1 Attribute der Klasse Day

Keine.

2.2.3.17 Classtable der Klasse DimmableLight

DimmableLight		
Attribute	**Type**	**Default Value**
Intensity	percentageType	
Operation	**Arguments**	**Return Type**
GetIntensity		percentageType
SetIntensity	value:intensityType	percentageType
DimmableLight	SectionNr:int, RoomNr:int, ActuatorName:char*	
DimmableLight	SectionNr:int, RoomNr:int, ActuatorName:char*, Intensity:intensityType	
Inherited Class::Operation	**Arguments**	**Return Type**

Portbaustein::CallBitOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::Portbaustein		
Actuator::IsOperatingNormal		int
Actuator::SendEmail		void
Actuator::SetOperationState	state:int	void
Actuator::Actuator		
Actuator::CallBitOutputPort	value:int	void
Actuator::CallIntOutputPort	value:int	void
Light::SwitchOn		int
Light::SwitchOff		int
Light::IsOn		int
Light::Light	SectionNr:int, RoomNr:int, ActuatorName:char*	

2.2.3.17.1 Attribute der Klasse DimmableLight

Attribut	Erläuterung
Intensity	Dieses Attribut repraesentiert die prozentuale Lichtintensitaet (0-100%), die am Dimmer eingestellt wurde.

2.2.3.17.2 Methodenschnittstellen der Klasse DimmableLight

Signatur: **percentageType GetIntensity()**

Beschreibung: Methode zur Abfrage der Lichtintensitaet, also der Dimmereinstellung (0-100%).

Signatur: **percentageType SetIntensity(value:intensityType)**

Beschreibung: Methode zum setzen der Lichtintensitaet, also der Stellung des Dimmers (0-100%).

2.2.3.18 Classtable der Klasse DimmableLightController

DimmableLightController		
Attribute	**Type**	**Default Value**
status	zLichtStatusType	CLichtAus
Operation	**Arguments**	**Return Type**
dim	intensity:intensityType	intensityType
switchLight	newState:zLichtStatusType	bool
updateLightStatus		void
DimmableLightController	SectionNr:int, RoomNr:int	

2.2.3.18.1 Attribute der Klasse DimmableLightController

Attribut	Erläuterung
status	Verwaltet den aktuellen Status (An/Aus) des jeweiligen DimmableLight. Dieser wird periodisch mittels IsOn()'ermittelt.BeieinerÄnderungdesStatus"vonaussen"(d.h.derBenutzerhatdenPushButtonbetäti gt)wirdimRoomLightRegulator(überUmwegdurchRoom)dieMethode'pushButtonActivated()'aufgerufen.'

2.2.3.18.2 Methodenschnittstellen der Klasse DimmableLightController

Signatur: **intensityType dim(intensity:intensityType)**

Beschreibung: Ändert die Helligkeit des jeweiligen DimmableLight auf den angegebenen Intensitätswert.

Signatur: **bool switchLight(newState:zLichtStatusType)**

Beschreibung: Schaltet die Spannung am jeweiligen DimmableLight ein. Diese Methode ändert die 'Status'-Variable ohne einen 'PushButtonActivated()'-Aufruf zur Folge zu haben.

Signatur: **void updateLightStatus()**

Beschreibung: Informiert den Room (über dessen Methode 'pushButtonActivated()'), wenn die Lampen im jeweiligen Raum per PushButton vom Benutzer aus-/bzw. eingeschaltet wurden. Wird periodisch von callBack() aufgerufen.

2.2.3.19 Classtable der Klasse HallwaySection

HallwaySection		
Attribute	**Type**	**Default Value**
RoomNr	int	CDefaultHallwayRoomNr
SectionNr	int	
Operation	**Arguments**	**Return Type**
EnterFMLichtAnsteuern	LichtStatus:zLichtStatus-Type	void
SetZeitspanneVerlassen	ZeitspanneVerlassenPa-ram:durationType	void
HallwaySection	InitialSectionNr:int, timerPtr:Timer*	

2.2.3.19.1 Attribute der Klasse HallwaySection

Attribut	Erläuterung
RoomNr	Das Attribut RoomNr wird zur Erzeugung der vier DoorContact-Objekte und des Contact-Objekts, die die Anwesenheitskontakte im Flurabschnitt repräsentieren, und des Light-Objekts, das die Flurlichtanlage repräsentiert, benötigt. Da sich alle Anwesenheitskontakte und die Lichtanlage im Flurabschnitt befinden und nicht in einem Raum, wird RaumNr mit einem Wert initialisiert, der diesen Sachverhalt kennzeichnet. Die DoorContact-Objekte bzw. das Contact-Objekt und das Light-Objekt können eindeutig durch ihren Namen identifiziert werden.
SectionNr	Das Attribut SectionNr beinhaltet die Nummer des Flurabschnitts, dem das HallwaySection-Objekt zugeordnet ist.

2.2.3.19.2 Methodenschnittstellen der Klasse HallwaySection

Signatur: **void EnterFMLichtAnsteuern(LichtStatus:zLichtStatusType)**

Beschreibung: Die Methode EnterFMLichtAnsteuern() ruft die Methode LightControlUnit.FMLichtAnsteuern() auf und übergibt ihr einen Wert LichtStatus vom Typ zLichtStatusType.
Diese Methode wird zum Durchreichen des Wertes LichtStatus vom ControlPanelFM-Objekt zum LightControlUnit-Objekt des entsprechenden Flurabschnitts benötigt.

Signatur: **void SetZeitspanneVerlassen(ZeitspanneVerlassenParam:durationType)**

Beschreibung: Die Methode SetZeitspanneVerlassen() ruft die Methode PRHallwayLightController.SetzeZeitspanneVerlassen() auf und übergibt ihr einen Wert ZeitspanneVerlassen vom Typ durationType.
Die Methode wird zum Durchreichen des Wertes ZeitspanneVerlassen vom ControlPanelFM-Objekt zum PRHallwayLightController-Objekt des entsprechenden Flurabschnitts benötigt.

2.2.3.20 Classtable der Klasse HallwaySector

HallwaySector		
Attribute	**Type**	**Default Value**
SectionNr	int	
RoomNr	int	

Operation	Arguments	Return Type
HallwaySector	SectionNr:int	

2.2.3.20.1 Attribute der Klasse HallwaySector

Attribut	Erläuterung
SectionNr	Hält die Sektionnummer fest.
RoomNr	Keine Bedeutung: Die Einführung dieses Attributes ist nur durch die Hardware-Schnittstelle bedingt.

2.2.3.20.2 Methodenschnittstellen der Klasse HallwaySector

Die Klasse HallwaySector besitzt keine Methodenschnittstellen.

2.2.3.21 Classtable der Klasse Heizperioden

Heizperioden		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
IsHeatingPeriod	Time:int	bool
Heizperioden		

2.2.3.21.1 Attribute der Klasse Heizperioden

Keine.

2.2.3.21.2 Methodenschnittstellen der Klasse Heizperioden

Signatur: **bool IsHeatingPeriod(Time:int)**

Beschreibung: Überprüft ob der aktuelle Tag ein Tag innerhalb oder ein Tag außerhalb einer Heizperiode ist.

2.2.3.22 Classtable der Klasse House

House		
Attribute	**Type**	**Default Value**
Abschnittsanzahl		
Operation	**Arguments**	**Return Type**
KesselTempAktualisieren		
RegenHaus		
House	SectionZimmerListe:SList<double>, ZeitPtr:Timer*	
EnterFMLichtAnsteuern	LichtStatus:zLichtStatusType, SectionNr:int	void
EnterSensorDefaultValue	SensorName:char*, SectionNr:int, RoomNr:int, Value:int	int
EnterMalfunction	Name:char*, SectionNr:int, RoomNr:int, State:malfunctionType	int
SetZeitspanneVerlassen	ZeitspanneVerlassenParam:durationType, SectionNr:int	void
SetZeitspanneVerlassenIntervall	Intervall:intervallType	void
SetDT2Intervall	Intervall:intervallType	void
SetDT2StandbyIntervall	Intervall:intervallType	void
SetHeatingPeriod	Intervall:intervallType	void
DeleteHeatingPeriod	Intervall:intervallType	void
SetZeitspanneVerlassen-StandbyDefault	Zeitspanne:int	void
SetZeitspanneEintrittDefault	Zeitspanne:int	void
SetZeitspanneAufheizen-StandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempIntervall	Intervall:intervallType	void
SetAbwesenheitsTempStandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempIntervall	Intervall:intervallType	void

SetZeitspanneEintrittIntervall	Intervall:intervallType	void
SetZeitspanneEintrittStandbyIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenStandbyIntervall	Intervall:intervallType	void
SetZeitspanneVerlassenStandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempDefault	Temperatur:int	void
SetZeitspanneEintrittStandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempDefault	Temperatur:int	void
SetAbwesenheitsTempStandbyDefault	Temperatur:int	void
SetDT2StandbyDefault	Zeitspanne:int	void
SetDT2Default	Zeitspanne:int	void
SetZeitspanneVerlassenDefault	Zeitspanne:int	void
SetZeitspanneAufheizenDefault	Zeitspanne:int	void
DeleteIntervall		
SetRoomLightLT3FM	Zeitspanne:int, RoomNr:int, SectionNr:int	bool
SetRoomLightLT1FM	Zeitspanne:int, RoomNr:int, SectionNr:int	bool
SetRoomLightDefaultAmbientLightLevelFM	Level:intensityType, RoomNr:int, SectionNr:int	bool
SetRoomLightDefaultLightSceneFM	LS:lightSceneType, RoomNr:int, SectionNr:int	bool
SwitchRoomLightFM	State:zLichtStatusType, RoomNr:int, SectionNr:int	bool
GetOutdoorLightSensorValue	SensorNr:int	intensityType

2.2.3.22.1 Attribute der Klasse House

Attribut	Erläuterung
Abschnittsanzahl	Anzahl der Abschnitte

2.2.3.22.2 Methodenschnittstellen der Klasse House

Signatur:

Beschreibung: Meldet neu einsetzenden Regen an alle Abschnitte.

Signatur: **int EnterSensorDefaultValue(SensorName:char*, SectionNr:int, RoomNr:int, Value:int)**

Beschreibung: Ruft die Methode SetDefaultValue() der jeweiligen Sensorklasse auf um den Default Value zu setzen.

Signatur: **int EnterMalfunction(Name:char*, SectionNr:int, RoomNr:int, State:malfunctionType)**

Beschreibung: Ruft die Methode SetOperationStatus() der entsprechenden Sensorklasse mit
true : zum setzen einer Malfunction und
false: zum Aufheben einer Malfunction
auf.

Signatur: **void SetZeitspanneVerlassenIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetDT2Intervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetDT2StandbyIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetHeatingPeriod(Intervall:intervallType)**

Beschreibung:

Signatur: **void DeleteHeatingPeriod(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneVerlassenStandbyDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetZeitspanneEintrittDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetZeitspanneAufheizenStandbyDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetAnwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetAbwesenheitsTempStandbyIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetAbwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneEintrittIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneEintrittStandbyIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneAufheizenIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneAufheizenStandbyIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetZeitspanneVerlassenStandbyIntervall(Intervall:intervallType)**

Beschreibung:

Signatur: **void SetAbwesenheitsTempDefault(Temperatur:int)**

Beschreibung:

Signatur: **void SetZeitspanneEintrittStandbyDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetAnwesenheitsTempDefault(Temperatur:int)**

Beschreibung:

Signatur: **void SetAbwesenheitsTempStandbyDefault(Temperatur:int)**

Beschreibung:

Signatur: **void SetDT2StandbyDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetDT2Default(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetZeitspanneVerlassenDefault(Zeitspanne:int)**

Beschreibung:

Signatur: **void SetZeitspanneAufheizenDefault(Zeitspanne:int)**

Beschreibung:

Signatur:

Beschreibung:

Signatur: **bool SetRoomLightLT3FM(Zeitspanne:int, RoomNr:int, SectionNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightLT1FM(Zeitspanne:int, RoomNr:int, SectionNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultAmbientLightLevelFM(Level:intensityType, RoomNr:int, SectionNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultLightSceneFM(LS:lightSceneType, RoomNr:int, SectionNr:int)**

Beschreibung:

Signatur: **intensityType GetOutdoorLightSensorValue(SensorNr:int)**

Beschreibung: Liefert den Wert eines OutdoorLightSensors zurück. Sensoren müssen unter Angabe ihrer Nummer jeweils einzeln abgefragt werden.

Signatur: **bool SwitchRoomLightFM(State:zLichtStatusType, RoomNr:int, SectionNr:int)**

Beschreibung:

Signatur: **void EnterFMLichtAnsteuern(LichtStatus:zLichtStatusType, SectionNr:int)**

Beschreibung: Die Methode EnterFMLichtAnsteuern() ruft die Methode Section.EnterFMLichtAnsteuern() auf und übergibt ihr einen Wert LichtStatus vom Typ zLichtStatusType.
Diese Methode wird zum Durchreichen des Wertes LichtStatus vom ControlPanelFM-Objekt zur LightControlUnit-Objekt des entsprechenden Flurabschnitts benötigt.

Signatur: **void SetZeitspanneVerlassen(ZeitspanneVerlassenParam:durationType, SectionNr:int)**

Beschreibung: Die Methode SetZeitspanneVerlassen() ruft die Methode Section.SetZeitspanneVerlassen() auf und übergibt ihr einen Wert ZeitspanneVerlassen vom Typ durationType.
Diese Methode wird zum Durchreichen des Wertes ZeitspanneVerlassen vom ControlPanelFM-Objekt zum PRHallwayLightController-Objekt des entsprechenden Flurabschnitts benötigt.

2.2.3.23 Classtable der Klasse Jahresintervalle

Jahresintervalle		
Attribute	**Type**	**Default Value**
Tagesstatus	yeardayType[366]	nonheating
Operation	**Arguments**	**Return Type**
SetIntervall	minDay:Day; maxDay:Day	void
DeleteIntervall	minDay:Day, maxDay:Day	void
Jahresintervalle		

2.2.3.23.1 Attribute der Klasse Jahresintervalle

Attribut	Erläuterung
Tagesstatus	Hier ist für jeden Tag im Jahr gespeichert ob es ein Ferientag bzw. Nicht-Ferientag ist oder ob es ein Tag in der heating-period bzw. in der non-heating-period ist.

2.2.3.23.2 Methodenschnittstellen der Klasse Jahresintervalle

Signatur: **void SetIntervall(minDay:Day; maxDay:Day)**

Beschreibung: Hiermit setzt der Benutzer Ferienzeit bzw. der Facility Manager heating-period in Tagesstatus.

Signatur: **void DeleteIntervall(minDay:Day, maxDay:Day)**

Beschreibung: Hiermit löscht der Benutzer Ferienzeit bzw. der Facility Manager heating-period in Tagesstatus.

2.2.3.24 Classtable der Klasse Light

Light		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
SwitchOn		int
SwitchOff		int
IsOn		int
Light	SectionNr:int, RoomNr:int, ActuatorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void

Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Actuator::IsOperatingNormal		int
Actuator::SendEmail		
Actuator::SetOperationState	state:int	
Actuator::Actuator		

2.2.3.24.1 Attribute der Klasse Light

Keine.

2.2.3.24.2 Methodenschnittstellen der Klasse Light

Signatur: **int SwitchOn()**

Beschreibung: Diese Methode schaltet das Licht an.

Signatur: **int SwitchOff()**

Beschreibung: Diese Methode schaltet das Licht aus.

Signatur: **int IsOn()**

Beschreibung: Diese Methode ruft die Methode IsOn() der zugehoerigen StatusLine auf und liefert deren Wert zurueck.

2.2.3.25 Classtable der Klasse LightControlUnit

LightControlUnit		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**

LightControlUnit	lightControllerPtr:LightController*	
FMLichtAnsteuern	LichtStatus:zLichtStatusType	void
AnwesenheitMel-den	Anwesenheit:zAnwesenheitType	void
LichtStatusGewech-selt		void
ZeitAbgelaufen		void

2.2.3.25.1 Attribute der Klasse LightControlUnit

Keine.

2.2.3.25.2 Methodenschnittstellen der Klasse LightControlUnit

Signatur: **void LichtStatusGewechselt()**

Beschreibung: Die Methode LichtStatusGewechselt() wird vom LightController-Objekt aufgerufen, falls der Zustand der Lichtanlage im Flur sich geändert hat.

Signatur: **void ZeitAbgelaufen()**

Beschreibung: Über die Methode ZeitAbgelaufen meldet das PRHallwayLightController-Objekt dem LightControlUnit-Objekt, daß der Flur seit länger als ZeitVerlassen (LT2) Minuten leer ist.

Signatur: **void AnwesenheitMelden(Anwesenheit:zAnwesenheitType)**

Beschreibung: Durch die Methode AnwesenheitMelden() erfolgt die Benachrichtigung vom PRHallwayLightController-Objekt, daß jemand in den Flurabschnitt eingetreten ist bzw., daß alle Benutzer den Flurabschnitt verlassen haben.

Signatur: **void FMLichtAnsteuern(LichtStatus:zLichtStatusType)**

Beschreibung: Die Methode FMLichtAnsteuern() wird durch das HallwaySection-Objekt aufgerufen, falls der Hausmeister über das ControlPanelFM das Licht im Flurabschnitt ein- bzw. ausschalten möchte.

2.2.3.26 Classtable der Klasse LightController

LightController		
Attribute	**Type**	**Default Value**
SectionNr	int	
RoomNr	int	
Operation	**Arguments**	**Return Type**
LightController	SectionNr:int, InitialRoomNr:int	
LichtAnsteuern	zLichtStatusType:LichtStatus	void
AbfrageLichtStatus		zLichtStatusType
SetLightControlUnit	lightControlUnitPtr:LightControlUnit*	void

2.2.3.26.1 Attribute der Klasse LightController

Attribut	Erläuterung
SectionNr	Das Attribut SectionNr beinhaltet die Nummer des Flurabschnitts, dem das LightController-Objekt zugeordnet ist.
RoomNr	Das Attribut RoomNr wird zur Erzeugung eines Objekts der Klasse Light benötigt, das die Flurlichtanlage des Flurabschnitts repräsentiert. Da sich die Lichtanlage im Flurabschnitt befindet und nicht in einem Raum, wird RaumNr mit einem Wert initialisiert, der diesen Sachverhalt kennzeichnet. Die Lichtanlage kann eindeutig durch ihren Namen identifiziert werden.

2.2.3.26.2 Methodenschnittstellen der Klasse LightController

Signatur: **zLichtStatusType AbfrageLichtStatus()**

Beschreibung: Die Methode AbfrageLichtStatus() gibt den Status der Lichtanlage (Licht ein-/ausgeschaltet) zurück.

Signatur: **void LichtAnsteuern(zLichtStatusType:LichtStatus)**

Beschreibung: Die Methode LichtAnsteuern() ruft in Abhängigkeit von ihrem Parameter LichtStatus die Methoden SwitchOn() bzw. SwitchOff() eines Objekts der Klasse Light auf, um die Lichtanlage des Flurabschnitts entsprechend anzusteuern.

Signatur: **void SetLightControlUnit(lightControlUnitPtr:LightControlUnit*)**

Beschreibung: Die Methode SetLightControlUnit() dient der nachträglichen Übergabe eines Pointers auf ein Objekt der Klasse LightControlUnit. Die Notwendigkeit dieser Methode ergibt sich aus der Initialisierungsreihenfolge (das LightController-Objekt wird vor dem LightControlUnit-Objekt erzeugt) der Objekte der Klassen LightControlUnit und LightController.

2.2.3.27 Classtable der Klasse OutdoorLightSensor

OutdoorLightSensor		
Attribute	**Type**	**Default Value**
LowerIntensityLimit	intensityType	1
UpperIntensityLimit	intensityType	10000
Operation	**Arguments**	**Return Type**
GetIntensity		intensityType
SetDefaultValue	value:int	int
OutdoorLightSensor		
OutdoorLightSensor	SectionNr:int, RoomNr:int, SensorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Sensor::GetDefaultValue		int
Sensor::IsOperatingNormal		int
Sensor::SendEmail		int
Sensor::SetDefaultValue	value:int	int

Sensor::SetOperationState	state:int	
Sensor::Sensor		

2.2.3.27.1 Attribute der Klasse OutdoorLightSensor

Attribut	Erläuterung
LowerIntensityLimit	Das Attribut LowerIntensityLimit speichert die untere Grenze des gültigen Wertebereiches des realen Sensors.
UpperIntensityLimit	Das Attribut UpperIntensityLimit speichert den oberen Grenzwert des Wertebereiches des realen Sensors.

2.2.3.27.2 Methodenschnittstellen der Klasse OutdoorLightSensor

Signatur: **intensityType GetIntensity()**

Beschreibung: Liefert die aktuelle Lichintensitaet in lux zurueck und setzt diese im Attribut Intensity.

Signatur: **int SetDefaultValue(value:int)**

Beschreibung: Implementiert die abstrakte Methode der Klasse Sensor. Setzt den Defaultvalue des Sensors. Zulaessiger Bereich: 1 - 10000 lux.

2.2.3.28 Classtable der Klasse PRBlindController

PRBlindController		
Attribute	**Type**	**Default Value**
zeitspanneVerlassen	durationType	8
Operation	**Arguments**	**Return Type**
setzeZeitspanneVerlassen	Zeitspanne:durationType	void
PRBlindController	InitialSectionNr:int, InitialRoomNr:int, PresenceControllerRoomPtr:PresenceControllerRoom*, BlindControllerPtr:BlindController	

2.2.3.28.1 Attribute der Klasse PRBlindController

Attribut	Erläuterung
zeitspanneVerlassen	

2.2.3.28.2 Methodenschnittstellen der Klasse PRBlindController

Signatur: **void setzeZeitspanneVerlassen(Zeitspanne:durationType)**

Beschreibung: Setzt die ZeitspanneVerlassen

2.2.3.29 Classtable der Klasse PRHallwayLightController

PRHallwayLightController		
Attribute	**Type**	**Default Value**
ZeitspanneVer-lassen	durationType	10
Operation	**Arguments**	**Return Type**
PRHallway-LightController	presenceControllerHallwayPtr:PresenceControllerHallway*, lightControlUnitPtr:LightControlUnit*	
SetzeZeitspan-neVerlassen	ZeitspanneVerlassenParam:durationType	void

2.2.3.29.1 Attribute der Klasse PRHallwayLightController

Attribut	Erläuterung
ZeitspanneVerlassen	Das Attribut ZeitspanneVerlassen enthält den Wert LT2. Nachdem der Flur von allen Benutzern verlassen wurde, schaltet das System nach LT2 Minuten die Lichtanlage automatisch aus. Der Default-Wert dieser Variable beträgt 10 (Minuten).

2.2.3.29.2 Methodenschnittstellen der Klasse PRHallwayLightController

Signatur: **void SetzeZeitspanneVerlassen(ZeitspanneVerlassenParam:durationType)**

Beschreibung: Die Methode SetzeZeitspanneVerlassen() dient dem Setzen der Variablen Zeitspanne-Verlassen. Diese Methode wird vom HallwaySection-Objekt aufgerufen.

2.2.3.30 Classtable der Klasse PRLightController

PRLightController		
Attribute	**Type**	**Default Value**
lT3	int	5
lT1	int	10
Operation	**Arguments**	**Return Type**
fMSetLT3	newLT3:int	void
fMSetLT1	newLT1:int	void

2.2.3.30.1 Attribute der Klasse PRLightController

Attribut	Erläuterung
lT3	Zeitspanne in Minuten , die angibt, wie lange der Benutzer den Raum verlassen haben muss, bis das System regelnd eingreift.
lT1	Zeitspanne in Minuten, die angibt, wann die vom Benutzer gewählte LightScene nach Verlassen des Zimmers wieder auf den Defaultwert gesetzt wird.

2.2.3.30.2 Methodenschnittstellen der Klasse PRLightController

Signatur: **void fMSetLT3(newLT3:int)**

Beschreibung: Setzt die Zeitspanne LT3 neu. Darf nur vom Hausmeister benutzt werden.

Signatur: **void fMSetLT1(newLT1:int)**

Beschreibung: Setzt die Zeitspanne LT1 neu. Darf nur vom Hausmeister benutzt werden.

2.2.3.31 Classtable der Klasse PRTempController

PRTempController

Attribute	Type	Default Value
zeitspanneEintritt	durationType	10
zeitspanneVerlassen	durationType	5
dT2	durationType	10
Operation	**Arguments**	**Return Type**
setzeZeitspanneVerlassen	Zeitspanne:durationType	void
setzeZeitspanneEintritt	Zeitspanne:durationType	void
setzeDT2	Zeitspanne:durationType	void
PRTempController	PresenceControllerRoomPtr:PresenceControllerRoom*, TemperatureregulatorPtr:Temperatureregulator*	

2.2.3.31.1 Attribute der Klasse PRTempController

Attribut	Erläuterung
zeitspanneVerlassen	Wenn der Anwesenheitskontakt laenger als DT2 geschlossen war und wieder geoeffnet wird, muss mindestens ZeitspanneVerlassen vergehen (und der Kontakt nicht wieder geschlosssen werden), damit Anwesenheit auf falsch gesetzt wird.
zeitspanneEintritt	Wenn nach dem Schliessen des Anwesenheitskontakts diese Zeitspanne vergangen ist, wird die Anwesenheit im Temperaturregulator auf wahr gesetzt.
dT2	

2.2.3.31.2 Methodenschnittstellen der Klasse PRTempController

Signatur: **void setzeZeitspanneVerlassen(Zeitspanne:durationType)**

Beschreibung: Setzen des Attributs ZeitspanneVerlassenNormal auf Zeitspanne Minuten.

Signatur: **void setzeZeitspanneEintritt(Zeitspanne:durationType)**

Beschreibung: Setzen des Attributs ZeitspanneEintrittNormal auf Zeitspanne Minuten.

Signatur: **void setzeDT2(Zeitspanne:durationType)**

Beschreibung: Setzt den Wert von DT2.

2.2.3.32 Classtable der Klasse Portbaustein

Portbaustein		
Attribute	**Type**	**Default Value**
portHandlerPtr	PortHandler*	
Operation	**Arguments**	**Return Type**
CallBitInputPort	SectionNr:int, RoomNr:int, Sensor-Name:char*	int
CallBitOutputPort	SectionNr:int, RoomNr:int, Actuator-Name:char*, value:int	void
CallIntInputPort	SectionNr:int, RoomNr:int, Sensor-Name:char*	int
CallIntOutputPort	SectionNr:int, RoomNr:int, Actuator-Name:char*, value:int	void
Portbaustein		

2.2.3.32.1 Attribute der Klasse Portbaustein

Attribut	Erläuterung
portHandlerPtr	

2.2.3.32.2 Methodenschnittstellen der Klasse Portbaustein

Die Klasse Portbaustein besitzt keine Methodenschnittstellen.

2.2.3.33 Classtable der Klasse PresenceController

PresenceController		
Attribute	**Type**	**Default Value**
EintrittZeit	Time	
VerlassenZeit	Time	
SectionNr	int	
RoomNr	int	
Operation	**Arguments**	**Return Type**
StatusAktualisieren		zAnwesenheitZeitType
PresenceController	InitialSectionNr:int, InitialRoomNr:int	
HalteEintrittZeitFest		void
HalteVerlassenZeitFest		void
ErmittleAnwesenheit		int

2.2.3.33.1 Attribute der Klasse PresenceController

Attribut	Erläuterung
EintrittZeit	Zeitpunkt, seit dem ein Benutzer anwesend ist.
VerlassenZeit	Zeitpunkt, seit dem der letzte Benutzer das Zimmer verlassen hat.
SectionNr	Hält die Sektionnummer fest.
RoomNr	Keine Bedeutung: die Einführung dieses Attributes ist nur durch die Hardware-Schnittstelle bedingt.

2.2.3.33.2 Methodenschnittstellen der Klasse PresenceController

Signatur: **zAnwesenheitZeitType StatusAktualisieren()**

Beschreibung: Gibt den Wert des Anwesenheitskontaktes zurück und die Zeitspanne, wie lange sich eine Person im Zimmer befindet bzw. das Zimmer leer ist.

2.2.3.34 Classtable der Klasse PresenceControllerHallway

PresenceControllerHallway		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
PresenceControllerHallway	InitialSectionNr:int, InitialRoomNr:int, timerPtr:Timer*	
ErmittleAnwesenheit		
Inherited Class::Operation	**Arguments**	**Return Type**
ScheduledFct::callBack		
ScheduledFct::unregisterCall-Back		
ScheduledFct::registerCall-Back		
PresenceController::StatusAk-tualisieren		zAnwesenheit-ZeitType
PresenceController::Presence-Controller	InitialSectionNr:int, InitialRoomNr:int, ZeitPtr:Timer*	
PresenceController::Ermittle-Anwesenheit		int
PresenceController::HalteEin-trittZeitFest		void
PresenceController::HalteVer-lassenZeitFest		void
PresenceController::HoleVer-lassenZeit		Time

2.2.3.34.1 Attribute der Klasse PresenceControllerHallway

Keine.

2.2.3.34.2 Methodenschnittstellen der Klasse PresenceControllerHallway

Die Klasse PresenceControllerHallway besitzt keine Methodenschnittstellen.

2.2.3.35 Classtable der Klasse PresenceControllerRoom

PresenceControllerRoom		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
PresenceControllerRoom	InitialSectionNr:int, InitialRoomNr:int, TimerPtr:Timer*	
ErmittleAnwesenheit		int
ErmittleAbwesenheitsdauer		durationType
Inherited Class::Operation	**Arguments**	**Return Type**

2.2.3.35.1 Attribute der Klasse PresenceControllerRoom

Keine.

2.2.3.35.2 Methodenschnittstellen der Klasse PresenceControllerRoom

Signatur: **int ErmittleAnwesenheit()**

Beschreibung: Vererbt vom PresenceController: Fragt die Sensoren im Raum ab und liefert ihren Status.

Signatur: **durationType ErmittleAbwesenheitsdauer()**

Beschreibung:

2.2.3.36 Classtable der Klasse RadiatorController

RadiatorController		
Attribute	**Type**	**Default Value**
regelungsVentil-Einstellung	percentageType	
wasserTempAlt	tempType	
Operation	**Arguments**	**Return Type**
ventilFuerFrost-schutzEinstellen		void
ventilFuerNormal-betriebEinstellen		void
getWaterTempera-ture		tempType
RadiatorController	InitialSectionNr:int, InitialRoomNr:int, RoomTemperatureCon-trollerPtr:RoomTemperatureController*, WindowSectionAn-zahl:int	
ventilRegelung	zielTemp:tempType, zimmerTemp:tempType	void
ventilOeffnen		void
ventilSchliessen		void
ventilRegelung-Vorbereiten		void

2.2.3.36.1 Attribute der Klasse RadiatorController

Attribut	Erläuterung
regelungsVentilEinstellung	Speichert die durch die Temperaturregelung gewünschte Ventileinstellung, um sie nach Frostschutzregelung wieder herstellen zu können.
wasserTempAlt	Speichert den zuletzt gemessenen Wert der Heizkörpertemperatur.

2.2.3.36.2 Methodenschnittstellen der Klasse RadiatorController

Signatur: **void ventilOeffnen()**

Beschreibung: Öffnet das Ventil vollständig zum Heizen.

Signatur: **void ventilSchliessen()**

Beschreibung: Schließt das Ventil vollständig zum Kühlen.

Signatur: **void ventilRegelung(zielTemp:tempType, zimmerTemp:tempType)**

Beschreibung: Realisiert den Heizungsregelungsalgorithmus. Zu diesem Zweck erhält die Methode vom Temperatureregulator Ziel- und Zimmertemperatur übergeben und fragt die aktuelle Wassertemperatur des Heizkörpers ab. Die zuletzt gemessene Heizkörpertemperatur wird in einer Variablen wasserTempAlt festgehalten:

```
p=Zieltemperatur-Zimmertemperatur;
d=wasserTempAlt-aktuelle Wassertemperatur;
v=20*p+10*d;
v<-60=>v=-60;
v>60=>v=60;
RadiatorValve.SetPosition(RadiatorValve.GetPosition+v));
```

Signatur: **void ventilRegelungVorbereiten()**

Beschreibung:

2.2.3.37 Classtable der Klasse RadiatorValve

RadiatorValve		
Attribute	**Type**	**Default Value**
Position	percentageType	

Number	int	
RoomNr	int	
SectionNr	int	
OperationState	malfunctionType	
Operation	**Arguments**	**Return Type**
SetPosition	value:percentageType	percentageType
RadiatorValve	SectionNr:int, RoomNr:int, ActuatorName:char*	
RadiatorValve	SectionNr:int, RoomNr:int, ActuatorName:char*, Position:percentageType	
GetPosition		percentageType
SetOperationState	state:malfunctionType	void
IsOperatingNormal		int
Inherited Class::Operation	**Arguments**	**Return Type**

2.2.3.37.1 Attribute der Klasse RadiatorValve

Attribut	Erläuterung
Position	Vom System verwaltete Position des RadiatorValves. ACHTUNG: Dieser Wert kann vom tatsächlichen Wert in der realen Welt abweichen, da kein Sensor vorhanden ist, der diesen tatsächlichen Wert ermitteln könnte. Es werden also von einer Startposition beim Systemstart nur relative Positionsänderungen vorgenommen, und es wird davon ausgegangen, daß diese Änderungen auch exakt ausgeführt werden, was aber nicht immer möglich ist.
Number	Gibt die Anzahl der benötigten RadiatorValves an.
RoomNr	Gibt die Raumnummer an.
SectionNr	Gibt die Sektionsnummer an.
OperationState	Das Attribut OperationState gibt an, ob der Aktuator in Ordnung oder defekt ist.

2.2.3.37.2 Methodenschnittstellen der Klasse RadiatorValve

Signatur: **percentageType SetPosition(value:percentageType)**

Beschreibung: Setzt die Position des RadiatorValves und gibt den gesetzten Wert zurück.

ACHTUNG: Anmerkung siehe Attribut Position!

Signatur: **percentageType GetPosition()**

Beschreibung: Liefert die vom System verwaltete Position des RadiatorValves zurück.

ACHTUNG: Anmerkung siehe Attribut Position!

Signatur: **void SetOperationState(state:malfunctionType)**

Beschreibung: Mit der Methode SetOperationState kann man den aktuellen OperationState des Aktuators setzen.

Signatur: **int IsOperatingNormal()**

Beschreibung: Die Methode IsOperatingNormal liefert den aktuellen OperationState des Aktuators zurueck.

2.2.3.38 Classtable der Klasse RealBlind

RealBlind		
Attribute	**Type**	**Default Value**
Direction	directionType	
RunTime	durationType	0
CurrentRunTime	durationType	0
Operation	**Arguments**	**Return Type**
SetAngle	value:degreeType	degreeType
SetPosition	value:percentageType	percentageType
Up		int
Down		int
MoveUp		int

MoveDown		int
Stop		int
RealBlind	SectionNr:int, RoomNr:int, ActuatorName:char*	
RealBlind	SectionNr:int, RoomNr:int, ActuatorName:char*, Angle:degreeType, Position:percentageType	
SetStartTime		void
CalculatePosition		void
SetDirection	newPosition:percentageType	void
CalculateRunTime	position:percentageType	void
Inherited Class::Operation	**Arguments**	**Return Type**
ScheduledFct::callBack		
ScheduledFct::unregisterCallBack		
ScheduledFct::registerCallBack		
Portbaustein::CallBitOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::Portbaustein		
Actuator::IsOperatingNormal		int
Actuator::SendEmail		void
Actuator::SetOperationState	state:malfunctionType	void
Actuator::Actuator		
Actuator::CallBitOutputPort	value:int	void
Actuator::CallIntOutputPort	value:int	void

Timer::getActualTime		Time

2.2.3.38.1 Attribute der Klasse RealBlind

Attribut	Erläuterung
Direction	Wenn sich die Blinds bewegen, gibt dieses Attribut die Richtung der Bewegung an.
RunTime	Die beim Aufruf der Methode berechneten Zeit in Sekunden, die benötigt wird, um die Blinds auf eine bestimmte Postition zu fahren, wir in diesem Attribut gespeichert.
CurrentRunTime	Gibt in Sekunden an, wie lange die Blinds bereits in Bewegung sind.

2.2.3.38.2 Methodenschnittstellen der Klasse RealBlind

Signatur: **degreeType SetAngle(value:degreeType)**

Beschreibung: Setzt den Winkel einer einzelnen Sonnenblende auf den angegebenen Wert.

Signatur: **percentageType SetPosition(value:percentageType)**

Beschreibung: Setzt die Position einer einzelnen Sonnenblende auf den angegebenen Wert.

Signatur: **int Up()**

Beschreibung: Fährt eine einzelne Sonnenblende ganz nach oben (0%)

Signatur: **int Down()**

Beschreibung: Fährt eine einzelne Sonnenblende ganz nach unten (100%)

Signatur: **int MoveUp()**

Beschreibung: Nach Auruf dieser Methode wird ein einzelner RealBlind solange nach oben gefahren bis die Methode Stop() aufgerufen wird, oder der RealBlind ganz oben ist.

Signatur: **int MoveDown()**

Beschreibung: Nach Aufruf dieser Methode wird ein einzelner RealBlind solange nach unten gefahren bis die Methode Stop() aufgerufen wird oder der RealBlind ganz unten ist.

Signatur: **int Stop()**

Beschreibung: Wenn sich ein einzelner RealBlind bewegt, kann er mit Hilfe der Methode Stop() gestoppt werden.

2.2.3.39 Classtable der Klasse RealRadiatorValve

RealRadiatorValve		
Attribute	**Type**	**Default Value**
CurrentRun-Time	durationType	
RunTime	durationType	
Direction	directionType	
Position	percentageType	
Operation	**Arguments**	**Return Type**
SetPosition	value:percentageType	percentage-Type
RealRadiator-Valve	SectionNr:int, RoomNr:int, ActuatorName:char*, Position:per-centageType	
RealRadiator-Valve	SectionNr:int, RoomNr:int, ActuatorName:char*	
CalculateRun-Time	position:percentageType	void
SetDirection	newPosition:percentageType	void
CalculatePosi-tion		void
GetPosition		percentage-Type
Inherited Class::Operat ion	**Arguments**	**Return Type**
Schedu-ledFct::call-Back		
Schedu-ledFct::unregi-sterCallBack		

Schedu-ledFct::regi-sterCallBack		
Portbau-stein::CallBi-tOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbau-stein::CallIn-tInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbau-stein::CallIn-tOutputPort	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbau-stein::CallBi-tInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbau-stein::Portbau-stein		
Actua-tor::IsOpera-tingNormal		int
Actuator::Sen-dEmail		void
Actuator::Set-OperationState	state:malfunctionType	void
Actua-tor::Actuator		
Actua-tor::CallBi-tOutputPort	value:int	void
Actuator::Cal-lIntOutputPort	value:int	void
Timer::getAc-tualTime		Time

2.2.3.39.1 Attribute der Klasse RealRadiatorValve

Attribut	Erläuterung
CurrentRunTime	Gibt in Sekunden an, wie lange die Blinds bereits in Bewegung sind.

Attribut	Erläuterung
RunTime	Die beim Aufruf der Methode berechneten Zeit in Sekunden, die benötigt wird, um die Blinds auf eine bestimmte Postition zu fahren, wir in diesem Attribut gespeichert.
Direction	Wenn sich die Blinds bewegen, gibt dieses Attribut die Richtung der Bewegung an.
Position	Vom System verwaltete Position des RadiatorValves. ACHTUNG: Dieser Wert kann vom tatsächlichen Wert in der realen Welt abweichen, da kein Sensor vorhanden ist, der diesen tatsächlichen Wert ermitteln könnte. Es werden also von einer Startposition beim Systemstart nur relative Positionsänderungen vorgenommen, und es wird davon ausgegangen, daß diese Änderungen auch exakt ausgeführt werden, was aber nicht immer möglich ist.

2.2.3.39.2 Methodenschnittstellen der Klasse RealRadiatorValve

Signatur: **percentageType SetPosition(value:percentageType)**

Beschreibung: Setzt die Position eines einzelnen radiator valves auf den angegebenen Wert.

Signatur: **percentageType GetPosition()**

Beschreibung: Liefert die vom System verwaltete Position des RadiatorValves zurück.

ACHTUNG: Anmerkung siehe Attribut Position!

2.2.3.40 Classtable der Klasse Room

Room		
Attribute	**Type**	**Default Value**
Flaeche		
Operation	**Arguments**	**Return Type**
Room	InitialSectionNr:int, InitialRoomNr:int, Flaeche:double, ZeitPtr:Timer*, SectionPtr:Section*	
pushButtonActivated		void
EnterSensorDefaultValue	SensorName:char*, value:int	int

EnterMalfunction	Name:char*, state:mal-functionType	int
SetZeitspanneVerlassenIntervall	Intervall:intervallType	void
SetDT2Intervall	Intervall:intervallType	void
SetDT2StandbyIntervall	Intervall:intervallType	void
SetZeitspanneEintrittDefault	Zeitspanne:int	void
SetZeitspanneAufheizenStandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempIntervall	Intervall:intervallType	void
SetAbwesenheitsTempStandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempIntervall	Intervall:intervallType	void
SetZeitspanneEintrittIntervall	Intervall:intervallType	void
SetZeitspanneEintrittStandbyIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenStandby-Intervall	Intervall:intervallType	void
SetZeitspanneVerlassenStandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempDefault	Temperatur:int	void
SetZeitspanneEintrittStandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempDefault	Temperatur:int	void
SetAbwesenheitsTempStandbyDefault	Temperatur:int	void
SetDT2StandbyDefault	Zeitspanne:int	void
SetDT2Default	Zeitspanne:int	void
SetZeitspanneVerlassenDefault	Zeitspanne:int	void
SetZeitspanneAufheizenDefault	Zeitspanne:int	void
SetZeitspanneVerlassenStandbyDefault	Zeitspanne:int	void
SetHolidays	Intervall:intervallType	void
DeleteHolidays	Intervall:intervallType	void
DeleteWorkingDays	Intervall:intervallType	void
SetWorkingDays	Intervall:intervallType	void
DeleteComfortTime	Intervall:intervallType	void

DeleteStandbyTime	Intervall:intervallType	void
SetStandbyTime	Intervall:intervallType	void
SetComfortTime	Intervall:intervallType	void
SetDT2Standby	Zeitspanne:int	void
SetDT2	Zeitspanne:int	void
SetZeitspanneAufheizenStandbyNormal	Zeitspanne:int	void
SetZeitspanneVerlassenNormal	Zeitspanne:int	void
SetZeitspanneVerlassenStandbyNormal	Zeitspanne:int	void
SetAbwesenheitsTempNormal	Temperatur:int	void
SetZeitspanneEintrittNormal	Zeitspanne:int	void
SetZeitspanneEintrittStandbyNormal	Zeitspanne:int	void
SetAbwesenheitsTempStandbyNormal	Temperatur:int	void
SetAnwesenheitsTempNormal	Temperatur:int	void
SetZeitspanneAufheizenNormal	Zeitspanne:int	void
SetIntervall		
SetRoomLightLT3FM	Zeitspanne:int	bool
SetRoomLightLT1FM	Zeitspanne:int	bool
SetRoomLightDefaultAmbient-LightLevelFM	Level:intensityType	bool
SetRoomLightDefaultLightSceneFM	LS:lightSceneType	bool
SwitchRoomLightFM	State:zLichtStatusType	bool
GetSectionPtr		Section *
GetValueInterfacePtr		ValueInterface *
SetRoomLightAmbientLightLevel	Level:intensityType	bool
SetRoomLightScene	LS:lightSceneType	bool

2.2.3.40.1 Attribute der Klasse Room

Attribut	Erläuterung
Flaeche	Größe des Zimmers in qm.

2.2.3.40.2 Methodenschnittstellen der Klasse Room

Signatur: **void pushButtonActivated()**

Beschreibung: Wird vom DimmableLightController im jeweiligen Raum benachrichtigt, wenn ein PushButton gedrückt wurde. Leitet dieses Ereignis an den RoomLightRegulator weiter.

Signatur: **int EnterSensorDefaultValue(SensorName:char*, value:int)**

Beschreibung: Ruft die Methode SetDefaultValue() der jeweiligen Sensorklasse auf um den Default Value zu setzen.

Signatur: **int EnterMalfunction(Name:char*, state:malfunctionType)**

Beschreibung: Ruft die Methode SetOperationStatus() der entsprechenden Sensorklasse mit
true : zum setzen einer Malfunction und
false: zum Aufheben einer Malfunction
auf.

Signatur: **void SetZeitspanneVerlassenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Intervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2StandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneEintrittDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAnwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetAbwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAnwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetDT2StandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Default(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneAufheizenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetHolidays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Urlaubszeiten auf.

Signatur: **void DeleteHolidays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Urlaubszeiten auf.

Signatur: **void DeleteWorkingDays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Wochenintervalle auf.

Signatur: **void SetWorkingDays(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Wochenintervalle auf.

Signatur: **void DeleteComfortTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void DeleteStandbyTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void SetStandbyTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void SetComfortTime(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Tagesintervalle auf.

Signatur: **void SetDT2Standby(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneAufheizenStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetAbwesenheitsTempNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittStandbyNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAbwesenheitsTempStandbyNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAnwesenheitsTempNormal(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneAufheizenNormal(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur:

Beschreibung:

Signatur: **bool SetRoomLightLT3FM(Zeitspanne:int)**

Beschreibung:

Signatur: **bool SetRoomLightLT1FM(Zeitspanne:int)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultAmbientLightLevelFM(Level:intensityType)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultLightSceneFM(LS:lightSceneType)**

Beschreibung:

Signatur: **Section * GetSectionPtr()**

Beschreibung: Liefert einen Zeiger auf die eigene Section zurück.

Signatur: **ValueInterface * GetValueInterfacePtr()**

Beschreibung:

Signatur: **bool SetRoomLightAmbientLightLevel(Level:intensityType)**

Beschreibung:

Signatur: **bool SetRoomLightScene(LS:lightSceneType)**

Beschreibung:

Signatur: **bool SwitchRoomLightFM(State:zLichtStatusType)**

Beschreibung:

2.2.3.41 Classtable der Klasse RoomLightController

RoomLightController		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
getRoomLightIntensity		intensityType
getOutdoorSensorValue	sensor:sensorType	intensityType

RoomLightController	RoomPtr:Room*, RoomNr:int, TimerPtr:Timer*	

2.2.3.41.1 Attribute der Klasse RoomLightController

Keine.

2.2.3.41.2 Methodenschnittstellen der Klasse RoomLightController

Signatur: **intensityType getRoomLightIntensity()**

Beschreibung: Berechnet die Helligkeit im jeweiligen Raum unter Berücksichtigung der aktuellen Stellung der Blinds und der OutdoorLightSensoren.

Signatur: **intensityType getOutdoorSensorValue(sensor:sensorType)**

Beschreibung: Liefert den aktuellen Wert des gewünschten OutdoorLightSensors zurück.

2.2.3.42 Classtable der Klasse RoomLightControllerWithBlinds

RoomLightControllerWithBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
getIntensityLevelBlindsDown		intensityType
getIntensityLevelBlindsUp		intensityType
calculateOptimalBlindPos	brightness:intensityType	percentageType
calculateOptimalBlindAngle	brightness:intensityType	degreeType
RoomightControllerWithBlinds	RoomPtr:Room*, BlindControllerPtr:BlindController*, RoomNr:int, TimerPtr:Timer*	

2.2.3.42.1 Attribute der Klasse RoomLightControllerWithBlinds

Keine.

2.2.3.42.2 Methodenschnittstellen der Klasse RoomLightControllerWithBlinds

Signatur: **intensityType getIntensityLevelBlindsDown()**

Beschreibung: Berechnet die Helligkeit im jeweiligen Raum für den Fall, dass alle Blinds komplett heruntergefahren sind. Fragt dazu ausserdem die OutdoorLightSensoren ab.

Signatur: **intensityType getIntensityLevelBlindsUp()**

Beschreibung: Berechnet die Helligkeit im jeweiligen Raum für den Fall, dass alle Blinds komplett hochgefahren sind. Fragt dazu ausserdem die OutdoorLightSensoren ab.

Signatur: **percentageType calculateOptimalBlindPos(brightness:intensityType)**

Beschreibung: Berechnet ausgehend von den aktuellen Werten der OutdoorLightSensoren und der gewünschten Helligkeit im Raum die dafür nötige Position (0-100%) der Blinds.

Signatur: **degreeType calculateOptimalBlindAngle(brightness:intensityType)**

Beschreibung: Berechnet ausgehend von den aktuellen Werten der OutdoorLightSensoren und der gewünschten Helligkeit im Raum den dafür nötigen Winkel (0-180) der Blinds.

Signatur:

Beschreibung:

2.2.3.43 Classtable der Klasse RoomLightControllerWithoutBlinds

RoomLightControllerWithoutBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
RoomLightControllerWithout-Blinds	RoomPtr:Room*, RoomNr:int, TimerPtr:Timer*	

2.2.3.43.1 Attribute der Klasse RoomLightControllerWithoutBlinds

Keine.

2.2.3.43.2 Methodenschnittstellen der Klasse RoomLightControllerWithoutBlinds

Die Klasse RoomLightControllerWithoutBlinds besitzt keine Methodenschnittstellen.

2.2.3.44 Classtable der Klasse RoomLightRegulator

RoomLightRegulator		
Attribute	**Type**	**Default Value**
defaultLightScene	lightSceneType	14,14
userLightScene	lightSceneType	14,14
mode	operationModeType	manuell
presence	zAnwesenheitType	CNiemandDa
Operation	**Arguments**	**Return Type**
activateDefaultLight-Scene		void
pushButtonActivated		void
fMSetDefaultAmbi-entLightLevel	intensity:intensityType	void
fMSetDefaultLight-Scene	lightScene:lightSceneType	void
userSetAmbient-LightLevel	intensity:intensityType	void
userSetLightScene	lightScene:lightSceneType	void
updateLight		void
updateUserPresence	newPresence:zAnwesenheitType	void
fMSwitchLight	state:zLichtStatusType	void
RoomLightRegulator	DimmableLightControllerPtr1:DimmableLight-Controller*, DimmableLightControllerPtr2:DimmableLight-Controller*, RoomLightControllerPtr:RoomLight-Controller*	

2.2.3.44.1 Attribute der Klasse RoomLightRegulator

Attribut	Erläuterung
defaultLightScene	Die LightScene (vom Hausmeister gewählt), die standardmässig eingestellt wird, wenn der Benutzer keine eigene gewählt hat oder laenger als LT1 niemand im jeweiligen Raum war.
userLightScene	Die LightScene, die der Benutzer für die Dauer seiner Anwesenheit im jeweiligen Raum selbst wählen kann.
mode	Zustandsvariable beschreibt den operationsmodus des Systems. Mögliche Zustände: manual - Das Licht wurde vom Hausmeister/ Benutzer manuell ein-/ausgeschaltet. Es findet keine Lichtregelung statt. automatic - Normaler Betrieb mit Lichtregelung.
presence	Speichert, ob Personen im jeweiligen Raum anwesend sind oder nicht.

2.2.3.44.2 Methodenschnittstellen der Klasse RoomLightRegulator

Signatur: **void activateDefaultLightScene()**

Beschreibung: Setzt die (evtl. temporär geänderte) UserLightScene wieder auf die DefaultLightScene zurueck. Wird durch PRLightController nach Ablauf von LT1 bei Betreten des jeweiligen Raumes aufgerufen.

Signatur: **void pushButtonActivated()**

Beschreibung: Benachrichtigt den RoomLightRegulator, dass die jeweilige Lampe vom Benutzer per PushButton ein- bzw. ausgeschaltet wurde.

Signatur: **void fMSetDefaultAmbientLightLevel(intensity:intensityType)**

Beschreibung: Setzt die Standard-Helligkeit im einem Raum. Darf nur vom Hausmeister benutzt werden.

Signatur: **void fMSetDefaultLightScene(lightScene:lightSceneType)**

Beschreibung: Setzt die DefaultLightScene im einem Raum. Darf nur vom Hausmeister benutzt werden.

Signatur: **void userSetAmbientLightLevel(intensity:intensityType)**

Beschreibung: Setzt die Helligkeit der Hintergrundbeleuchtung in einem Raum mfür die Dauer der Anwesenheit des Benutzers. Darf nur vom normalen Benutzer benutzt werden.

Signatur: **void userSetLightScene(lightScene:lightSceneType)**

Beschreibung: Setzt die UserLightScene im einem Raum. Darf nur vom normalen Benutzer benutzt werden.

Signatur: **void updateLight()**

Beschreibung: Initiiert die Neuregulierung der Zimmerbeleuchtung/Blinds durch den RoomLightRegulator. Wird periodisch vom Scheduler aufgerufen.

Signatur: **void updateUserPresence(newPresence:zAnwesenheitType)**

Beschreibung: Informiert den RoomLightRegulator wenn sich der Anwesenheitsstatus von Personen im jeweiligen Zimmer geändert hat.

Signatur: **void fMSwitchLight(state:zLichtStatusType)**

Beschreibung: Schaltet alle Lampen im jeweiligen Raum auf volle/null Helligkeit (je nach Parameter). Kann nur vom Hausmeister benutzt werden und nur dann, wenn sich niemand im Raum befindet.

2.2.3.45 Classtable der Klasse RoomLightRegulatorWithBlinds

RoomLightRegulatorWithBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**

2.2.3.45.1 Attribute der Klasse RoomLightRegulatorWithBlinds

Keine.

2.2.3.45.2 Methodenschnittstellen der Klasse RoomLightRegulatorWithBlinds

2.2.3.46 Classtable der Klasse RoomLightRegulatorWithoutBlinds

RoomLightRegulatorWithoutBlinds		
Attribute	**Type**	**Default Value**

Operation	**Arguments**	**Return Type**

2.2.3.46.1 Attribute der Klasse RoomLightRegulatorWithoutBlinds

Keine.

2.2.3.46.2 Methodenschnittstellen der Klasse RoomLightRegulatorWithoutBlinds

2.2.3.47 Classtable der Klasse RoomTemperatureController

RoomTemperatureController		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
abfrageMinimalZimmertemperatur		tempType
RoomTemperatureController	InitialSectionNr:int, InitialRoomNr:int	
abfrageMittelwertZimmertemperatur		tempType

2.2.3.47.1 Attribute der Klasse RoomTemperatureController

Keine.

2.2.3.47.2 Methodenschnittstellen der Klasse RoomTemperatureController

Signatur: **tempType abfrageMittelwertZimmertemperatur()**

Beschreibung: Fragt die 3 Zimmertemperatursensoren für den Temperaturereguator ab und erhält direkt den Mittelwert.

Signatur: **tempType abfrageMinimalZimmertemperatur()**

Beschreibung: Fragt die drei Instanzen von RoomTemperatureSensor für den RadiatorController ab, und liefert den kleinsten Wert zurück.

2.2.3.48 Classtable der Klasse RoomTemperatureSensor

RoomTemperatureSensor		
Attribute	**Type**	**Default Value**
UpperTempLimit	tempType	+70
LowerTempLimit	tempType	-40
Operation	**Arguments**	**Return Type**
GetTemperature		tempType
SetDefaultValue	value:int	int
RoomTemperatureSensor	SectionNr:int, RoomNr:int, SensorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Sensor::GetDefaultValue		int
Sensor::IsOperatingNormal		int
Sensor::SendEmail		int
Sensor::SetDefaultValue	value:int	int
Sensor::SetOperationState	state:int	

Sensor::Sensor		

2.2.3.48.1 Attribute der Klasse RoomTemperatureSensor

Attribut	Erläuterung
UpperTempLimit	Das Attribut UpperTempLimit speichert die obere Grenze des Wertebereichs des realen Sensors.
LowerTempLimit	Das Attribut LowerTempLimit speichert die untere Grenze des Wertebereiches des realen Sensors.

2.2.3.48.2 Methodenschnittstellen der Klasse RoomTemperatureSensor

Signatur: **tempType GetTemperature()**

Beschreibung: Liefert die vom Temperaturfühler aktuell gemessene Temperatur zurück und setzt diesen Wert im Attribut Temperature.
Wertebereich: -40 - 70 Grad Celsius

Signatur: **int SetDefaultValue(value:int)**

Beschreibung: Implementiert die abstrakte Methode der Klasse Sensor. Wird verwendet, um den DefaultValue des Sensors zu setzen. Zulaessiger Bereich: -40 - 70 Grad Celsius.

2.2.3.49 Classtable der Klasse RoomWithBlinds

RoomWithBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
getValueInterfacePtr		ValueInterface *
RoomWithBlinds	sectionNr:int, roomNr:int	

2.2.3.49.1 Attribute der Klasse RoomWithBlinds

Keine.

2.2.3.49.2 Methodenschnittstellen der Klasse RoomWithBlinds

Signatur: **ValueInterface * getValueInterfacePtr()**

Beschreibung:

2.2.3.50 Classtable der Klasse RoomWithoutBlinds

RoomWithoutBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
getValueInterfacePtr		ValueInterface *
RoomWithoutBlinds	sectionNr:int, roomNr:int	

2.2.3.50.1 Attribute der Klasse RoomWithoutBlinds

Keine.

2.2.3.50.2 Methodenschnittstellen der Klasse RoomWithoutBlinds

Signatur: **ValueInterface * getValueInterfacePtr()**

Beschreibung:

2.2.3.51 Classtable der Klasse Section

Section		
Attribute	**Type**	**Default Value**
Zimmeranzahl		
SectionNr	int	
Operation	**Arguments**	**Return Type**
GetSectionNr		int
BenoetigteKesselTempAbfragen		

RegenAbschnitt		
Section	InitialSectionNr:int, ZimmerFlaechenListe:SList<double>, ZeitPtr:Timer, HousePtr:House*	
EnterFMLichtAnsteuern	LichtStatus:zLichtStatusType	void
EnterSensorDefaultValue	SensorName:char*, RoomNr:int, Value:int	int
EnterMalfunction	Name:char*, RoomNr:int, State:malfunctionType	int
SetDT2Intervall	Intervall:intervallType	void
SetDT2StandbyIntervall	Intervall:intervallType	void
SetAbwesenheitsTempIntervall	Intervall:intervallType	void
SetZeitspanneVerlassen	ZeitspanneVerlassenParam:durationType	void
SetZeitspanneEintrittIntervall	Intervall:intervallType	void
SetZeitspanneEintrittStandbyIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenIntervall	Intervall:intervallType	void
SetZeitspanneAufheizenStandbyIntervall	Intervall:intervallType	void
SetZeitspanneVerlassenStandbyIntervall	Intervall:intervallType	void
SetZeitspanneVerlassenIntervall	Intervall:intervallType	void
SetAbwesenheitsTempDefault	Temperatur:int	void
SetZeitspanneEintrittStandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempDefault	Temperatur:int	void
SetAbwesenheitsTempStandbyDefault	Temperatur:int	void
SetDT2StandbyDefault	Zeitspanne:int	void
SetDT2Default	Zeitspanne:int	void
SetZeitspanneVerlassenDefault	Zeitspanne:int	void
SetZeitspanneAufheizenDefault	Zeitspanne:int	void
SetZeitspanneVerlassenStandbyDefault	Zeitspanne:int	void

SetZeitspanneEintrittDefault	Zeitspanne:int	void
SetZeitspanneAufheizenStandbyDefault	Zeitspanne:int	void
SetAnwesenheitsTempIntervall	Intervall:intervallType	void
SetAbwesenheitsTempStandbyIntervall	Intervall:intervallType	void
SetRoomLightLT3FM	Zeitspanne:int, RoomNr:int	bool
SetRoomLightLT1FM	Zeitspanne:int, RoomNr:int	bool
SetRoomLightDefaultAmbient-LightLevelFM	Level:intensityType, RoomNr:int	bool
SetRoomLightDefaultLightSceneFM	LS:lightSceneType, RoomNr:int	bool
SwitchRoomLightFM	SwitchOn:bool, RoomNr:int	bool
GetHousePtr		House *

2.2.3.51.1 Attribute der Klasse Section

Attribut	Erläuterung
Zimmeranzahl	Anzahl der Zimmer
SectionNr	Durch das Attribut SectionNr wird die SectionNr des Section-Objekts gespeichert. Da jede Etage des Gebäudes in drei Abschnitte eingeteilt ist, enthält SectionNr einen Integer-Wert 1, 2, oder 3.

2.2.3.51.2 Methodenschnittstellen der Klasse Section

Signatur:

Beschreibung: Fragt alle Zimmer nach der benötigten Kesseltemperatur und übermittelt das Maximum als Ergebnis an das Haus.

Signatur:

Beschreibung: Meldet neu einsetzenden Regen an alle Zimmer.

Signatur: **int EnterSensorDefaultValue(SensorName:char*, RoomNr:int, Value:int)**

Beschreibung: Ruft die Methode SetDefaultValue() der jeweiligen Sensorklasse auf um den Default Value zu setzen.

Signatur: **int EnterMalfunction(Name:char*, RoomNr:int, State:malfunctionType)**

Beschreibung: Ruft die Methode SetOperationStatus() der entsprechenden Sensorklasse mit
true : zum setzen einer Malfunction und
false: zum Aufheben einer Malfunction
auf.

Signatur: **void SetDT2Intervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2StandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetAbwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneEintrittStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetZeitspanneVerlassenIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetAbwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetZeitspanneEintrittStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAnwesenheitsTempDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyDefault(Temperatur:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetDT2StandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetDT2Default(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneVerlassenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Zeitspannen auf.

Signatur: **void SetZeitspanneAufheizenDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneVerlassenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode im PRTempController auf.

Signatur: **void SetZeitspanneEintrittDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetZeitspanneAufheizenStandbyDefault(Zeitspanne:int)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Spezialzeiten auf.

Signatur: **void SetAnwesenheitsTempIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **void SetAbwesenheitsTempStandbyIntervall(Intervall:intervallType)**

Beschreibung: Ruft die entsprechende Methode in einer Instanz von Temperaturdaten auf.

Signatur: **bool SetRoomLightLT3FM(Zeitspanne:int, RoomNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightLT1FM(Zeitspanne:int, RoomNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultAmbientLightLevelFM(Level:intensityType, RoomNr:int)**

Beschreibung:

Signatur: **bool SetRoomLightDefaultLightSceneFM(LS:lightSceneType, RoomNr:int)**

Beschreibung:

Signatur: **bool SwitchRoomLightFM(SwitchOn:bool, RoomNr:int)**

Beschreibung:

Signatur: **House * GetHousePtr()**

Beschreibung:

Signatur: **void SetZeitspanneVerlassen(ZeitspanneVerlassenParam:durationType)**

Beschreibung: Die Methode SetZeitspanneVerlassen() ruft die Methode HallwaySection.SetZeitspanneVerlassen() auf und übergibt ihr einen Wert ZeitspanneVerlassen vom Typ durationType.
Diese Methode wird zum Durchreichen des Wertes ZeitspanneVerlassen vom ControlPanelFM-Objekt zum PRHallwayLightController-Objekt des entsprechenden Flurabschnitts benötigt.

Signatur: **void EnterFMLichtAnsteuern(LichtStatus:zLichtStatusType)**

Beschreibung: Die Methode EnterFMLichtAnsteuern() ruft die Methode HallwaySection.EnterFMLichtAnsteuern() auf und übergibt ihr einen Wert LichtStatus vom Typ zLichtStatusType.
Diese Methode wird zum Durchreichen des Wertes LichtStatus vom ControlPanelFM-Objekt zum LightControlUnit-Objekt des entsprechenden Flurabschnitts benötigt.

Signatur: **int GetSectionNr()**

Beschreibung: Die Methode GetSectionNr() gibt die SectionNr des Section-Objekts zurück.

2.2.3.52 Classtable der Klasse Sensor

Sensor		
Attribute	**Type**	**Default Value**
DefaultValue	int	
OperationState	malfunctionType	
SensorName	char*	
SectionNr	int	
RoomNr	int	
Operation	**Arguments**	**Return Type**
GetDefaultValue		int
IsOperatingNormal		int
SendEmail		int
SetDefaultValue	value:int	int
SetOperationState	state:malfunctionType	void
Sensor		
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void

2.2.3.52.1 Attribute der Klasse Sensor

Attribut	Erläuterung
DefaultValue	Das Attribut DefaultValue speichert den DefaultValue des Sensors, der bei einer Fehlfunktion übermittelt wird.
OperationState	Das Attribut OperationState gibt an, ob der Sensor in Ordnung oder defekt ist.
SensorName	Das Attribut Name speichert den Namen des Sensors. Z.B. : wts1 für WaterTemperatureSensor1
SectionNr	Das Attribut SectionNr speichert die Nummer der Section, in der sich der reale Sensor befindet.
RoomNr	Das Attribut RoomNr speichert die Nummer des Raumes, in dem sich der reale Sensor befindet.

2.2.3.52.2 Methodenschnittstellen der Klasse Sensor

Signatur: **int IsOperatingNormal()**

Beschreibung: Die Methode IsOperatingNormal liefert den aktuellen OperationState des Sensors zurueck.

Signatur: **int SetDefaultValue(value:int)**

Beschreibung: Abstrakte Methode zum setzen des DefaultValues des Sensors. Wird in den abgeleiteten Klassen implementiert.

Signatur: **void SetOperationState(state:malfunctionType)**

Beschreibung: Mit der Methode SetOperationState kann man den aktuellen OperationState des Sensors setzen.

2.2.3.53 Classtable der Klasse StatusLine

StatusLine		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
IsOn		int

StatusLine	SectionNr:int, RoomNr:int, SensorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Sensor::GetDefaultValue		int
Sensor::IsOperatingNormal		int
Sensor::SendEmail		int
Sensor::SetDefaultValue	value:int	int
Sensor::SetOperationState	state:int	
Sensor::Sensor		

2.2.3.53.1 Attribute der Klasse StatusLine

Keine.

2.2.3.53.2 Methodenschnittstellen der Klasse StatusLine

Signatur: **int IsOn()**

Beschreibung: Gibt Boolean Werte zurueck:
TRUE : Wenn das Licht tatsaechlich an ist (Strom bekommt)
FALSE: Wenn das Licht tatsaechlich aus ist (keinen Strom bekommt)

Der tatsaechliche Status des Lichtes kann nur ueber diese Methode ermittelt werden, da z.B. ein Benutzer des Systems das Licht ueber einen Schalter ausschalten kann, ohne dass das System davon Nachricht bekommt, es koennte also intern immer noch davon ausgehen, dass das Licht an ist...

2.2.3.54 Classtable der Klasse Tagesintervalle

Tagesintervalle		
Attribute	**Type**	**Default Value**
Minutenstatus	modeType[1440]	off
Operation	**Arguments**	**Return Type**
DeleteStandbyTime	minTime:Clock, max-Time:Clock	void
SetComfortTime	minTime:Clock, max-Time:Clock	void
SetStandbyTime	minTime:Clock, max-Time:Clock	void
DeleteComforttime	minTime:Clock, max-Time:Clock	void
GetMinMode	Time:int	modeType
Tagesintervalle		

2.2.3.54.1 Attribute der Klasse Tagesintervalle

Attribut	Erläuterung
Minutenstatus	Hier ist für jede Minute eines Tages der jeweils eingegebene Modus (off, standby oder comfort) gespeichert.

2.2.3.54.2 Methodenschnittstellen der Klasse Tagesintervalle

Signatur: **void DeleteStandbyTime(minTime:Clock, maxTime:Clock)**

Beschreibung: Hiermit löscht der Benutzer eine Standbyzeit aus Minutenstatus.

Signatur: **void SetComfortTime(minTime:Clock, maxTime:Clock)**

Beschreibung: Hiermit setzt der Benutzer neue Comfortzeit in Minutenstatus.

Signatur: **void SetStandbyTime(minTime:Clock, maxTime:Clock)**

Beschreibung: Hiermit setzt der Benutzer neue Standbyzeit in Minutenstatus.

Signatur: **void DeleteComforttime(minTime:Clock, maxTime:Clock)**

Beschreibung: Hiermit löscht der Benutzer alte Comfortzeit aus Minutenstatus.

Signatur: **modeType GetMinMode(Time:int)**

Beschreibung: Liefert den Modus der aktuellen Minute zurück.

2.2.3.55 Classtable der Klasse TempOpModusErmittlung

TempOpModusErmittlung		
Attribute	**Type**	**Default Value**
AktuellerModus	modeType	off
Operation	**Arguments**	**Return Type**
UpdateWerte		void
TempOpModusErmittlung	WochenintervallePtr:Wochenintervalle*, UrlaubszeitenPtr:Urlaubszeiten*, HeizperiodenPtr:Heizperioden*, TagesintervallePtr:Tagesintervalle*	
ErmittleNeuenModus		void
callBack		void
SetPointers	AnwesenheitsTempNormalPtr:Temperaturdaten*, AbwesenheitsTempNormalPtr:Temperaturdaten*, AbwesenheitsTempStandbyNormalPtr:Temperaturdaten*, ZeitspanneVerlassenNormalPtr:ZeitspannenVerlassen*, ZeitspanneVerlassenStandbyNormalPtr:ZeitspannenVerlassen*, DT2StandbyPtr:ClassDT2Standby*, DT2Ptr:ClassDT2*, ZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal*, ZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*, ZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*, ZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*	void

2.2.3.55.1 Attribute der Klasse TempOpModusErmittlung

Attribut	Erläuterung
AktuellerModus	Gibt den aktuellen Modus (off, standby, comfort oder non-heating) des Systems an.

2.2.3.55.2 Methodenschnittstellen der Klasse TempOpModusErmittlung

Signatur: **void UpdateWerte()**

Beschreibung: Übermittelt die für den aktuellen Modus gültigen Heizregelwerte und den aktuellen Modus an das Teilsystem Temperaturregelung.

Signatur: **void ErmittleNeuenModus()**

Beschreibung: Berechnet jede Minute den aktuellen Modus des Systems und ruft bei Moduswechsel UpdateWerte auf.

Signatur: **void callBack()**

Beschreibung:

Signatur: **void SetPointers(AnwesenheitsTempNormalPtr:Temperaturdaten*, AbwesenheitsTempNormalPtr:Temperaturdaten*, AbwesenheitsTempStandbyNormalPtr:Temperaturdaten*, ZeitspanneVerlassenNormalPtr:ZeitspannenVerlassen*, ZeitspanneVerlassenStandbyNormalPtr:ZeitspannenVerlassen*, DT2StandbyPtr:ClassDT2Standby*, DT2Ptr:ClassDT2*, ZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittStandbyNormal*, ZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*, ZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*, ZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*)**

Beschreibung: Setzt alle verbleibenden Pointer auf assoziierte oder aggregierte Objekte, die im Konstruktor auf Grund der Initialisierungsreihenfolge nicht übergeben werden konnten.

2.2.3.56 Classtable der Klasse Temperaturdaten

Temperaturdaten		
Attribute	**Type**	**Default Value**

Operation	Arguments	Return Type
Check	Eingabe:tempType	void
CheckIntervall	Eingabeintervall:tempIntervall-Type	void
Temperaturdaten	TempOpModusErmitt-lungPtr:TempOpModusErmitt-lung*, InitialDefaultwert:tempType, InitialWert:tempType, Initial-Akzeptanzintervall:tempInter-vallType	

2.2.3.56.1 Attribute der Klasse Temperaturdaten

Keine.

2.2.3.56.2 Methodenschnittstellen der Klasse Temperaturdaten

Die Klasse Temperaturdaten besitzt keine Methodenschnittstellen.

2.2.3.57 Classtable der Klasse Temperatureregulator

Temperatureregulator		
Attribute	**Type**	**Default Value**
anwesenheit	zAnwesenheitType	
anwesenheitTemp	tempType	
abwesenheitTemp	tempType	
operationsModus	modeType	
minTemp	tempType	
zielTemp	tempType	
maxTemp	tempType	
zimmerTemp	tempType	
Operation	**Arguments**	**Return Type**
anwesenheitSetzen	anwesenheit:zAnwesenheitType	void

setzeTemperaturenUndModus	operationsModus:modeType, abwesenheitTemp:tempType, anwesenheitTemp:tempType	void
ventilNachregeln		void
setzeZieltemperatur		void
Temperatureregulator	InitialSectionNr:int, InitialRoomNr:int, WindowSectionAnzahl:int	
Inherited Class::Operation	**Arguments**	**Return Type**

2.2.3.57.1 Attribute der Klasse Temperatureregulator

Attribut	Erläuterung
operationsModus	Im Operationsmodus comfort muss immer die Temperatur AnwesenheitsTemp gehalten werden. Im Modus non-heating wird nicht geheizt, ausser wenn die Rohre einzufrieren drohen. In den Modi Standby und Off haengt die Zieltemperatur vom Wert des Booleans Anwesenheit ab.
abwesenheitTemp	Im Off- und Standby-Modus ist dies die Mindesttemperatur des Raumes, falls Anwesenheit falsch ist.
anwesenheitTemp	Im Off- oder Standby-Modus soll diese Temperatur erreicht werden, wenn Anwesenheit wahr ist.
anwesenheit	Im Off- und Standby-Modus wird Anwesenheit auf wahr gesetzt nachdem jemand das Zimmer betreten hat und die ZeitspanneEintritt vergangen ist.
zielTemp	In diesem Attribut wird die gewünschte Zieltemperatur gespeichert.
maxTemp	Anwesenheittemp + 1.
minTemp	Zieltemperatur - 1.
zimmerTemp	Aktuelle Zimmertemperatur, d.h. Durchschnittswert der drei an den Raumtemperatursensoren gemessenen Werte.

2.2.3.57.2 Methodenschnittstellen der Klasse Temperatureregulator

Signatur: **void ventilNachregeln()**

Beschreibung: Diese Methode benutzt ventilRegelung() des RadiatorController um eine Regelung der Zimmertemperatur amit den entsprechenden Parametern auf

Signatur: **void setzeTemperaturenUndModus(operationsModus:modeType, abwesenheit-Temp:tempType, anwesenheitTemp:tempType)**

Beschreibung: Methode zum Setzen der Attribute anwesenheitTemp, abwesenheitTemp, operations-Modus, maxTemp, minTemp und zielTemp.

Signatur: **void anwesenheitSetzen(anwesenheit:zAnwesenheitType)**

Beschreibung: Setzt das Attribut ZeitspanneAufheizenNormal auf den Wert des Attributs Zeitspanne-AufheizenDefault.

2.2.3.58 Classtable der Klasse TemperatureregulatorWithBlinds

TemperatureregulatorWithBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
istBlind-einstel-lungRelevant		bool
Tempe-raturere-gulator With-Blinds	InitialSectioNr:int, InitialRoomNr:int, BlindControllerPtr:BlindController*, RoomLightControllerWithBlindsPtr:RoomLightControllerWithBlinds*, WindowSectionAnzahl:int	

2.2.3.58.1 Attribute der Klasse TemperatureregulatorWithBlinds

Keine.

2.2.3.58.2 Methodenschnittstellen der Klasse TemperatureregulatorWithBlinds

Signatur: **bool istBlindeinstellungRelevant()**

Beschreibung: Die Funktion stellt über den RoomLightController fest, ob eine Änderung der Blind-einstellung zum Heizen/Kühlen genutzt werden kann.

2.2.3.59 Classtable der Klasse TemperatureregulatorWithoutBlinds

TemperatureregulatorWithoutBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
TemperatureregulatorWithout-Blinds	InitialSectionNr:int, InitialRoomNr:int, WindowSectionAnzahl:int	

2.2.3.59.1 Attribute der Klasse TemperatureregulatorWithoutBlinds

Keine.

2.2.3.59.2 Methodenschnittstellen der Klasse TemperatureregulatorWithoutBlinds

Die Klasse TemperatureregulatorWithoutBlinds besitzt keine Methodenschnittstellen.

2.2.3.60 Classtable der Klasse Time

Time		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
operator=	t:Time&	Time&
setTime	h:timeCompType, mi:timeCompType, d:timeCompType, mo:timeCompType, y:timeCompType	void
getTime	h:timeCompType&, mi:timeCompType&, d:timeCompType&, mo:timeCompType&.y:timeCompType&	void
operator++		Time

operator!=	t:Time&	bool
operator==	t:Time&	bool
operator>=	t:Time&	bool
operator>	t:Time&	bool
operator<=	t:Time&	bool
operator<	t:Time&	bool

2.2.3.60.1 Attribute der Klasse Time

Keine.

2.2.3.61 Classtable der Klasse Timer

Timer		
Attribute	**Type**	**Default Value**
speedup	int	1
Operation	**Arguments**	**Return Type**
getActualTime		Time

2.2.3.61.1 Attribute der Klasse Timer

Attribut	Erläuterung
speedup	Beschleunigung der Systemzeit im Vergleich zur realen Zeit.

2.2.3.61.2 Methodenschnittstellen der Klasse Timer

Die Klasse Timer besitzt keine Methodenschnittstellen.

2.2.3.62 Classtable der Klasse Urlaubszeiten

Urlaubszeiten		
Attribute	**Type**	**Default Value**

Operation	**Arguments**	**Return Type**
IsHoliday	Time:int	bool
Urlaubszeiten		

2.2.3.62.1 Attribute der Klasse Urlaubszeiten

Keine.

2.2.3.62.2 Methodenschnittstellen der Klasse Urlaubszeiten

Signatur: **bool IsHoliday(Time:int)**

Beschreibung: Überprüft ob der aktuelle Tag ein Urlaubstag ist oder ein Nicht-Urlaubstag ist.

2.2.3.63 Classtable der Klasse ValueInterface

ValueInterface		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
SetValues	OpModus:modeType, AbwesenheitTemp:tempType, AnwesenheitTemp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType	void
ValueInterface	temperatureregulatorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*	
Inherited Class::Operation	**Arguments**	**Return Type**

Signals Received	**Arguments**	
Signals Sent	**Arguments**	

2.2.3.63.1 Attribute der Klasse ValueInterface

Keine.

2.2.3.63.2 Methodenschnittstellen der Klasse ValueInterface

Signatur: **void SetValues(OpModus:modeType, AbwesenheitTemp:tempType, Anwesenheit-Temp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType)**

Beschreibung: Setzt die Werte An- und AbwesendTemp, Operationsmodus, ZeitspanneEintritt, DT2 und ZeitspanneVerlassen. Die Werte werden von der Systemmodusermittlung geliefert.

2.2.3.64 Classtable der Klasse ValueInterfaceWithBlinds

ValueInterfaceWithBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
SetValues	OpModus:modeType, AbwesenheitTemp:tempType, Anwesenheit-Temp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType	void
ValueInterfaceWithBlinds	temperatureregulatorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*, pRBlindControllerPtr:PRBlindController*	

2.2.3.64.1 Attribute der Klasse ValueInterfaceWithBlinds

Keine.

2.2.3.64.2 Methodenschnittstellen der Klasse ValueInterfaceWithBlinds

Signatur: **void SetValues(OpModus:modeType, AbwesenheitTemp:tempType, AnwesenheitTemp:tempType, ZeitspanneVerlassen:durationType, ZeitspanneEintritt:durationType, DT2:durationType, ZeitspanneAufheizen:durationType)**

Beschreibung: Ruft die gleiche Methode der Oberklasse auf, und setzt zusätzlich ZeitspanneVerlassen beim PRBlindController.

2.2.3.65 Classtable der Klasse ValueInterfaceWithoutBlinds

ValueInterfaceWithoutBlinds		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
ValueInterfaceWithoutBlinds	temperatureregualtorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*	

2.2.3.65.1 Attribute der Klasse ValueInterfaceWithoutBlinds

Keine.

2.2.3.65.2 Methodenschnittstellen der Klasse ValueInterfaceWithoutBlinds

Die Klasse ValueInterfaceWithoutBlinds besitzt keine Methodenschnittstellen.

2.2.3.66 Classtable der Klasse WaterTemperatureSensor

WaterTemperatureSensor		
Attribute	**Type**	**Default Value**
UpperTempLimit	tempType	100

LowerTempLimit	tempType	0
Operation	**Arguments**	**Return Type**
GetTemperature		tempType
SetDefaultValue	value:int	int
WaterTemperatureSensor	SectionNr:int, RoomNr:int, SensorName:char*	
Inherited Class::Operation	**Arguments**	**Return Type**
Portbaustein::CallBitInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallBitOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Portbaustein::CallIntInputPort	SectionNr:int, RoomNr:int, SensorName:char*	int
Portbaustein::CallIntOutput-Port	SectionNr:int, RoomNr:int, ActuatorName:char*, value:int	void
Sensor::GetDefaultValue		int
Sensor::IsOperatingNormal		int
Sensor::SendEmail		int
Sensor::SetDefaultValue	value:int	int
Sensor::SetOperationState	state:int	
Sensor::Sensor		

2.2.3.66.1 Attribute der Klasse WaterTemperatureSensor

Attribut	Erläuterung
UpperTempLimit	Das Attribut UpperTempLimit speichert die obere Grenze des Wertebereichs des realen Sensors.
LowerTempLimit	Das Attribut LowerTempLimit speichert die untere Grenze des Wertebereiches des realen Sensors.

2.2.3.66.2 Methodenschnittstellen der Klasse WaterTemperatureSensor

Signatur: **tempType GetTemperature()**

Beschreibung: Liefert die aktuelle Temperatur in Grad Celsius zurueck und setzt diesen Wert im Attribut Temperature. Wertebereich: 0-100 Grad Celsius.

Signatur: **int SetDefaultValue(value:int)**

Beschreibung: Implementiert die abstrakte Methode der Klasse Sensor. Setzt den Defaultvalue des Sensors. Zulaessiger Bereich: 0-100 Grad Celsius.

2.2.3.67 Classtable der Klasse Wochenintervalle

Wochenintervalle		
Attribute	**Type**	**Default Value**
Wochentagstatus	weekdayType[7]	nonworking
Operation	**Arguments**	**Return Type**
SetWorkingDay	Tagesintervall:weekIntervall-Type	void
DeleteWorkingDay	Tagesintervall:weekIntervall-Type	void
IsWorkingDay	Time:int	bool
Wochenintervalle		

2.2.3.67.1 Attribute der Klasse Wochenintervalle

Attribut	Erläuterung
Wochentagstatus	Hier ist für jeden Tag der Woche gespeichert, ob es ein working-day oder ein non-working-day ist.

2.2.3.67.2 Methodenschnittstellen der Klasse Wochenintervalle

Signatur: **void SetWorkingDay(Tagesintervall:weekIntervallType)**

Beschreibung: Hiermit setzt der Benutzer einen working-day in Wochentagstatus.

Signatur: **void DeleteWorkingDay(Tagesintervall:weekIntervallType)**

Beschreibung: Hiermit löscht der Benutzer einen working-day aus Wochentagstatus.

Signatur: **bool IsWorkingDay(Time:int)**

Beschreibung: Überprüft ob der aktuelle Tag ein working-day oder ein non-working-day ist.

2.2.3.68 Classtable der Klasse ZeitspannenVerlassen

ZeitspannenVerlassen		
Attribute	**Type**	**Default Value**
Operation	**Arguments**	**Return Type**
Check	Eingabe:durationType	void
CheckIntervall	Eingabeintervall:durationInter-vallType	void
ZeitspannenVerlassen	TempOpModusErmitt-lungPtr:TempOpModusErmitt-lung*, InitialDefaultwert:durati-onType, InitialWert:durati-onType, InitialAkzeptanzintervall:dura-tionIntervallType	

2.2.3.68.1 Attribute der Klasse ZeitspannenVerlassen

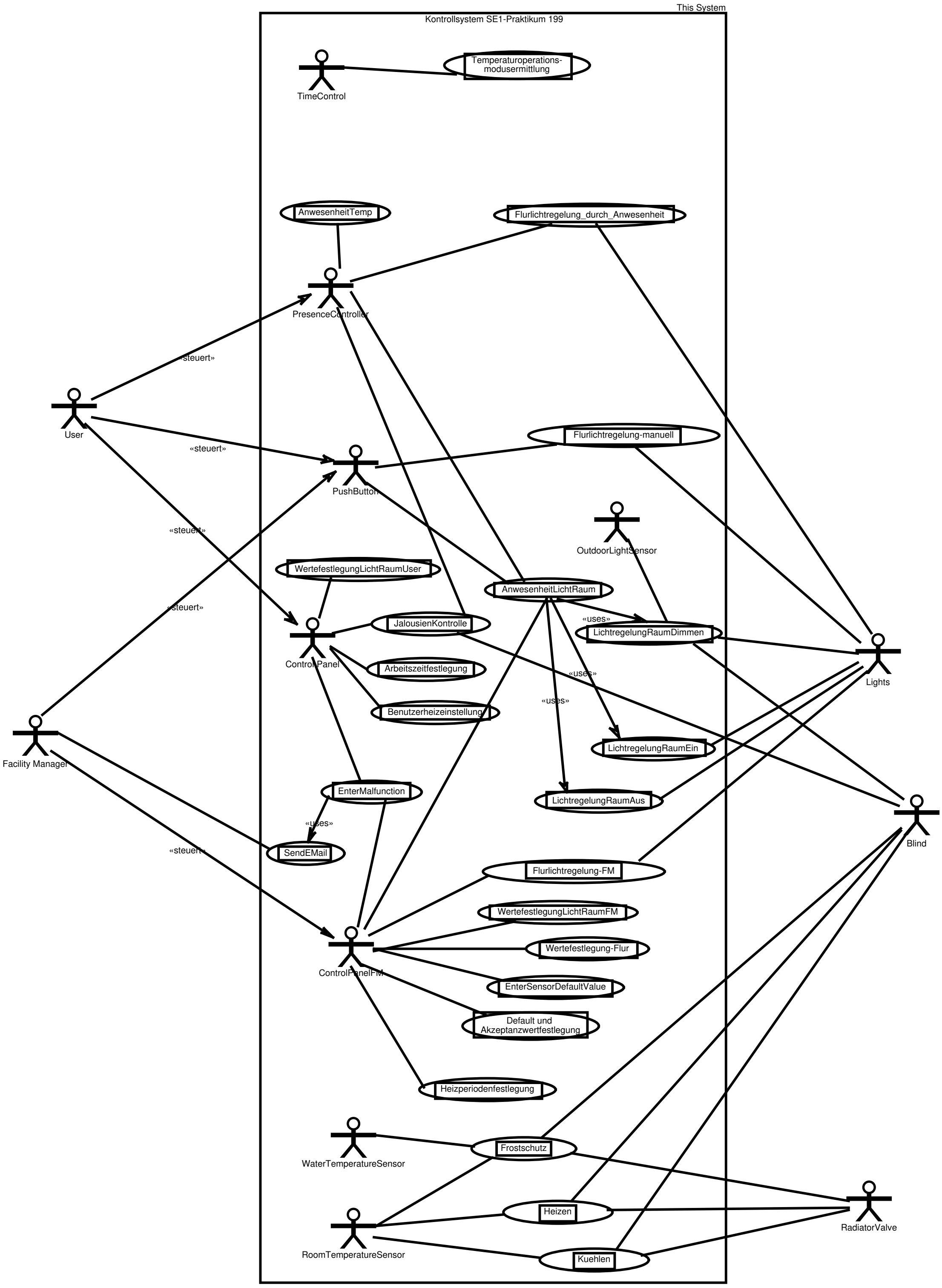
Keine.

2.2.3.68.2 Methodenschnittstellen der Klasse ZeitspannenVerlassen

Die Klasse ZeitspannenVerlassen besitzt keine Methodenschnittstellen.

2.2.4 UML Use Case-Diagramm

Da die Sequenzdiagramme des Entwurfs entsprechend den Use Cases der Analyse geordnet sind, wird an dieser Stelle das Use Case-Diagramm erneut wiederholt.

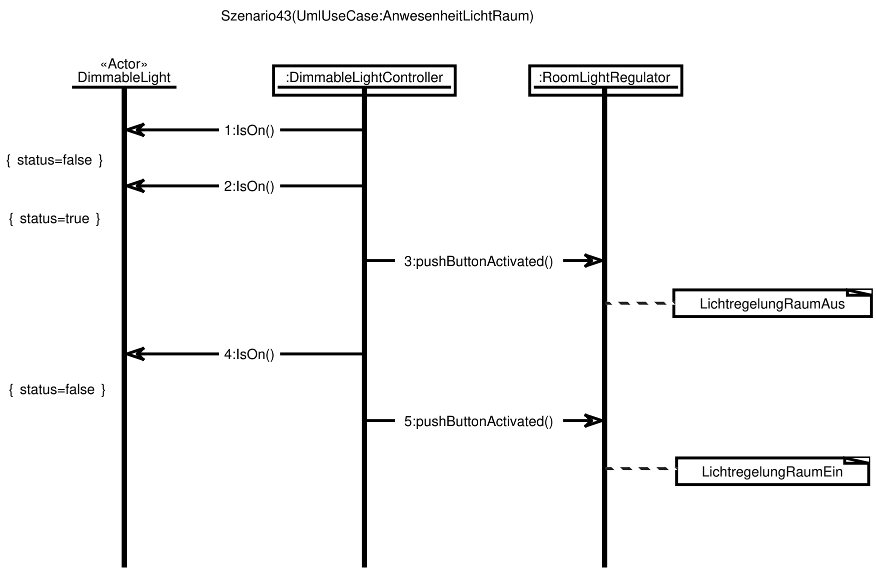


2.2.5 UML-Sequenzdiagramme

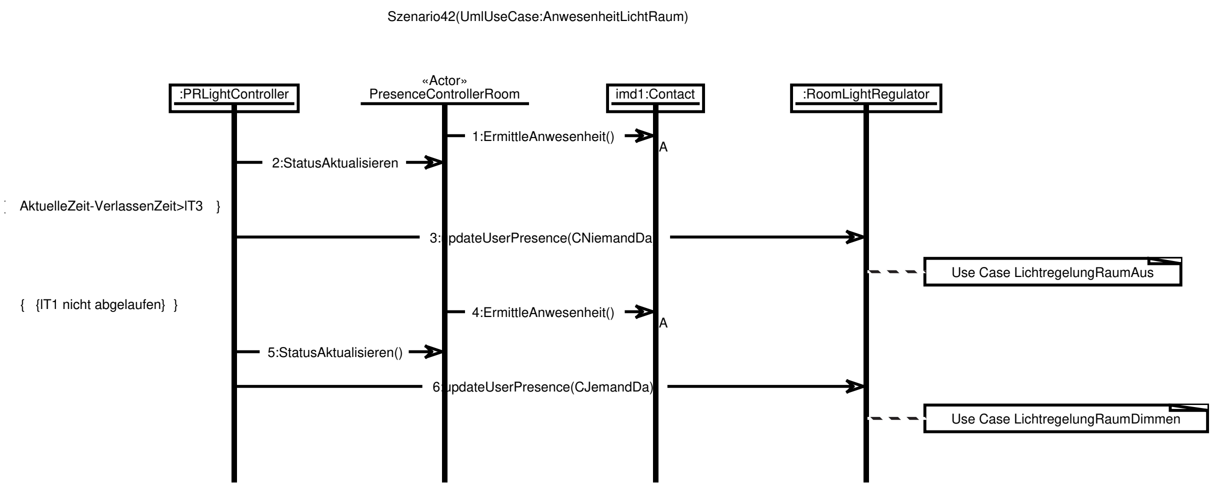
Die UML-Sequenzdiagramme des Entwurfs, basieren auf denen der UML Analyse, d.h. sie basieren auf den Szenarien der Analyse.Die Klassen in den UML-Sequenzdiagrammen wurden entsprechend den Änderungen im UML-Klassendiagramm angepasst. Alle Events aus der Analyse wurden in Methodenaufrufe aus dem Entwurf transformiert.

2.2.5.1 Sequenzdiagramme zu UseCase AnwesenheitLichtRaum

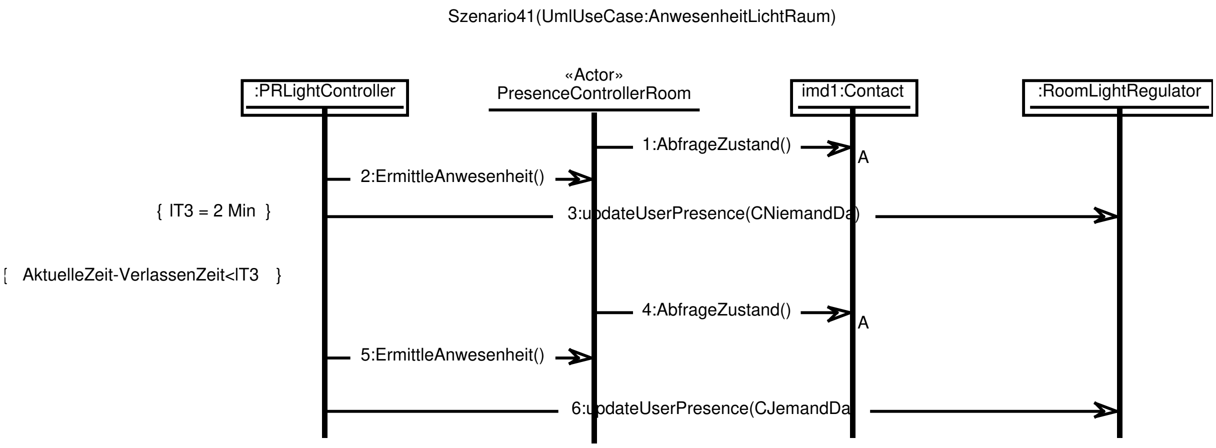
Sequence-Diagram zu Szenario43:



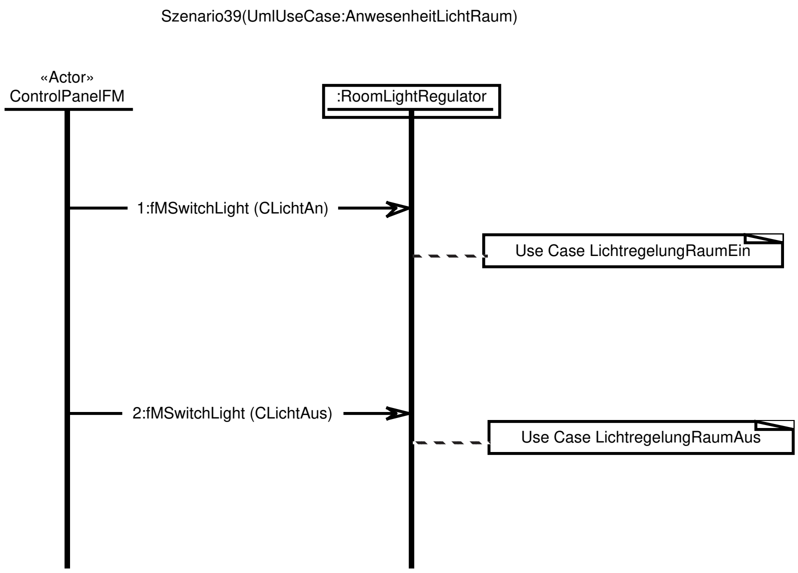
Sequence-Diagram zu Szenario42:



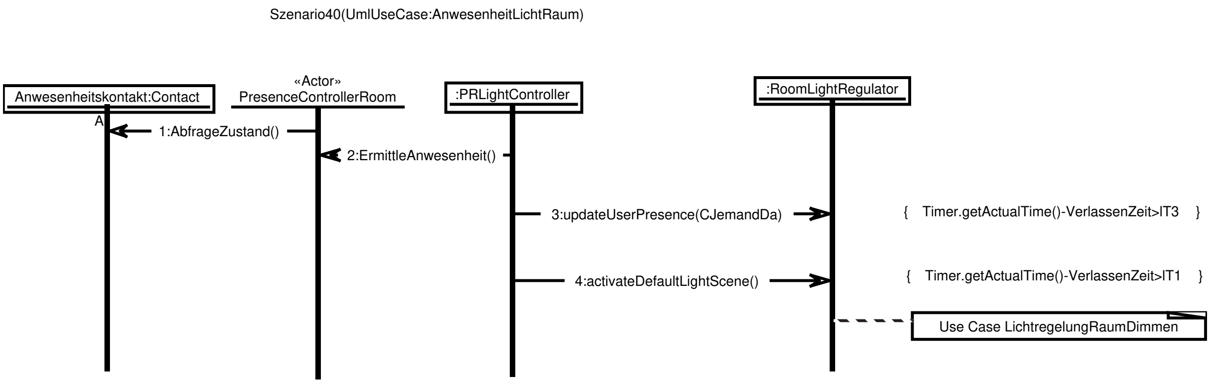
Sequence-Diagram zu Szenario41:



Sequence-Diagram zu Szenario39:

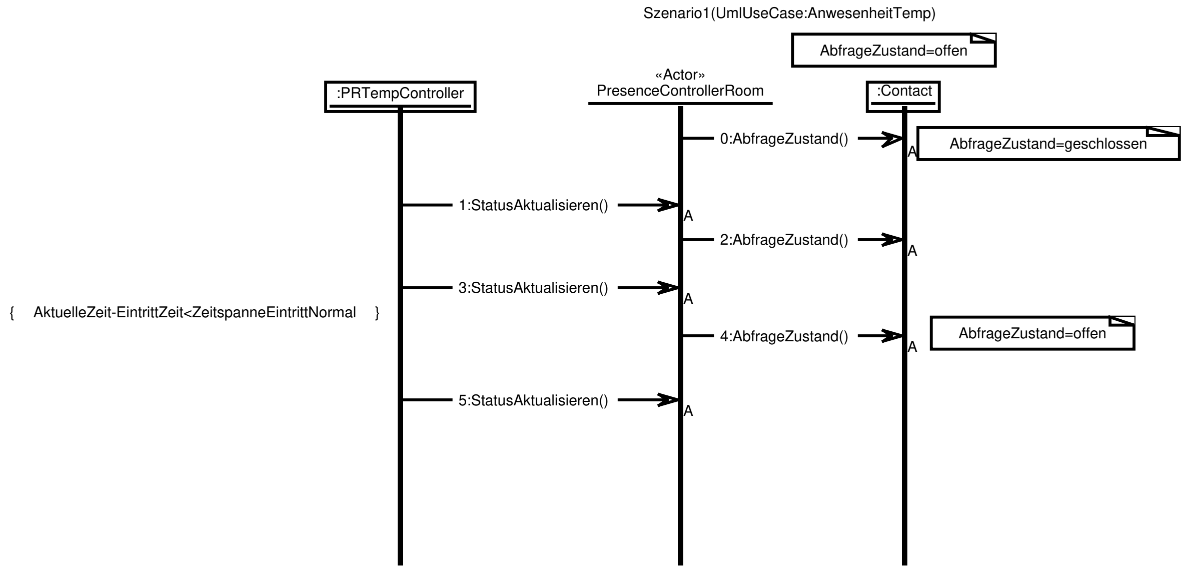


Sequence-Diagram zu Szenario40:

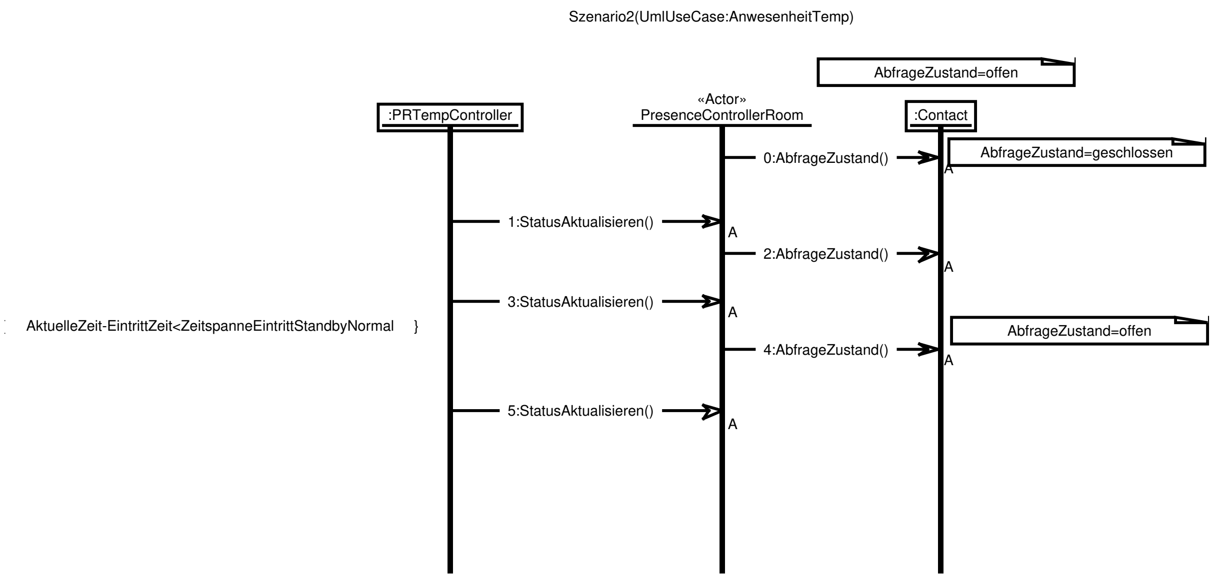


2.2.5.2 Sequenzdiagramme zu UseCase AnwesenheitTemp

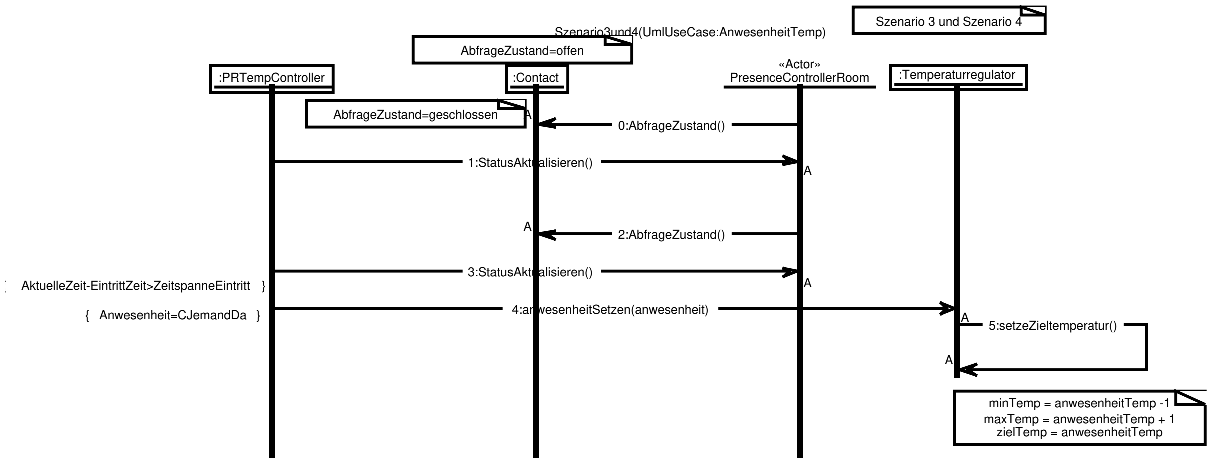
Sequence-Diagram zu Szenario1:



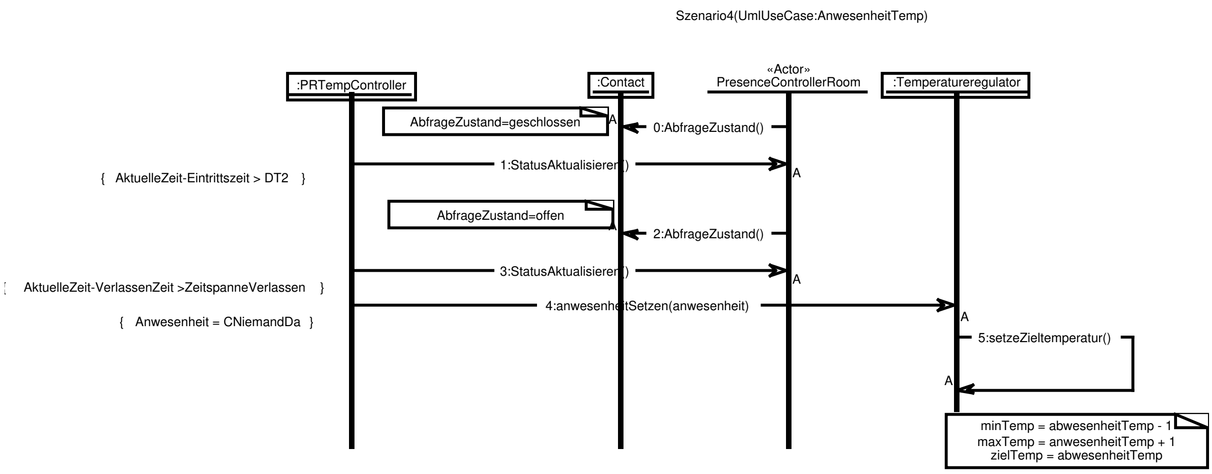
Sequence-Diagram zu Szenario2:



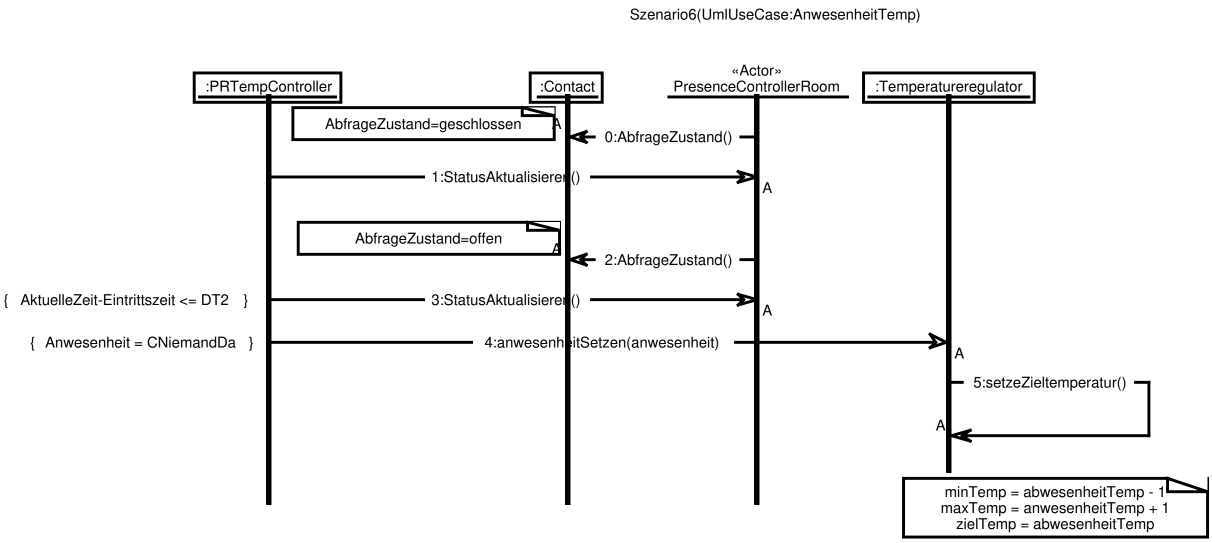
Sequence-Diagram zu Szenario3und4:



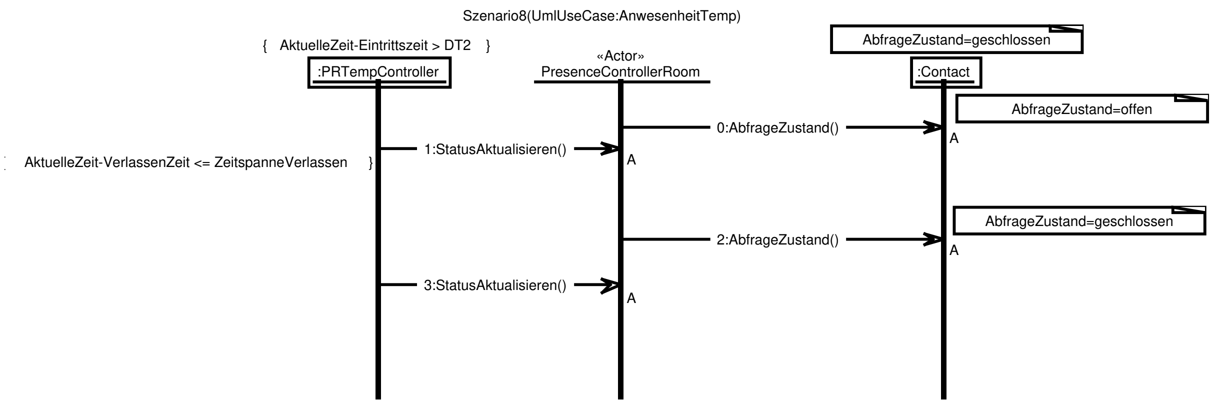
Sequence-Diagram zu Szenario4:



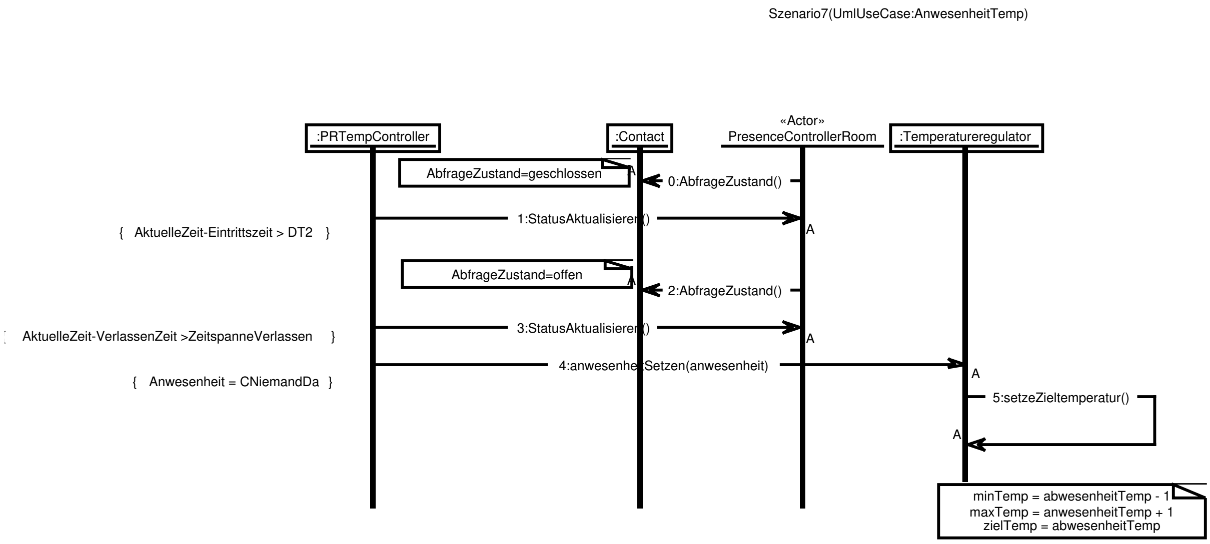
Sequence-Diagram zu Szenario6:



Sequence-Diagram zu Szenario8:

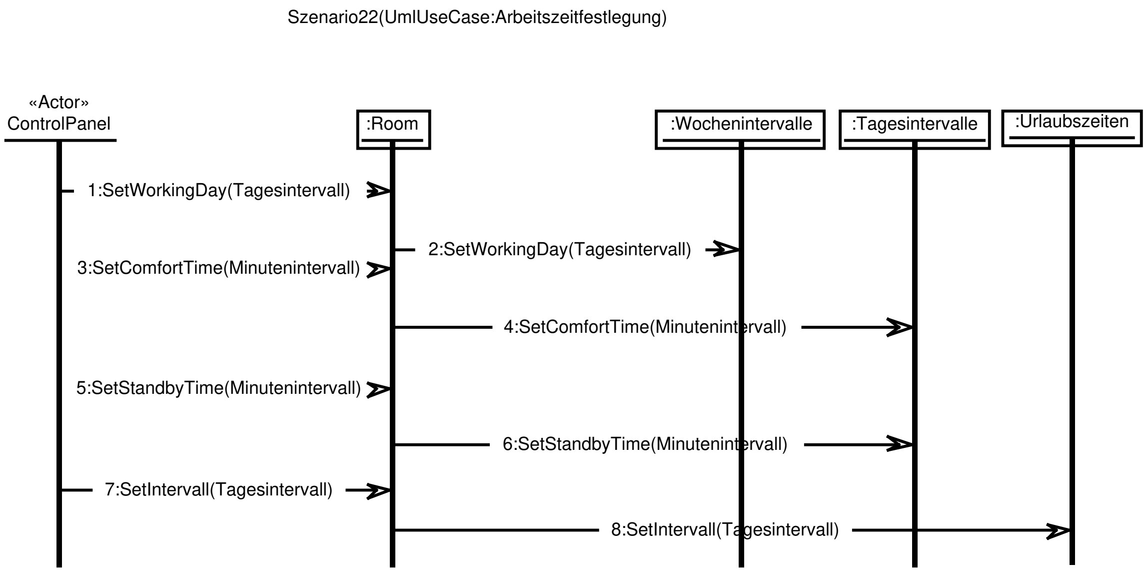


Sequence-Diagram zu Szenario7:

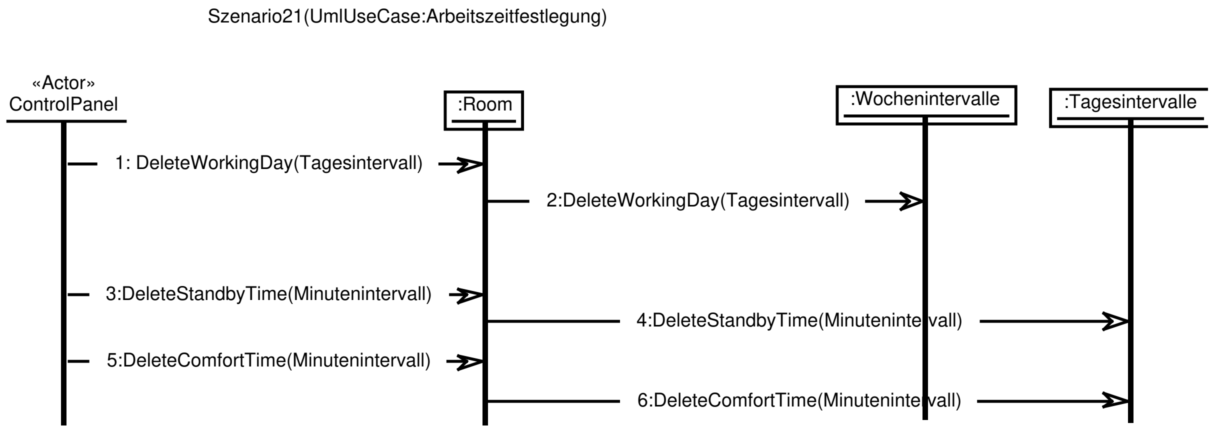


2.2.5.3 Sequenzdiagramme zu UseCase Arbeitszeitfestlegung

Sequence-Diagram zu Szenario22:



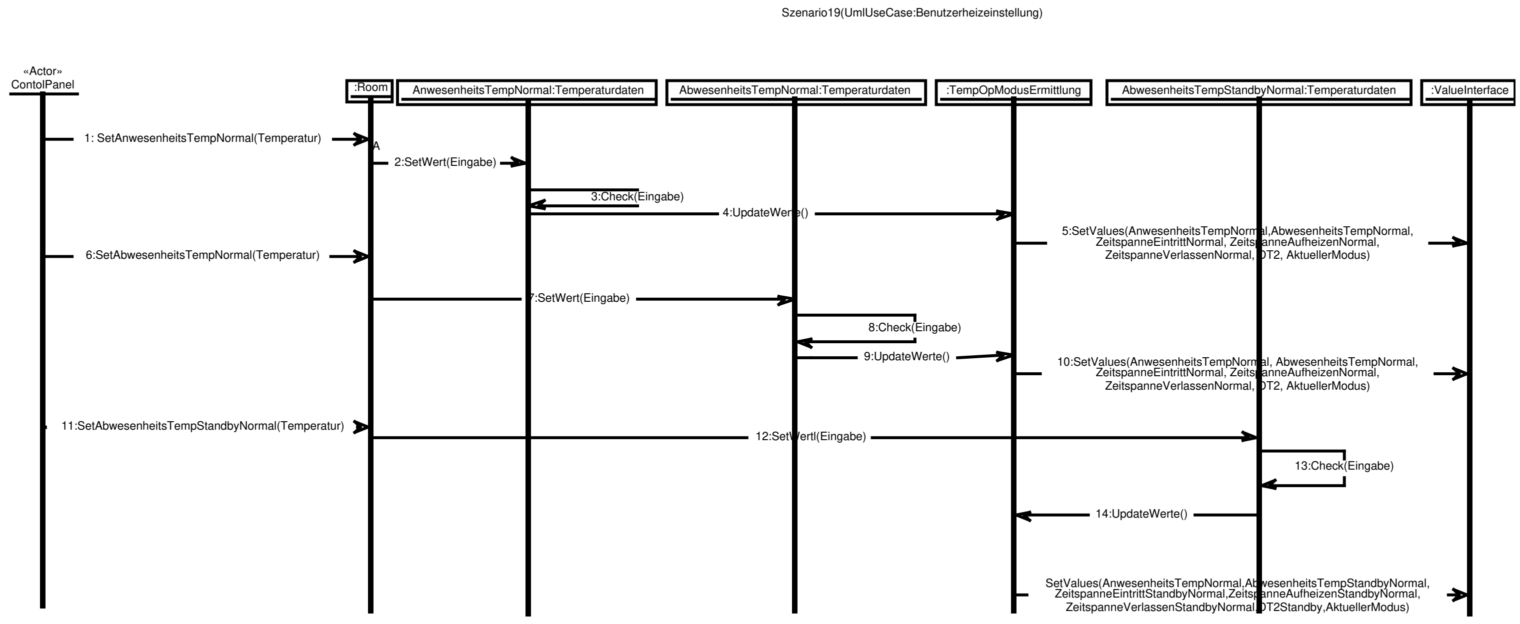
Sequence-Diagram zu Szenario21:

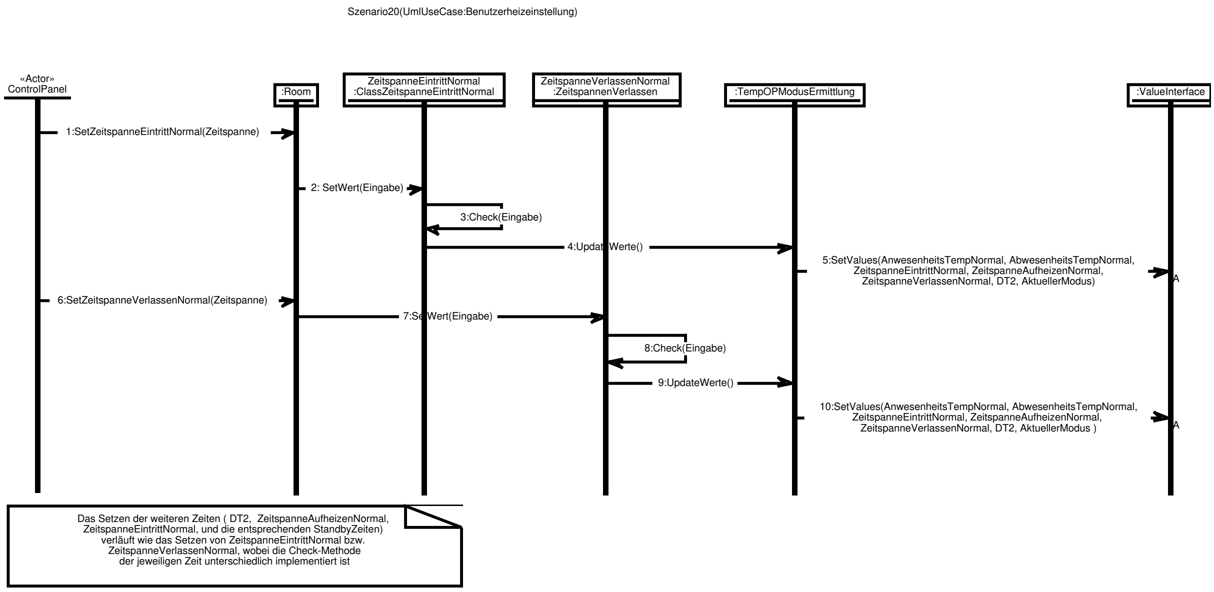


2.2.5.4 Sequenzdiagramme zu UseCase Benutzerheizeinstellung

Sequence-Diagram zu Szenario19:

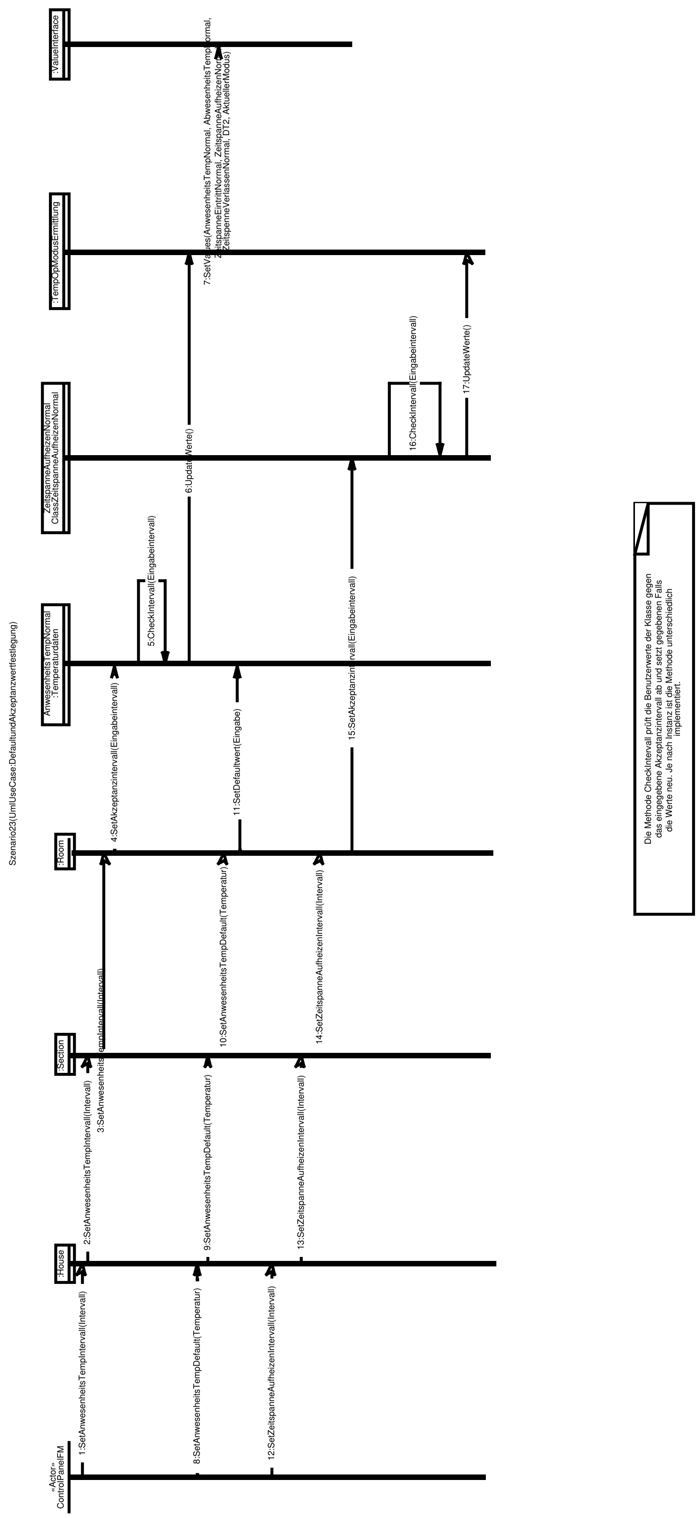
Sequence-Diagram zu Szenario20:





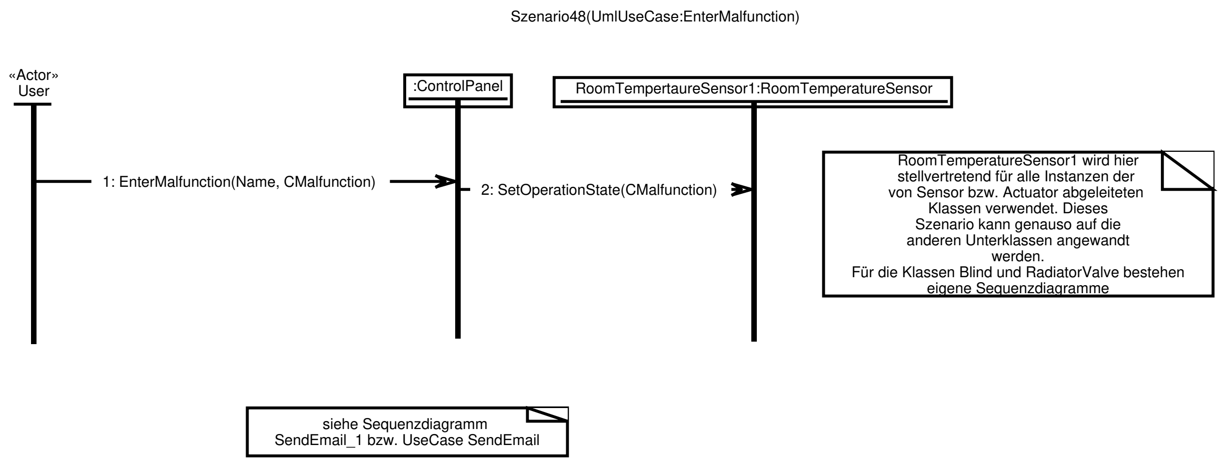
2.2.5.5 Sequenzdiagramme zu UseCase DefaultundAkzeptanzwertfestlegung

Sequence-Diagram zu Szenario23:

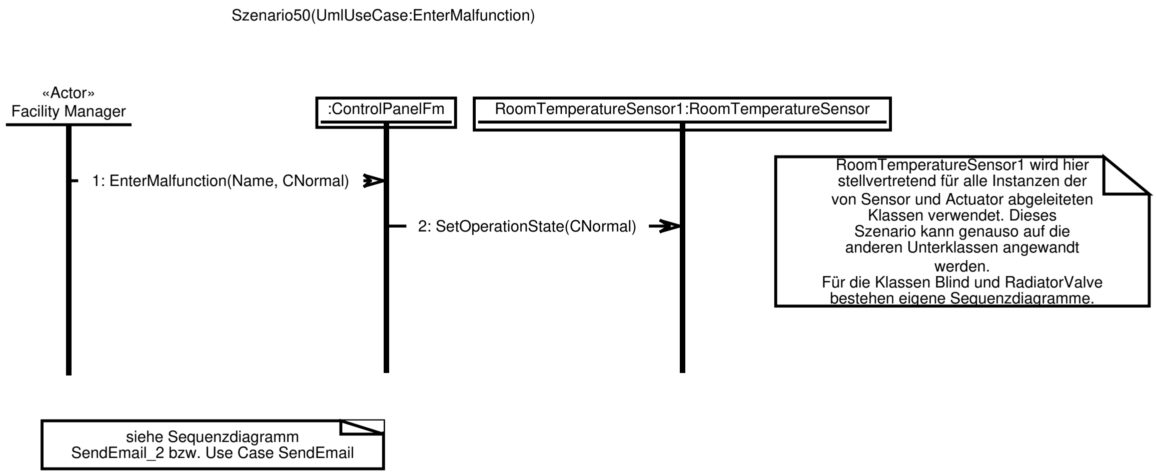


2.2.5.6 Sequenzdiagramme zu UseCase EnterMalfunction

Sequence-Diagram zu Szenario48:

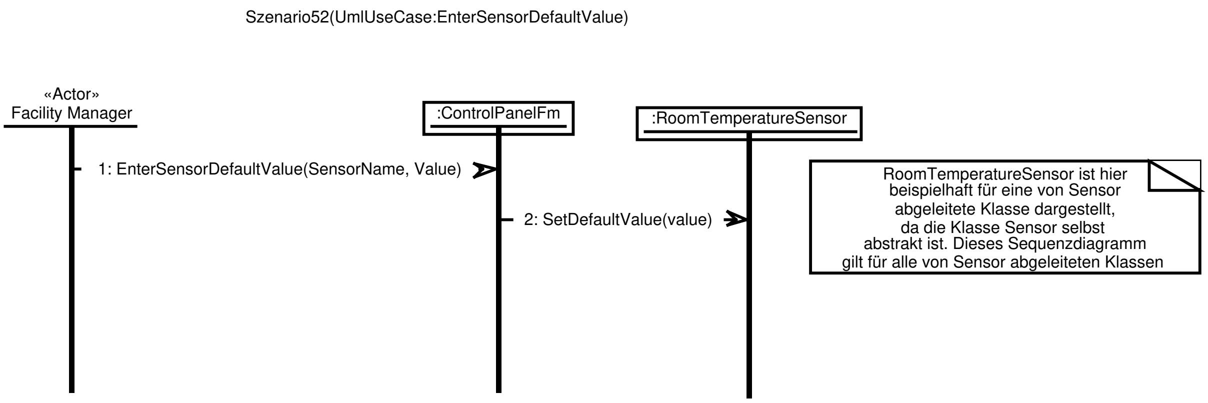


Sequence-Diagram zu Szenario50:



2.2.5.7 Sequenzdiagramme zu UseCase EnterSensorDefaultValue

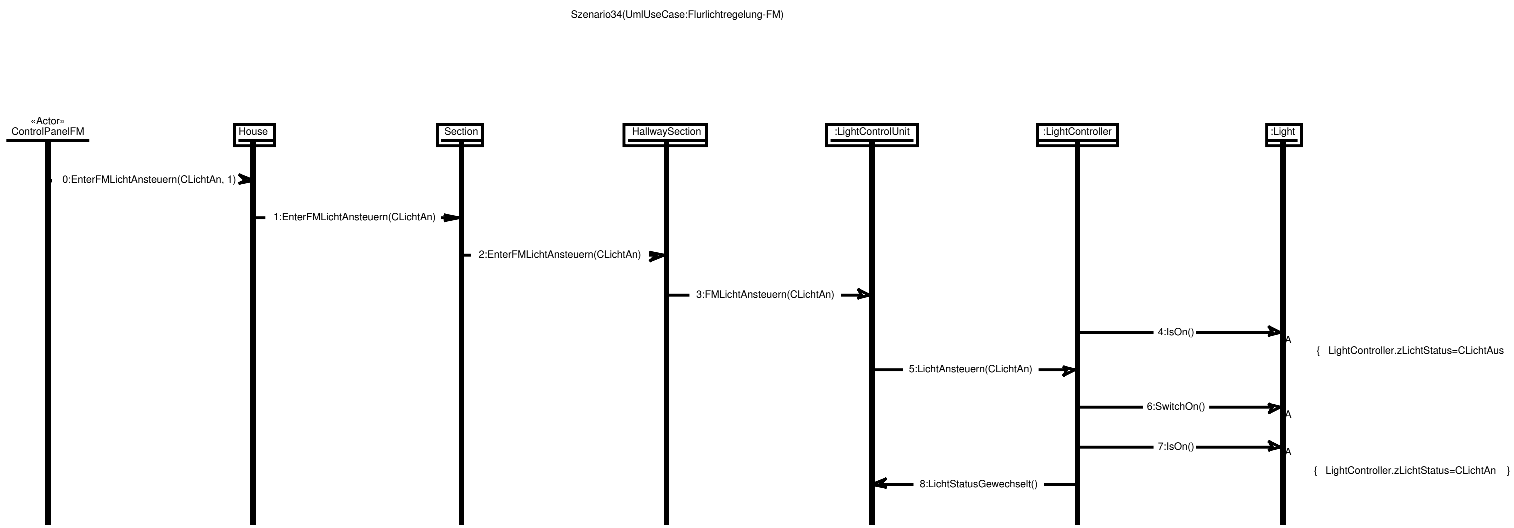
Sequence-Diagram zu Szenario52:



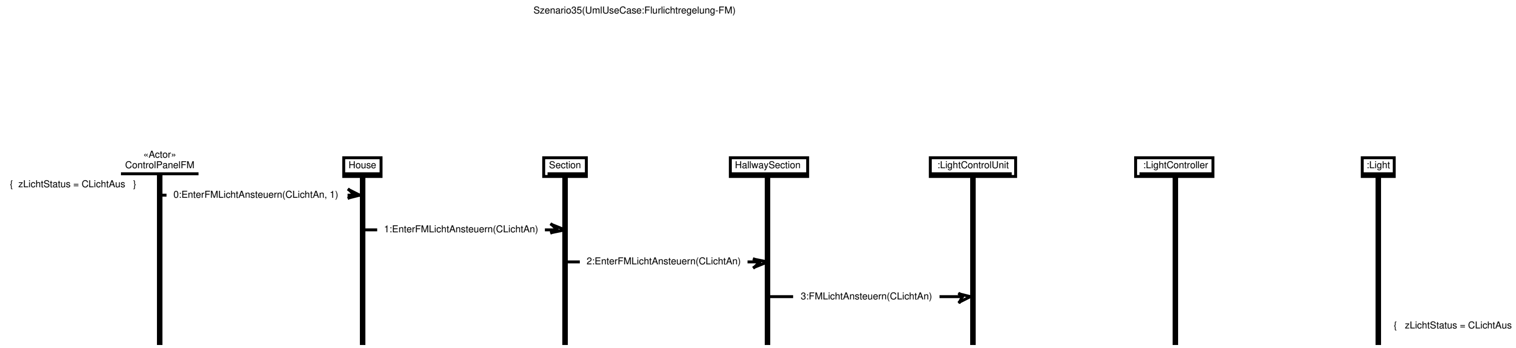
2.2.5.8 Sequenzdiagramme zu UseCase Flurlichtregelung-FM

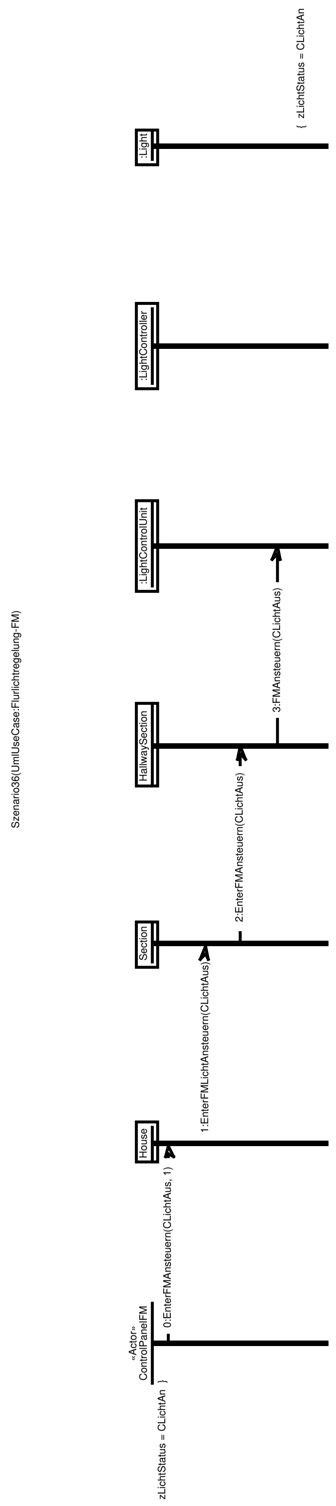
Sequence-Diagram zu Szenario34:

Sequence-Diagram zu Szenario35:



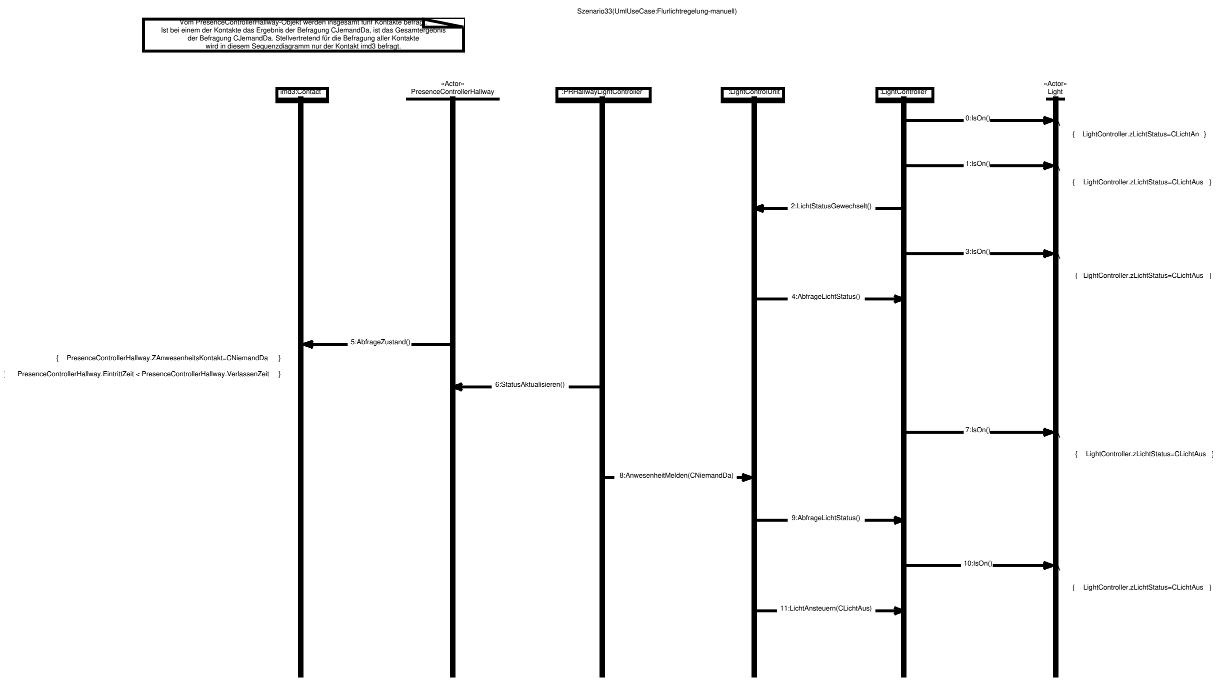
Sequence-Diagram zu Szenario36:





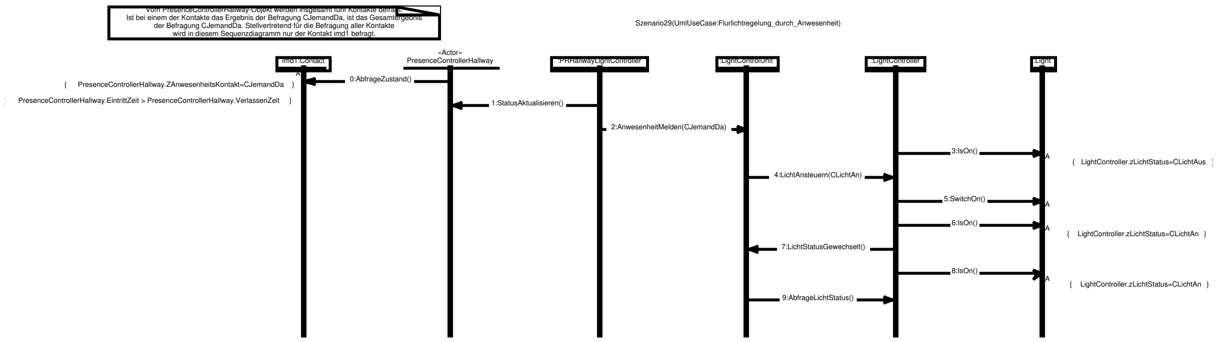
2.2.5.9 Sequenzdiagramme zu UseCase Flurlichtregelung-manuell

Sequence-Diagram zu Szenario33:

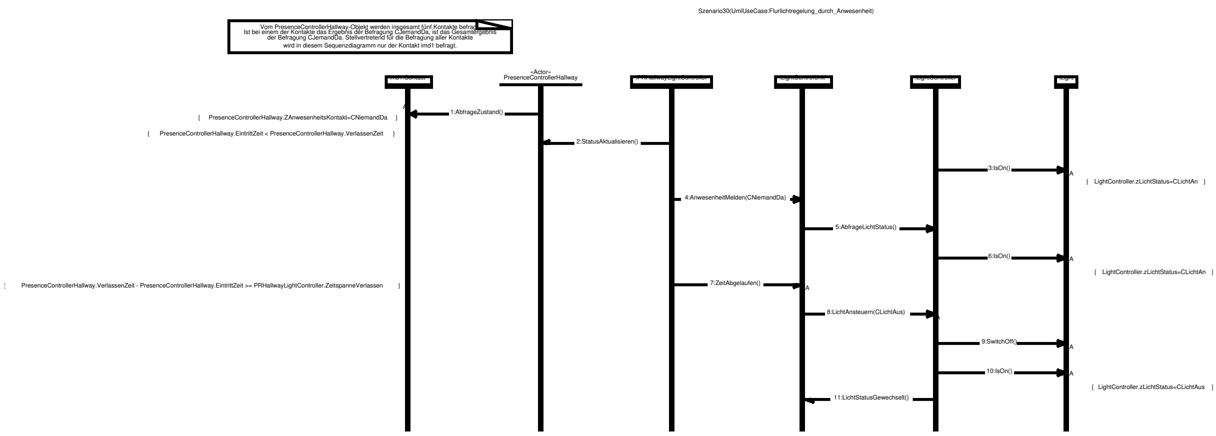


2.2.5.10 Sequenzdiagramme zu UseCase Flurlichtregelung_durch_Anwesenheit

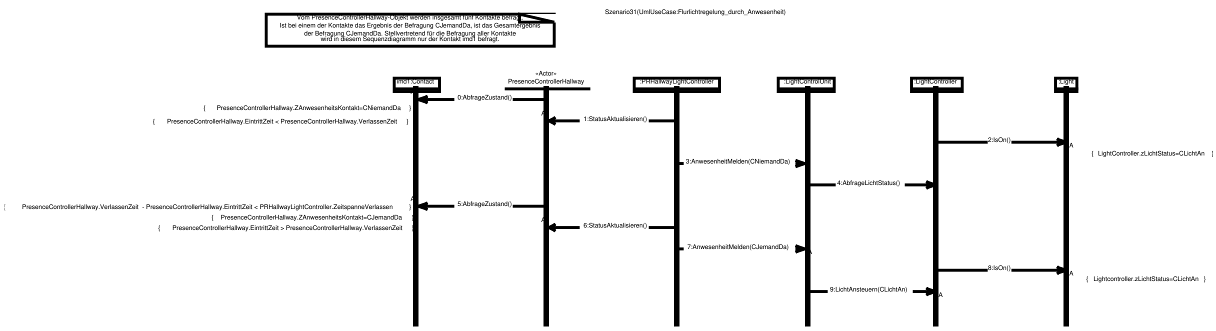
Sequence-Diagram zu Szenario29:



Sequence-Diagram zu Szenario30:

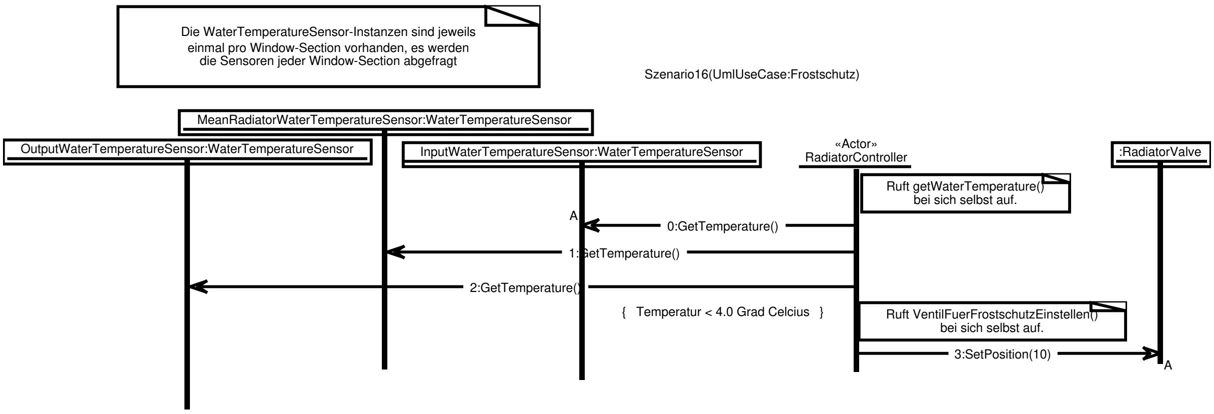


Sequence-Diagram zu Szenario31:

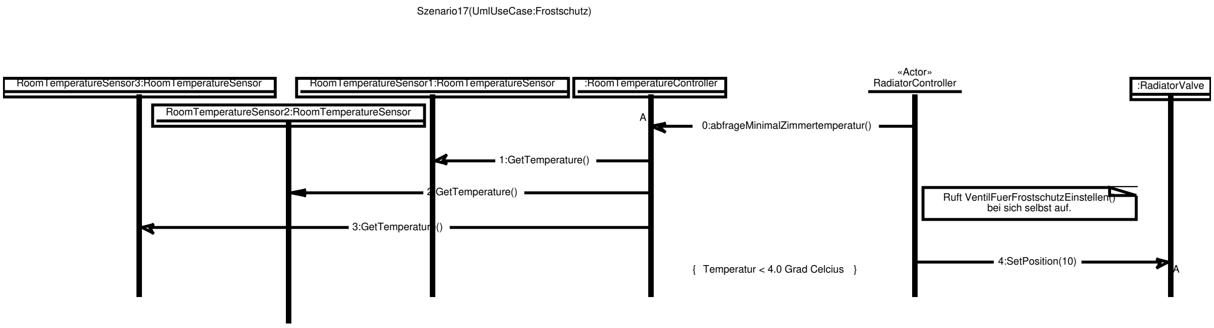


2.2.5.11 Sequenzdiagramme zu UseCase Frostschutz

Sequence-Diagram zu Szenario16:

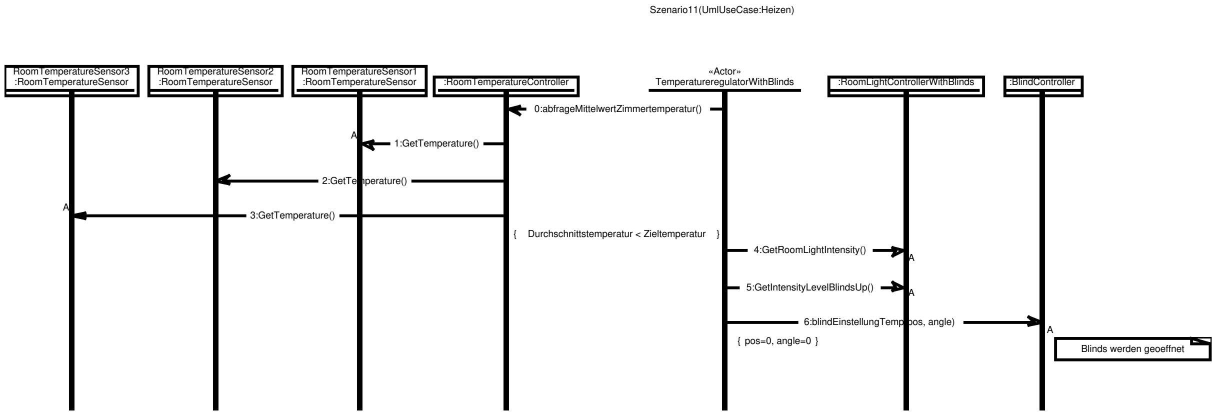


Sequence-Diagram zu Szenario17:



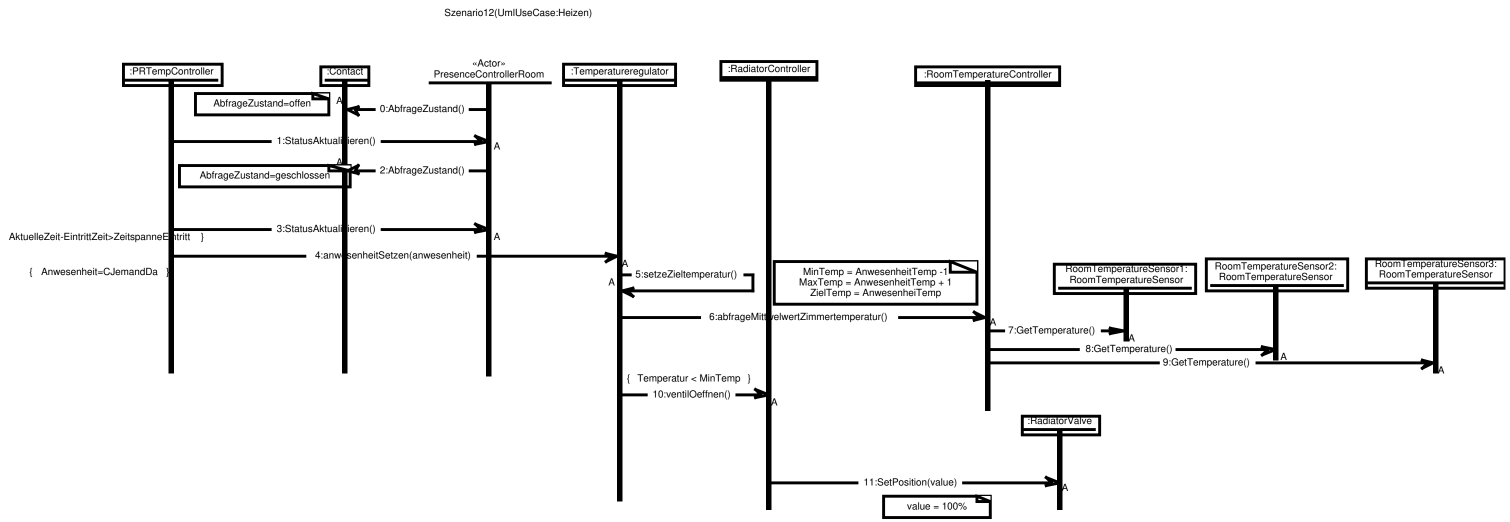
2.2.5.12 Sequenzdiagramme zu UseCase Heizen

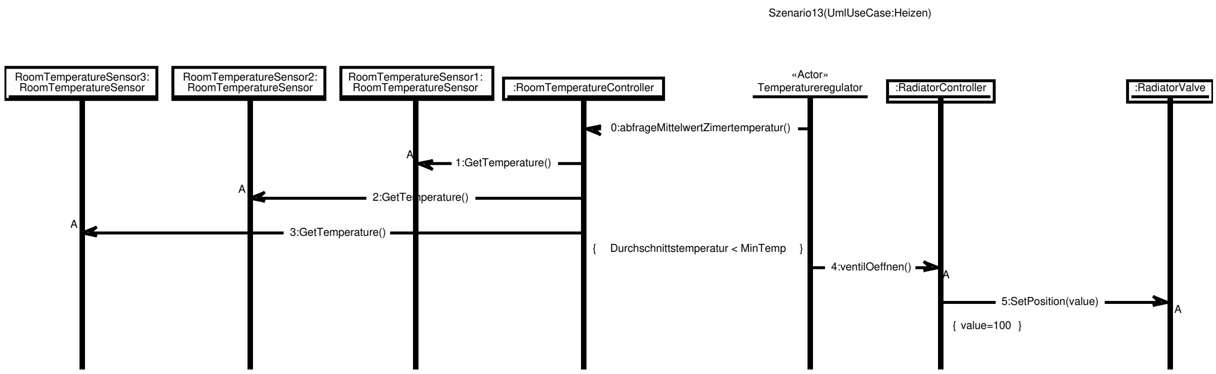
Sequence-Diagram zu Szenario11:



Sequence-Diagram zu Szenario12:

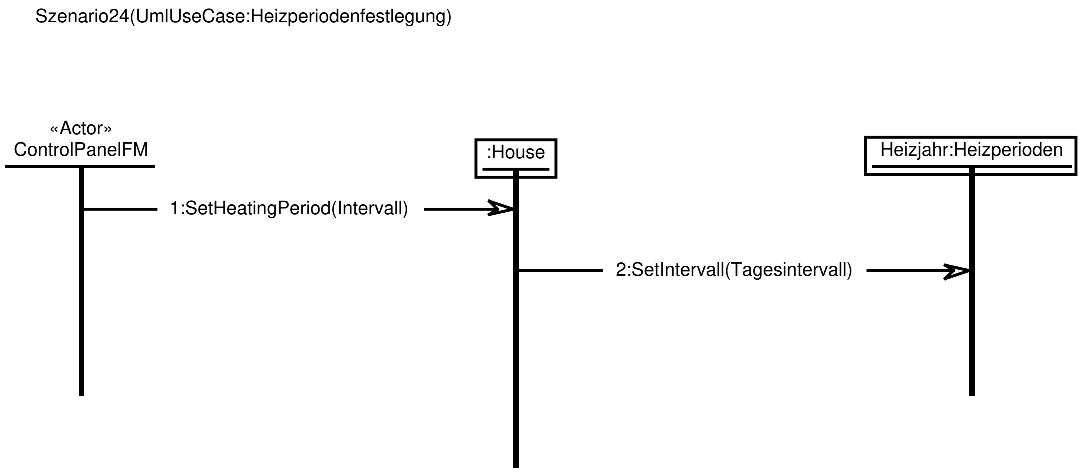
Sequence-Diagram zu Szenario13:



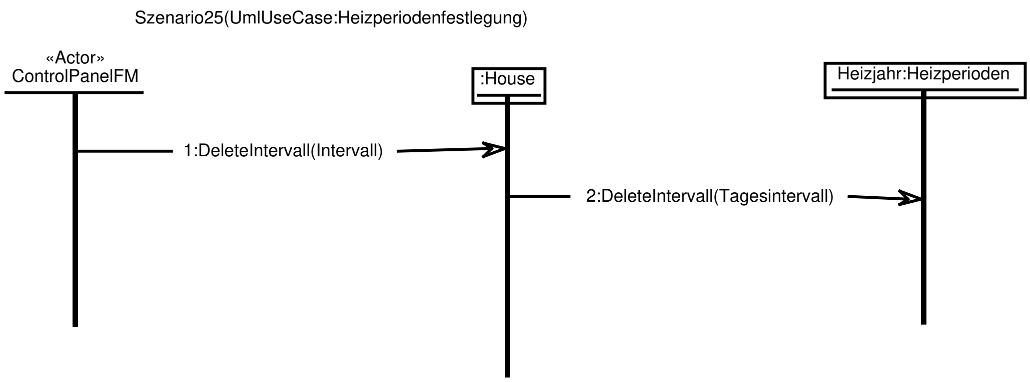


2.2.5.13 Sequenzdiagramme zu UseCase Heizperiodenfestlegung

Sequence-Diagram zu Szenario24:

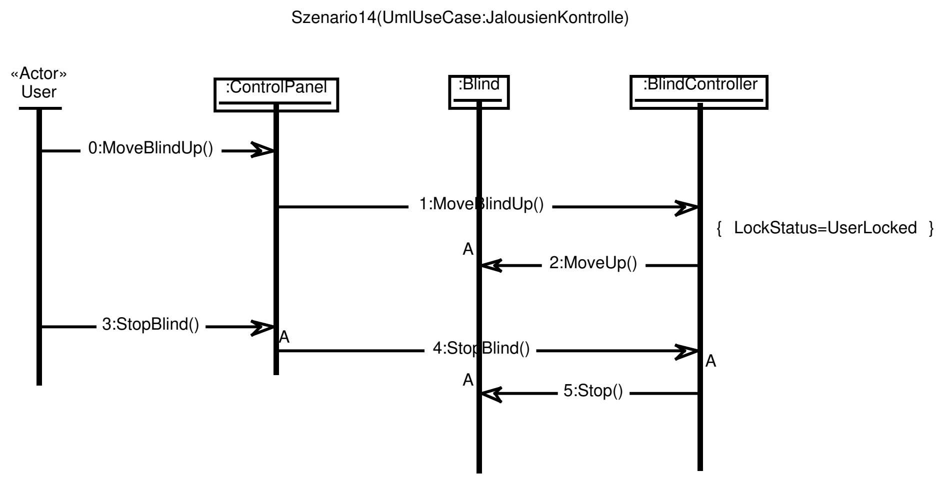


Sequence-Diagram zu Szenario25:

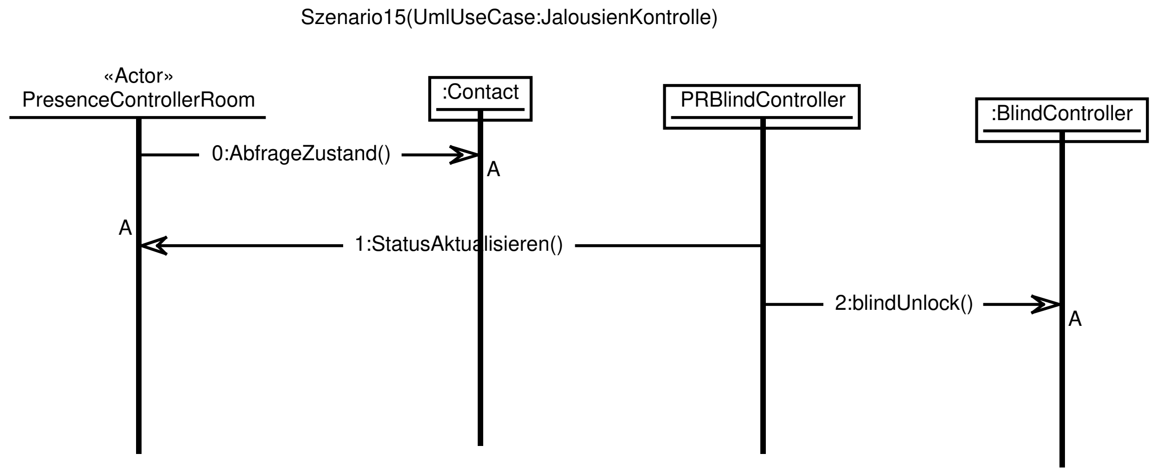


2.2.5.14 Sequenzdiagramme zu UseCase JalousienKontrolle

Sequence-Diagram zu Szenario14:



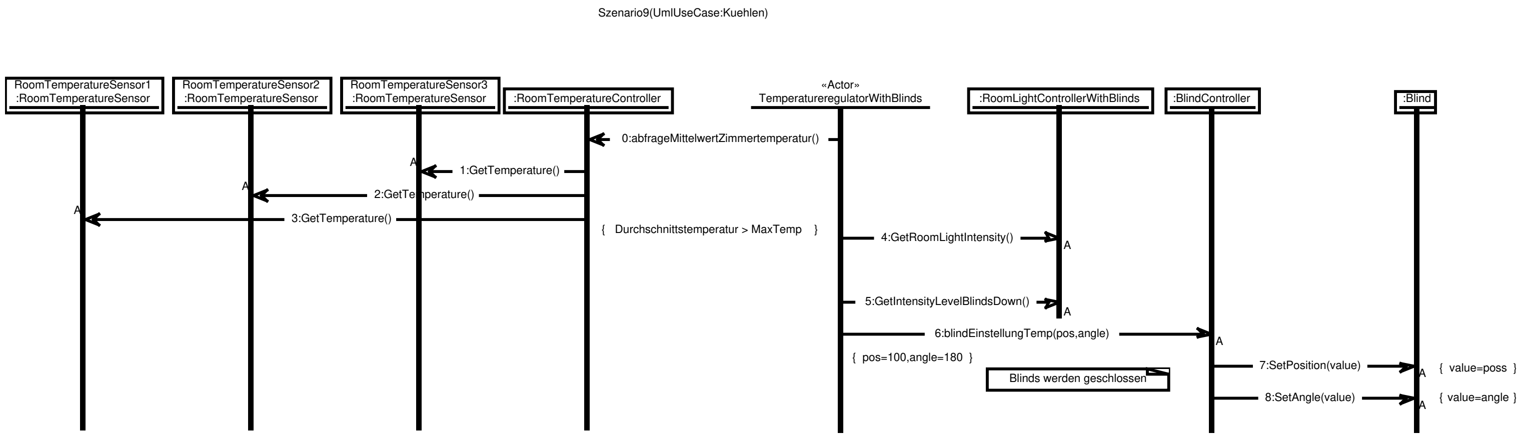
Sequence-Diagram zu Szenario15:

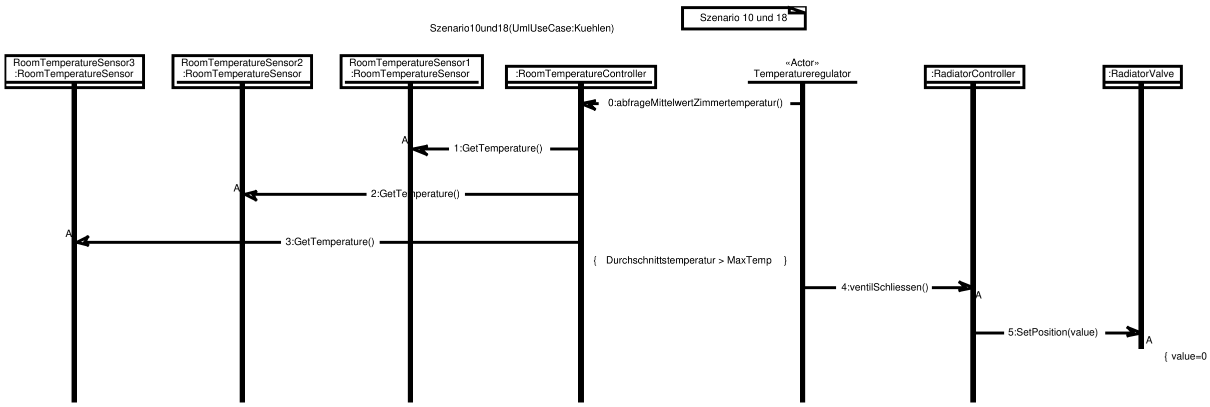


2.2.5.15 Sequenzdiagramme zu UseCase Kuehlen

Sequence-Diagram zu Szenario9:

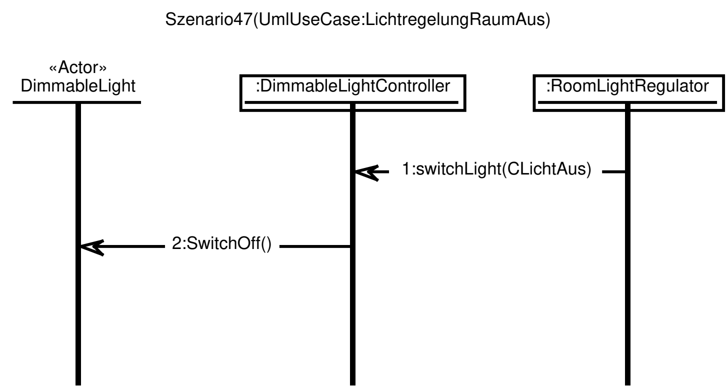
Sequence-Diagram zu Szenario10und18:





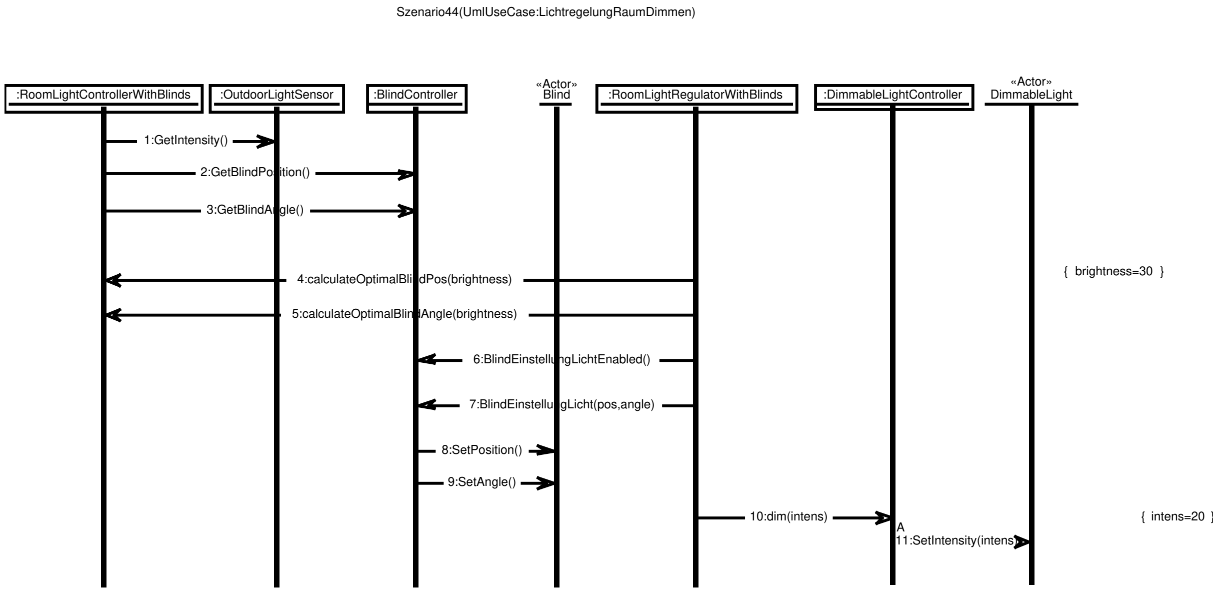
2.2.5.16 Sequenzdiagramme zu UseCase LichtregelungRaumAus

Sequence-Diagram zu Szenario47:

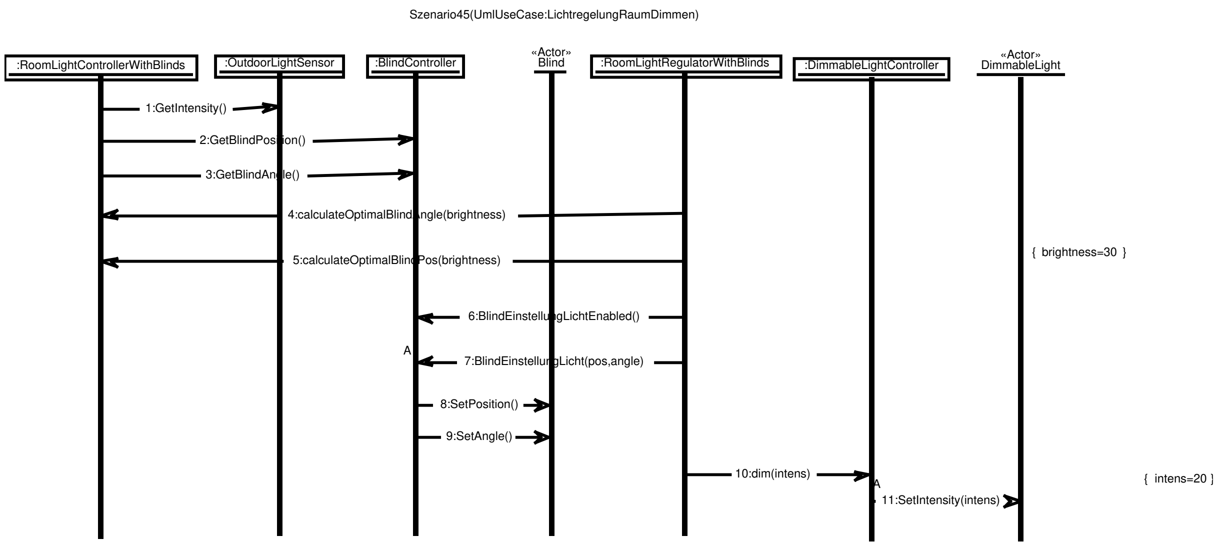


2.2.5.17 Sequenzdiagramme zu UseCase LichtregelungRaumDimmen

Sequence-Diagram zu Szenario44:

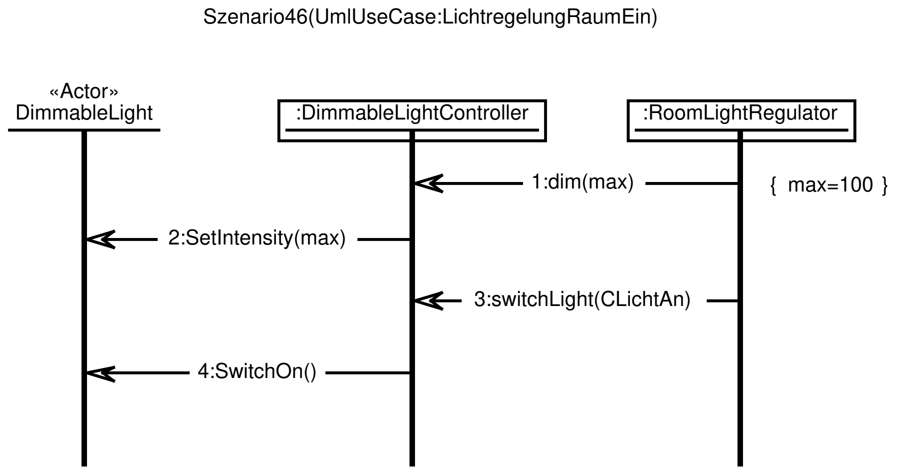


Sequence-Diagram zu Szenario45:



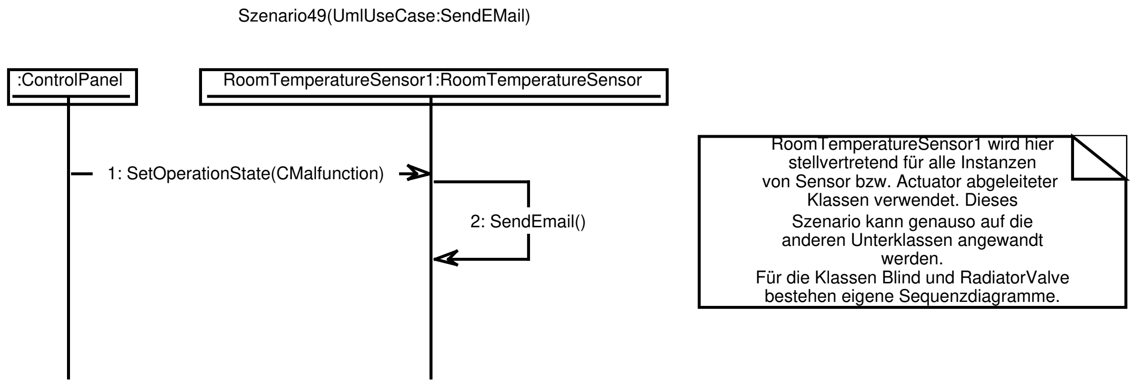
2.2.5.18 Sequenzdiagramme zu UseCase LichtregelungRaumEin

Sequence-Diagram zu Szenario46:

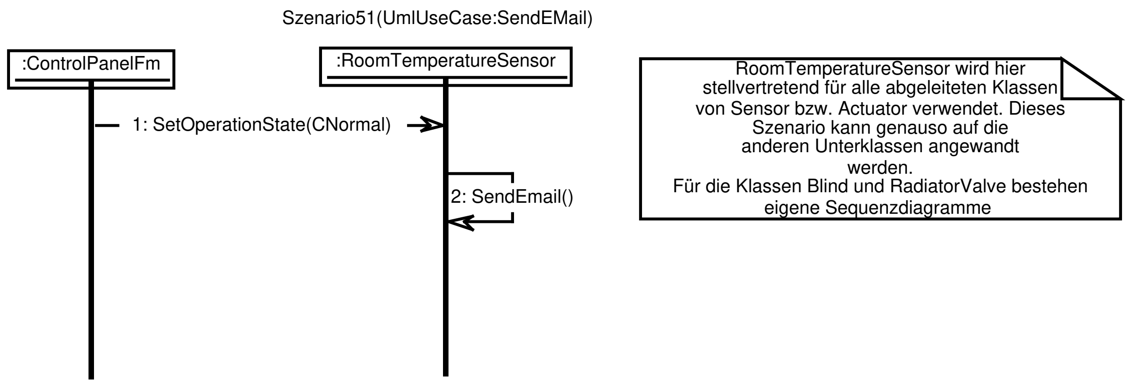


2.2.5.19 Sequenzdiagramme zu UseCase SendEMail

Sequence-Diagram zu Szenario49:

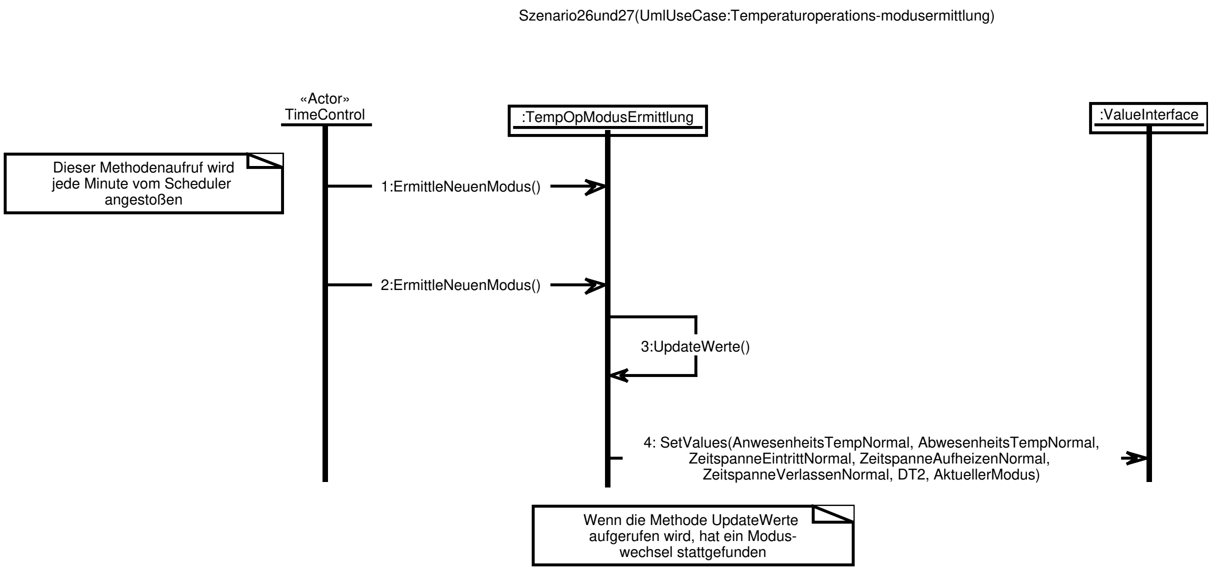


Sequence-Diagram zu Szenario51:

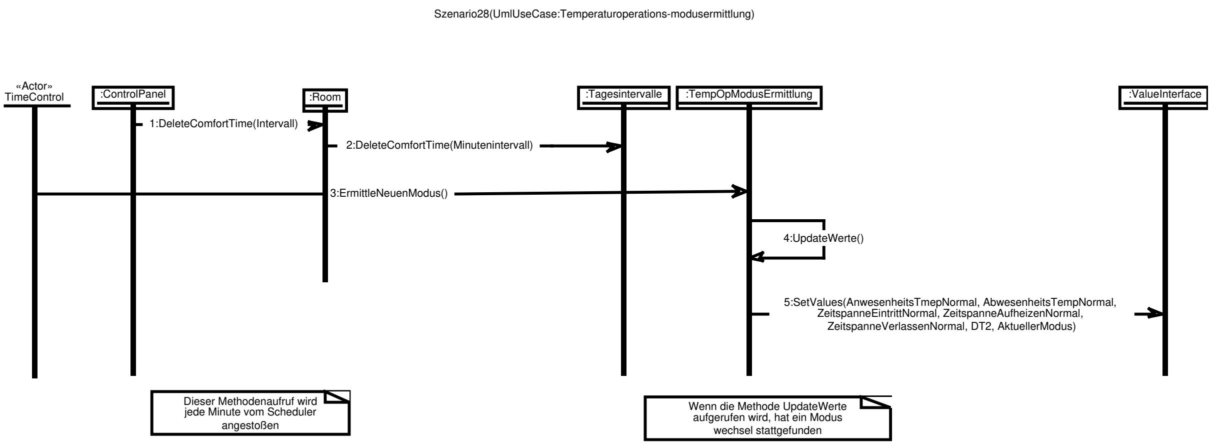


2.2.5.20 Sequenzdiagramme zu UseCase Temperaturoperations-modusermittlung

Sequence-Diagram zu Szenario26und27:

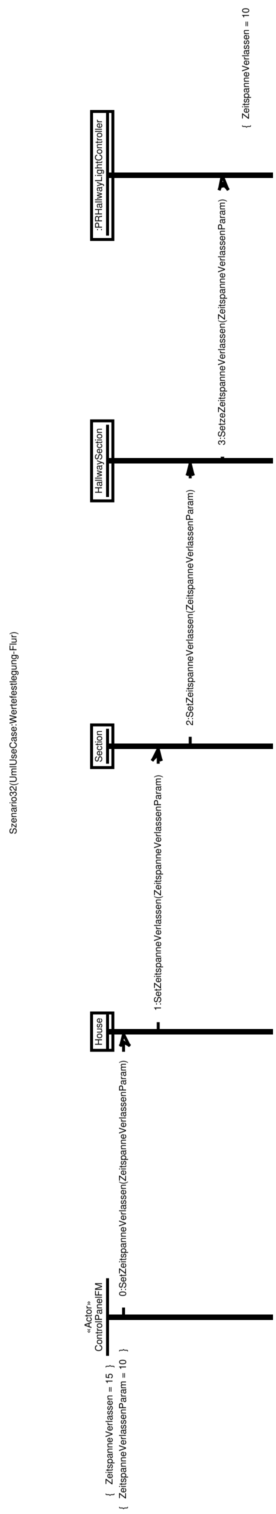


Sequence-Diagram zu Szenario28:



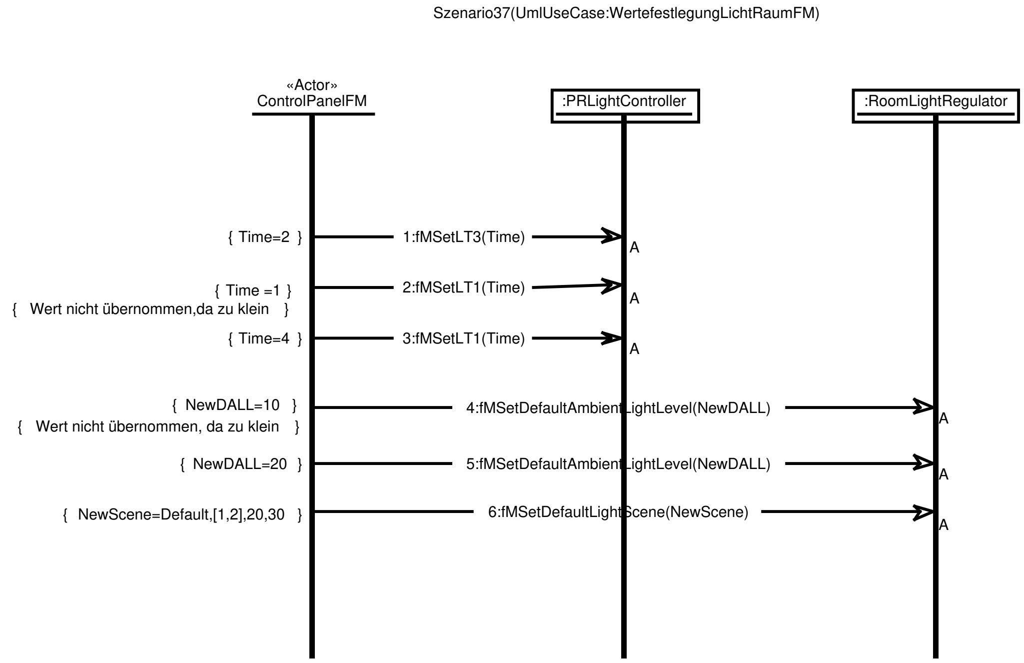
2.2.5.21 Sequenzdiagramme zu UseCase Wertefestlegung-Flur

Sequence-Diagram zu Szenario32:



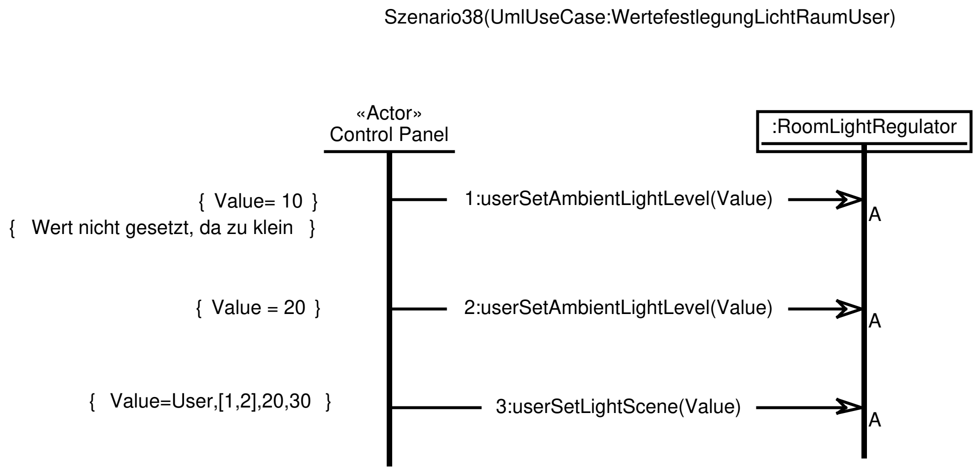
2.2.5.22 Sequenzdiagramme zu UseCase WertefestlegungLichtRaumFM

Sequence-Diagram zu Szenario37:



2.2.5.23 Sequenzdiagramme zu UseCase WertefestlegungLichtRaumUser

Sequence-Diagram zu Szenario38:

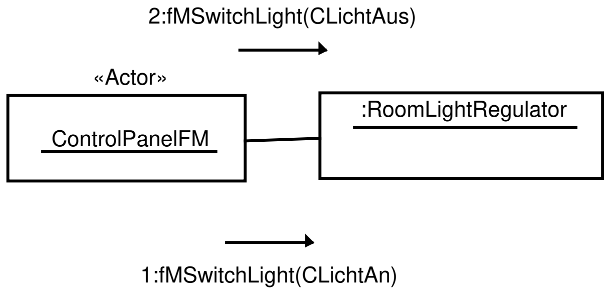


2.2.6 UML-Kollaborationsdiagramme

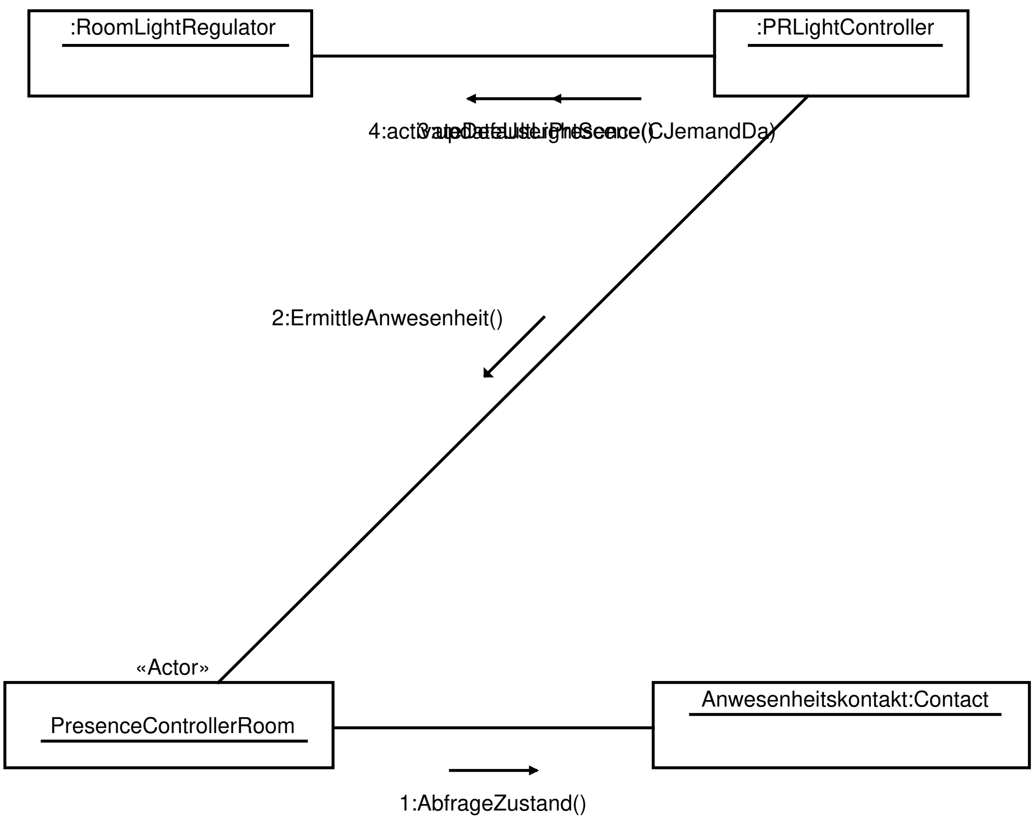
Im folgenden sind die Kollaborationsdiagramme der Objekte der UML-Klassendiagramme aufgeführt. Sie wurden mit Hilfe des Werkzeugs StP/UML automatisch aus den Sequenzdiagrammen abgeleitet.

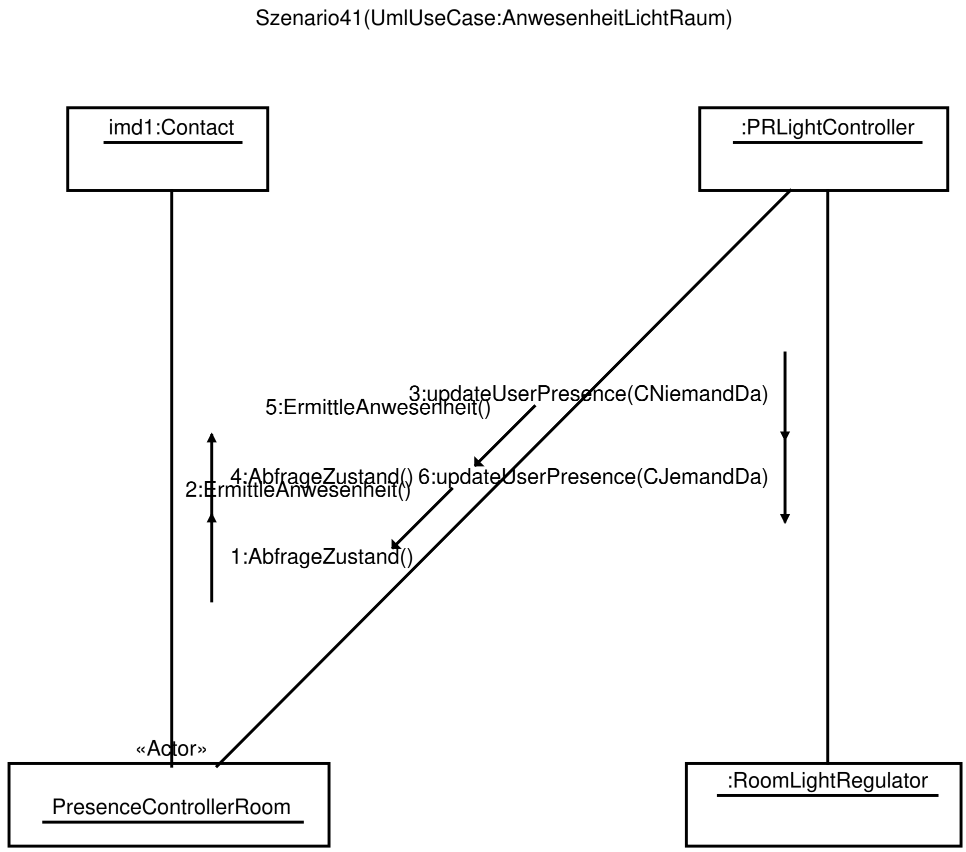
2.2.6.1 AnwesenheitLichtRaum

Szenario39(UmlUseCase:AnwesenheitLichtRaum)

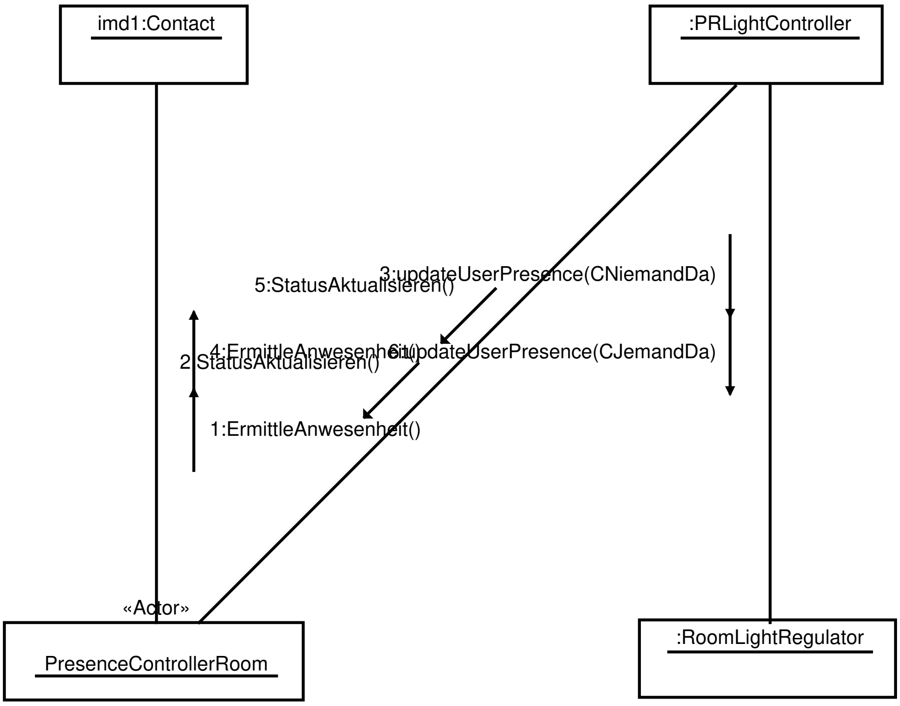


Szenario40(UmlUseCase:AnwesenheitLichtRaum)

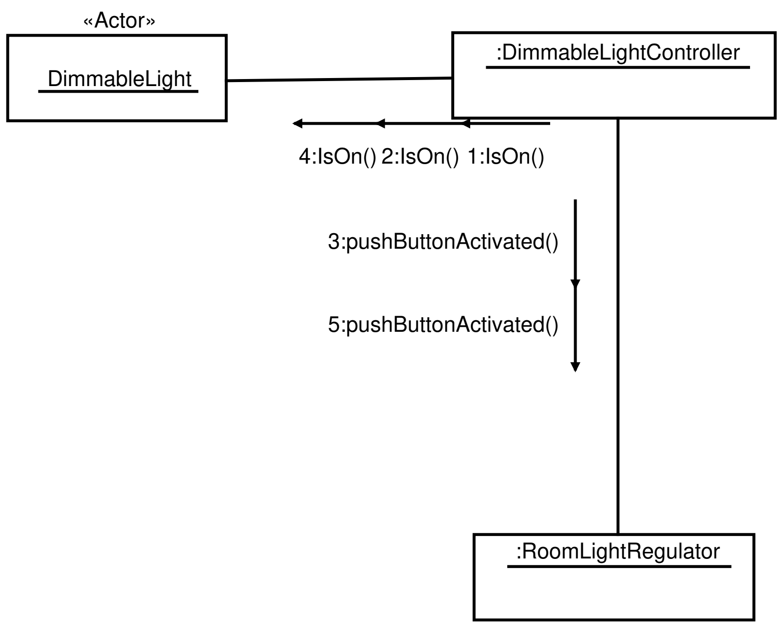



Szenario41(UmlUseCase:AnwesenheitLichtRaum)
imd1:Contact
:PRLightController
3:updateUserPresence(CNiemandDa)
5:ErmittleAnwesenheit()
4:AbfrageZustand()
6:updateUserPresence(CJemandDa)
2:ErmittleAnwesenheit()
1:AbfrageZustand()
«Actor»
PresenceControllerRoom
:RoomLightRegulator

Szenario42(UmlUseCase:AnwesenheitLichtRaum)

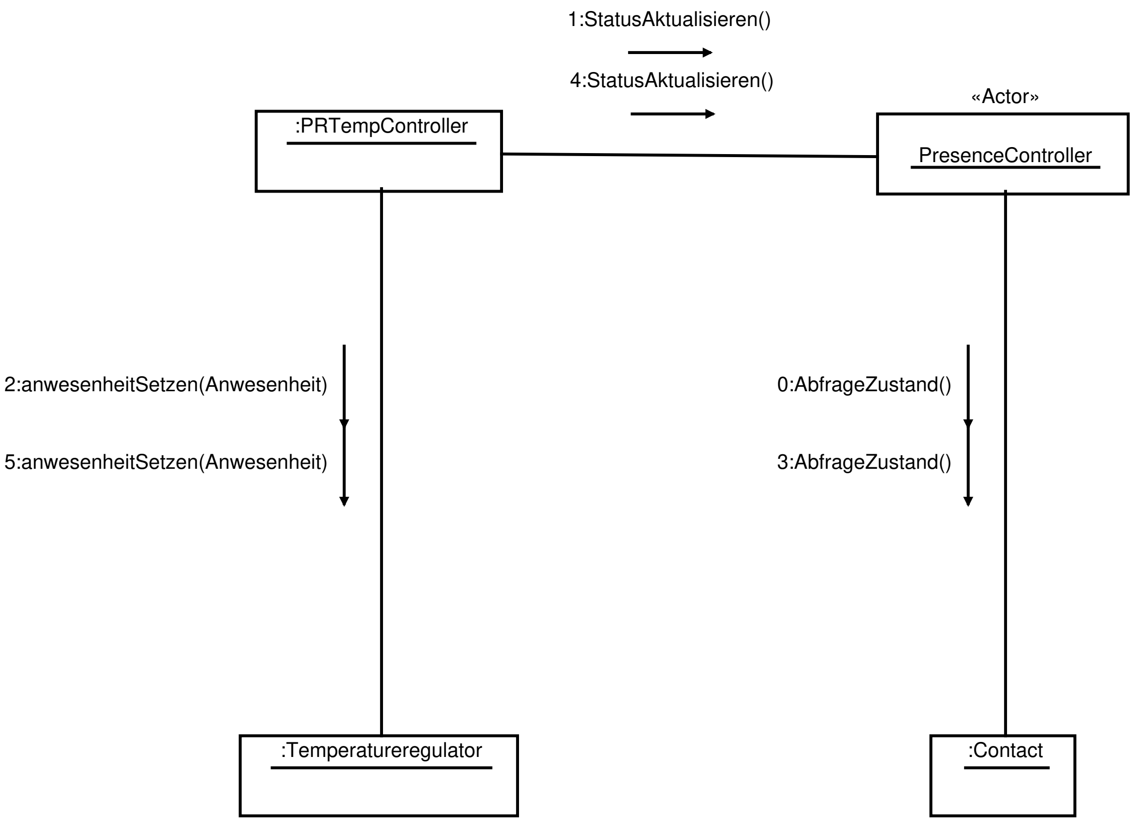


Szenario43(UmlUseCase:AnwesenheitLichtRaum)

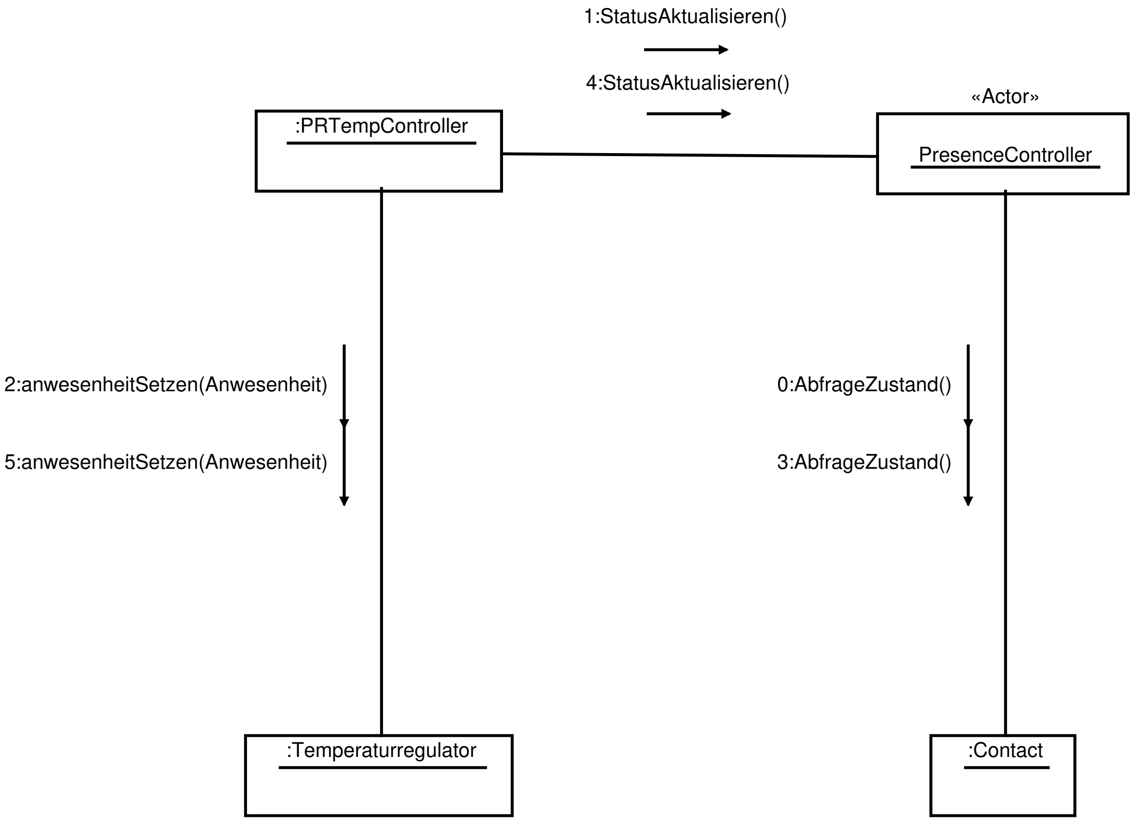


2.2.6.2 AnwesenheitTemp

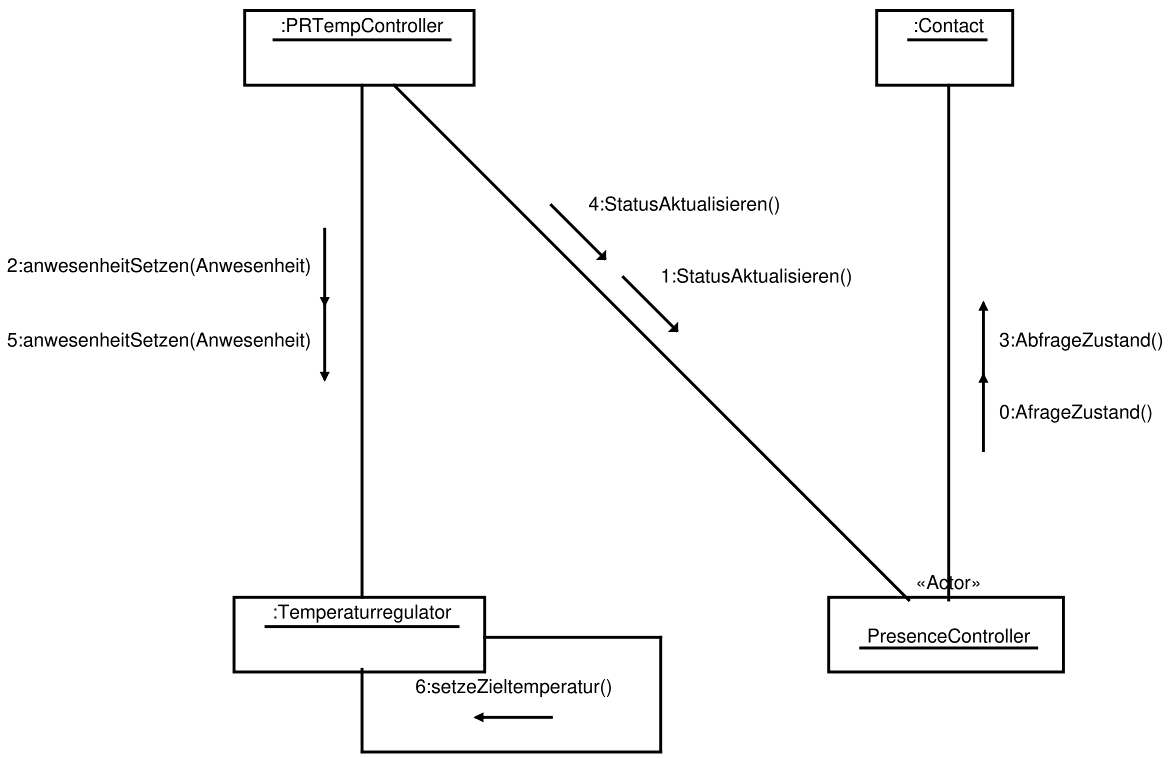
Szenario1(UmlUseCase:AnwesenheitTemp)



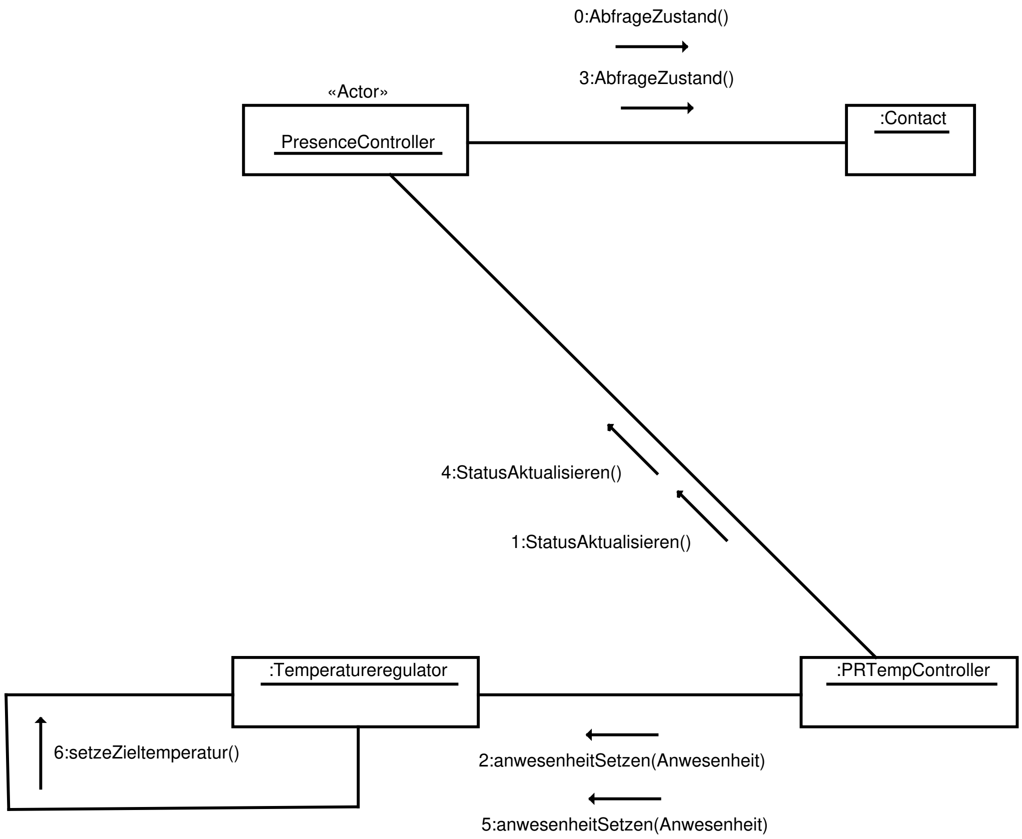
Szenario2(UmlUseCase:AnwesenheitTemp)



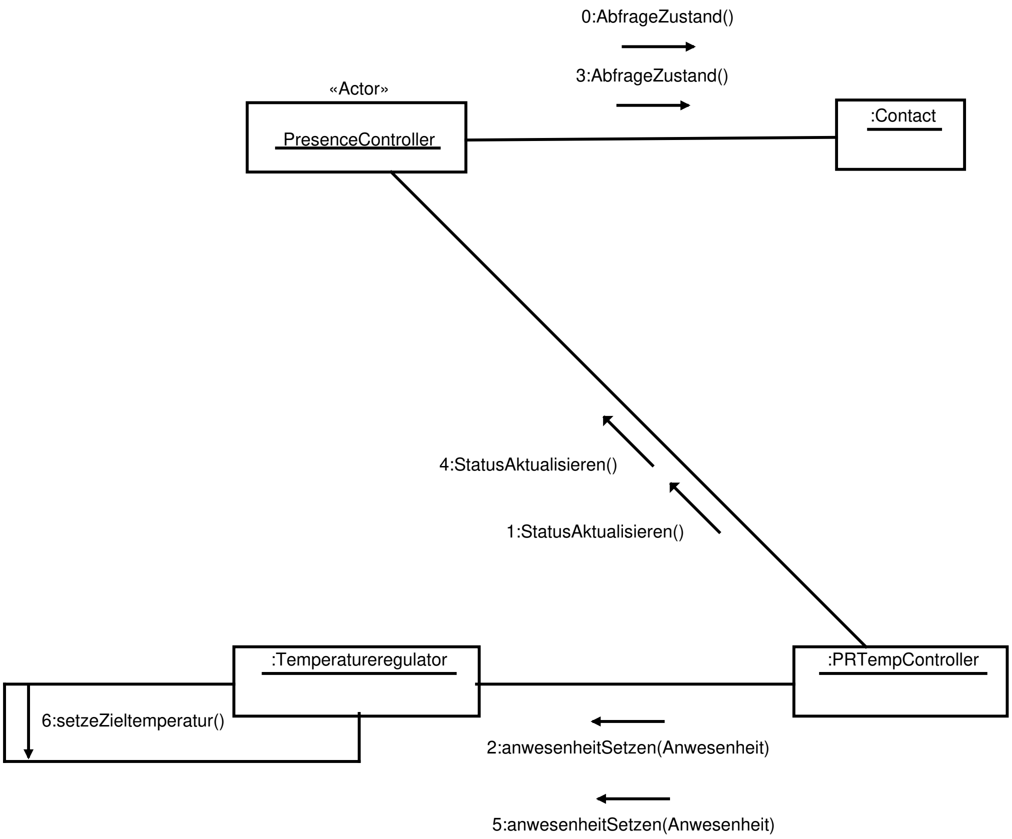
Szenario3und4(UmlUseCase:AnwesenheitTemp)



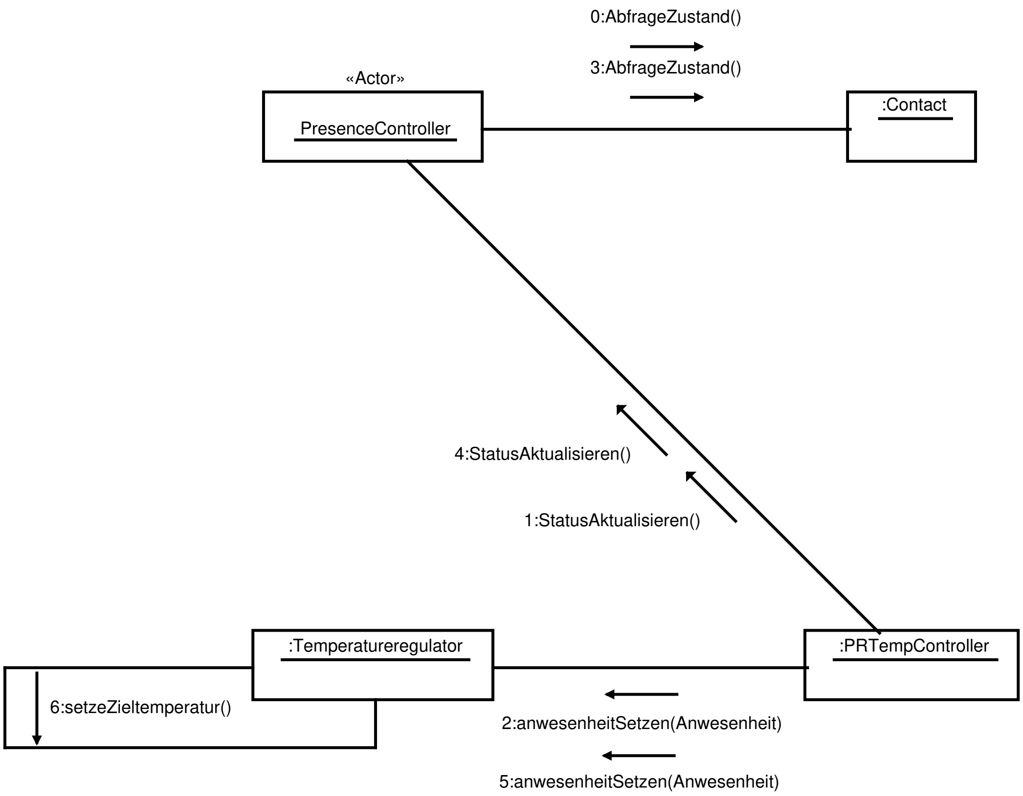
Szenario4(UmlUseCase:AnwesenheitTemp)



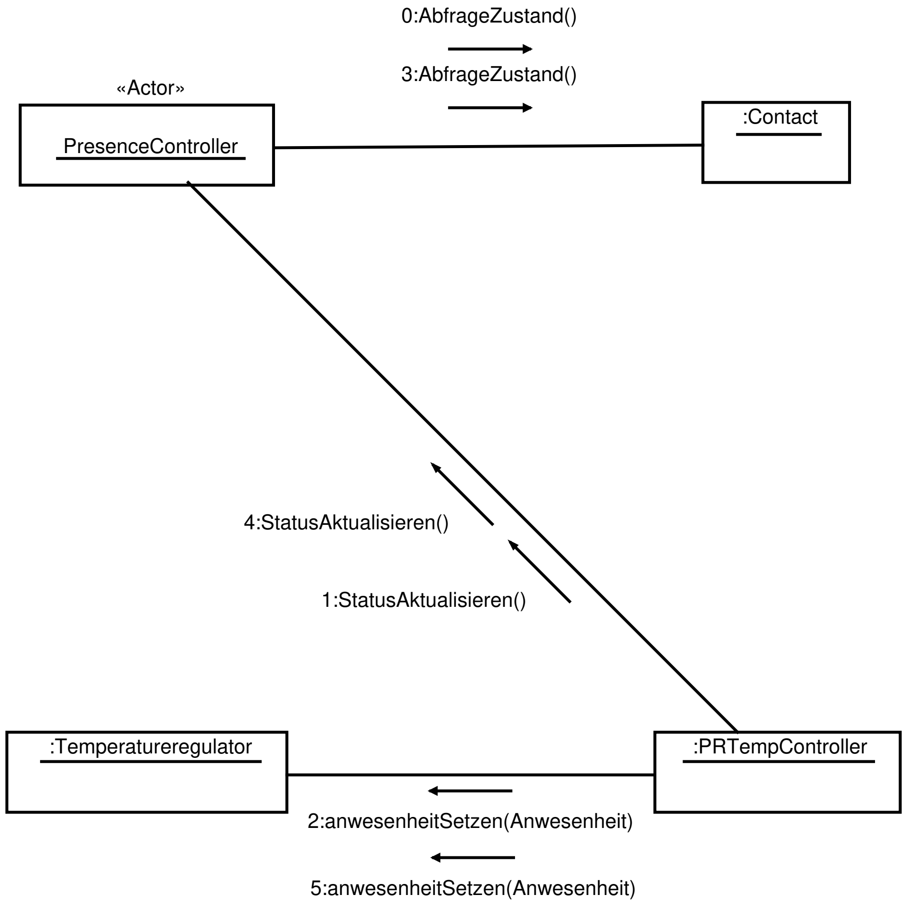
Szenario6(UmlUseCase:AnwesenheitTemp)



Szenario7(UmlUseCase:AnwesenheitTemp)

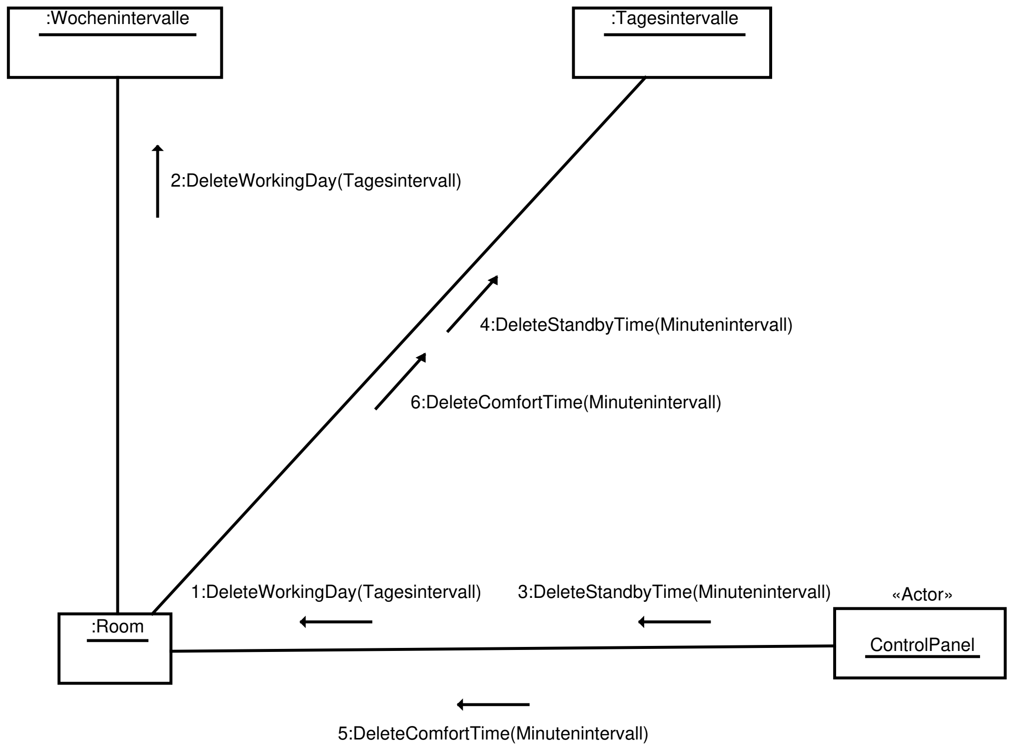


Szenario8(UmlUseCase:AnwesenheitTemp)

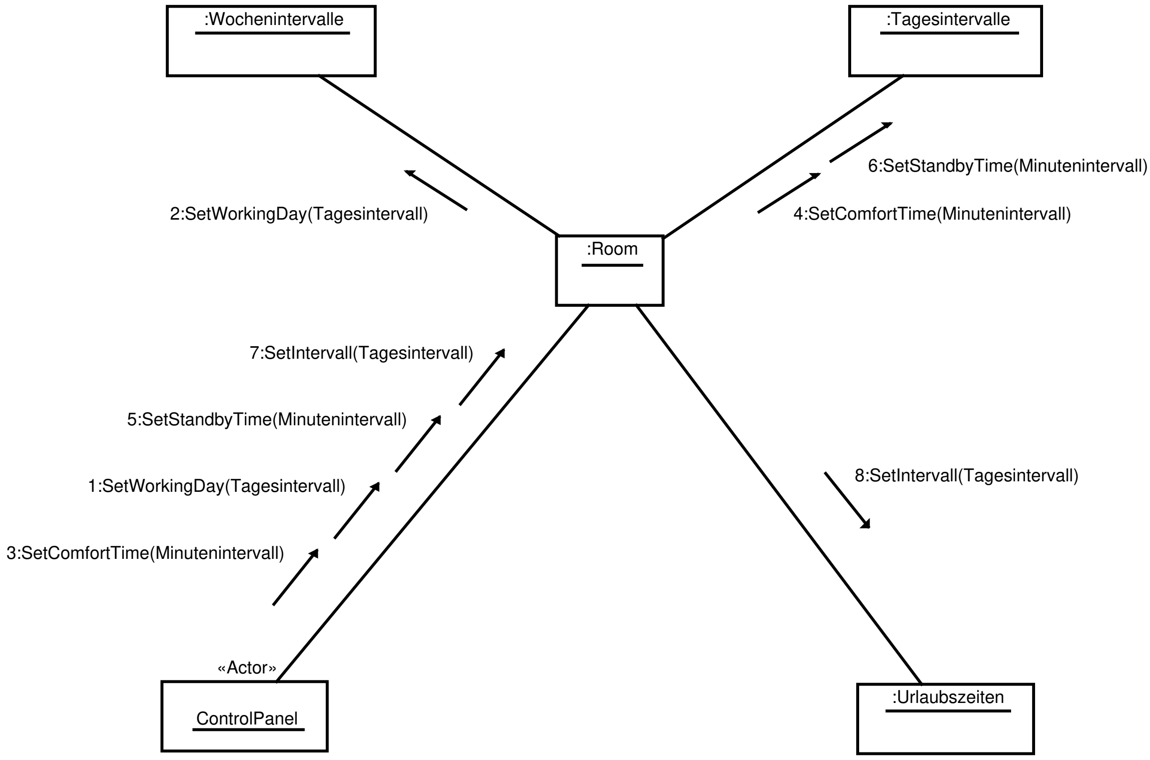


2.2.6.3 Arbeitszeitfestlegung

Szenario21(UmlUseCase:Arbeitszeitfestlegung)

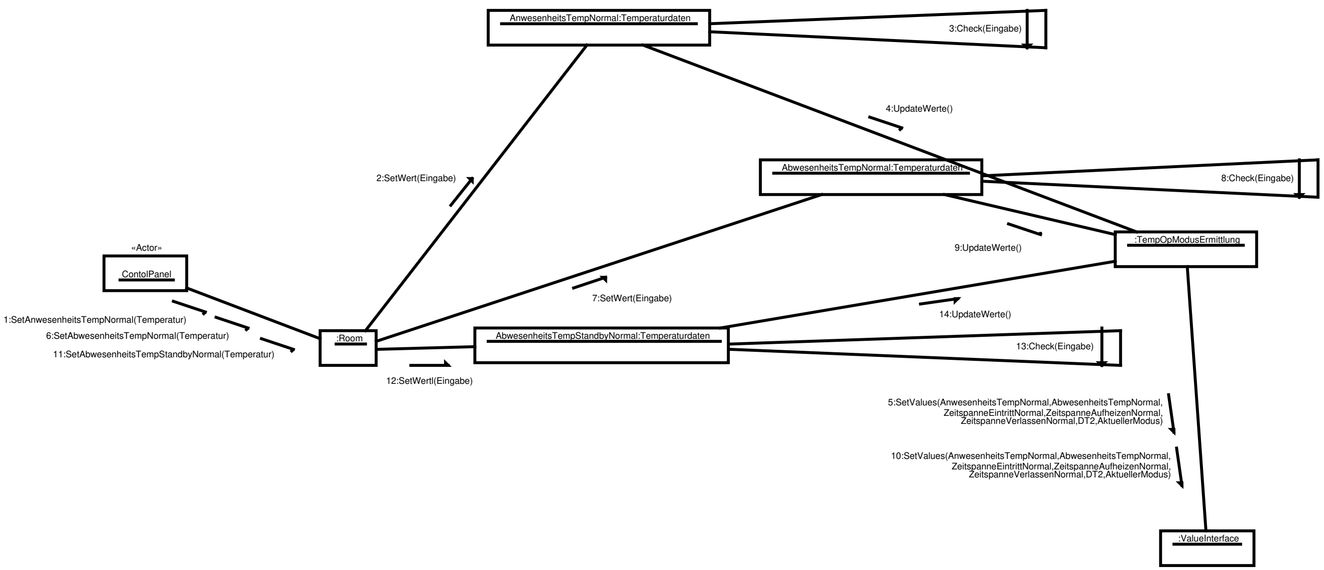


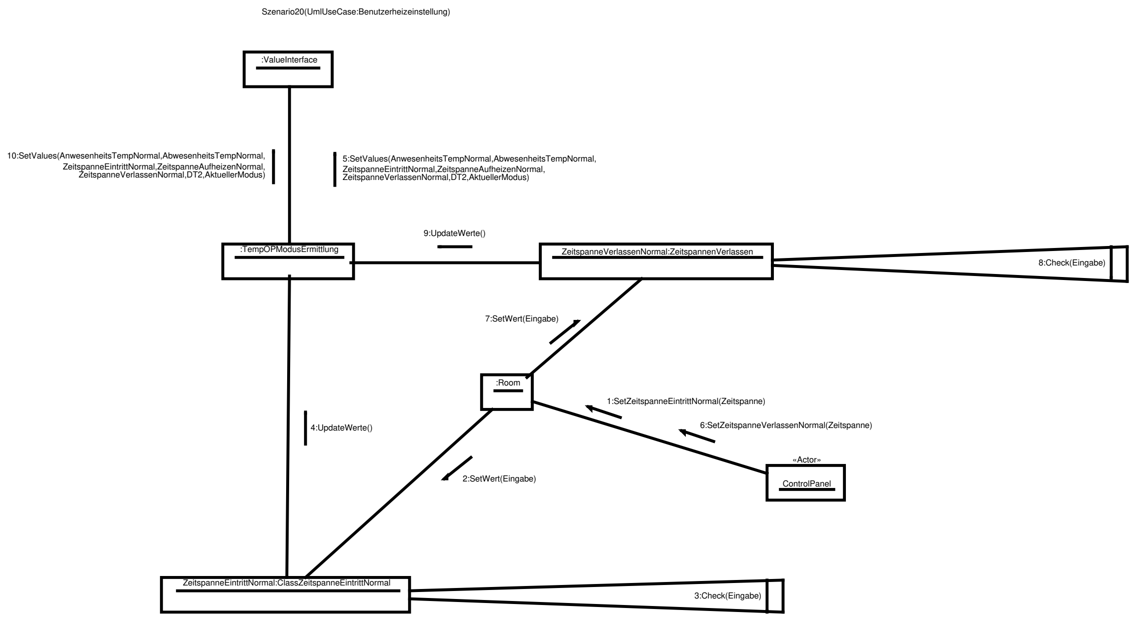
Szenario22(UmlUseCase:Arbeitszeitfestlegung)



2.2.6.4 Benutzerheizeinstellung

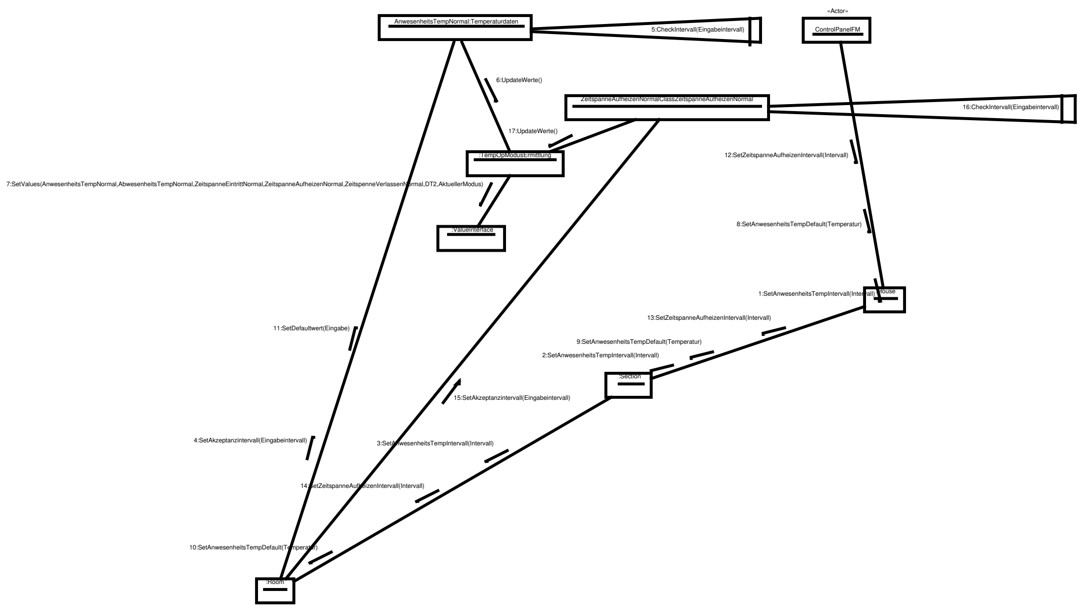
Szenario19(UmlUseCase:Benutzerheizeinstellung)




Szenario20(UmlUseCase:Benutzerheizeinstellung)
:ValueInterface
10:SetValues(AnwesenheitsTempNormal,AbwesenheitsTempNormal, ZeitspanneEintrittNormal,ZeitspanneAufheizenNormal, ZeitspanneVerlassenNormal,DT2,AktuellerModus)
5:SetValues(AnwesenheitsTempNormal,AbwesenheitsTempNormal, ZeitspanneEintrittNormal,ZeitspanneAufheizenNormal, ZeitspanneVerlassenNormal,DT2,AktuellerModus)
9:UpdateWerte()
:TempOPModusErmittlung
ZeitspanneVerlassenNormal:ZeitspannenVerlassen
8:Check(Eingabe)
7:SetWert(Eingabe)
:Room
1:SetZeitspanneEintrittNormal(Zeitspanne)
6:SetZeitspanneVerlassenNormal(Zeitspanne)
4:UpdateWerte()
«Actor»
ControlPanel
2:SetWert(Eingabe)
ZeitspanneEintrittNormal:ClassZeitspanneEintrittNormal
3:Check(Eingabe)

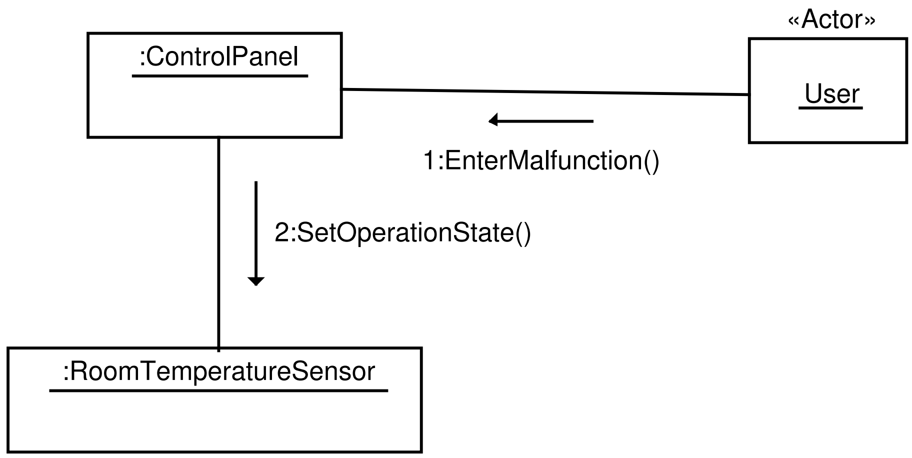
2.2.6.5 DefaultundAkzeptanzwertfestlegung

Szenario23(UmlUseCase:DefaultundAkzeptanzwertfestlegung)

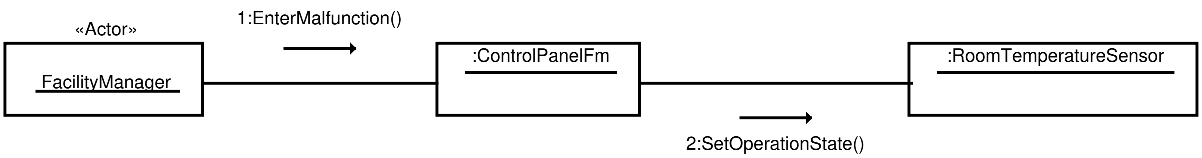


2.2.6.6 EnterMalfunction

Szenario48(UmlUseCase:EnterMalfunction)

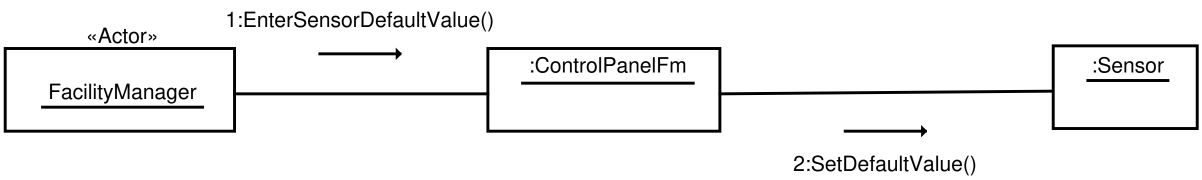


Szenario50(UmlUseCase:EnterMalfunction)

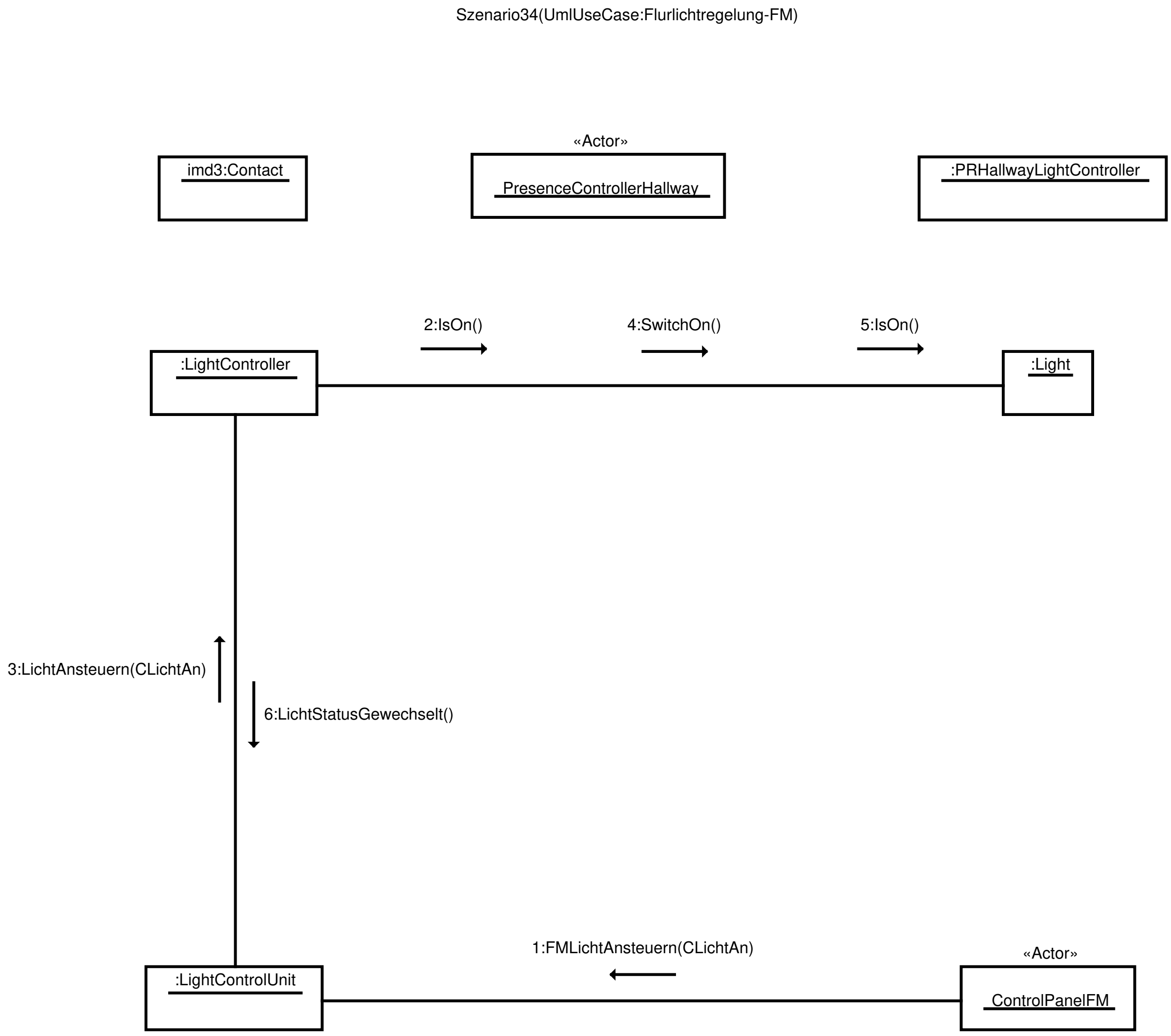


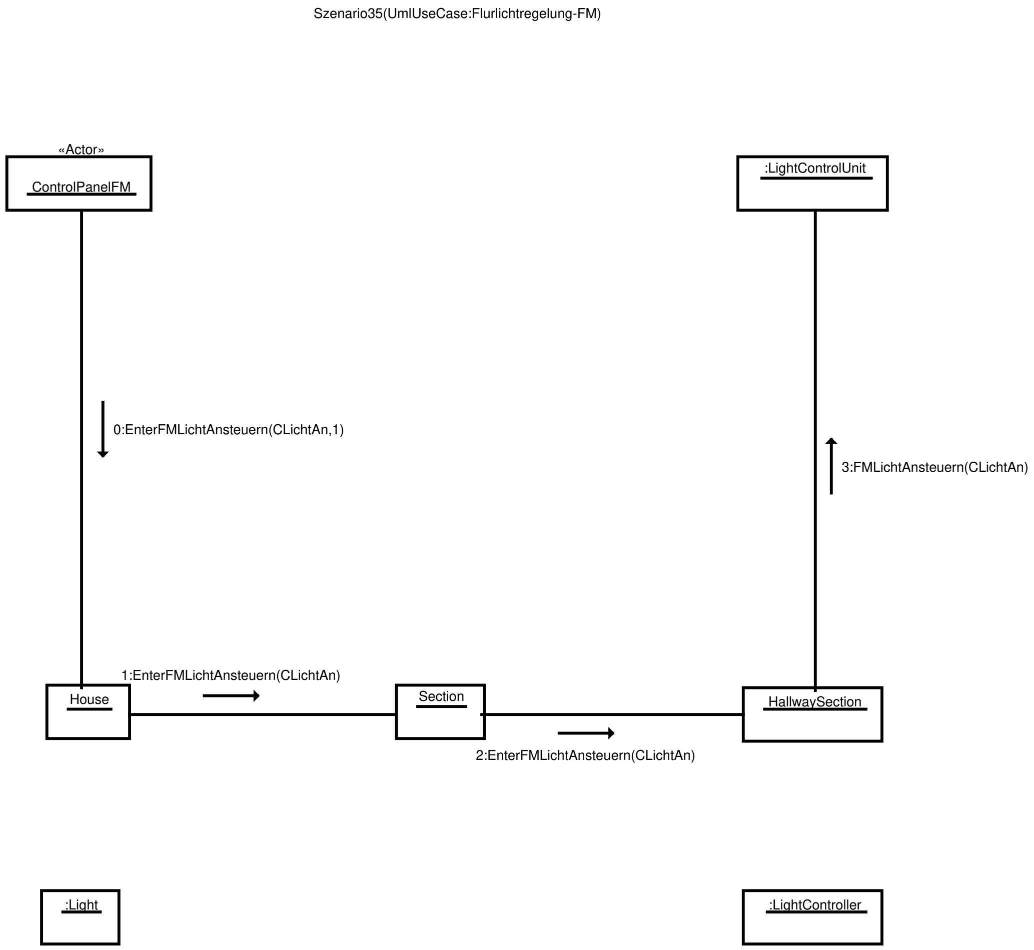
2.2.6.7 EnterSensorDefaultValue

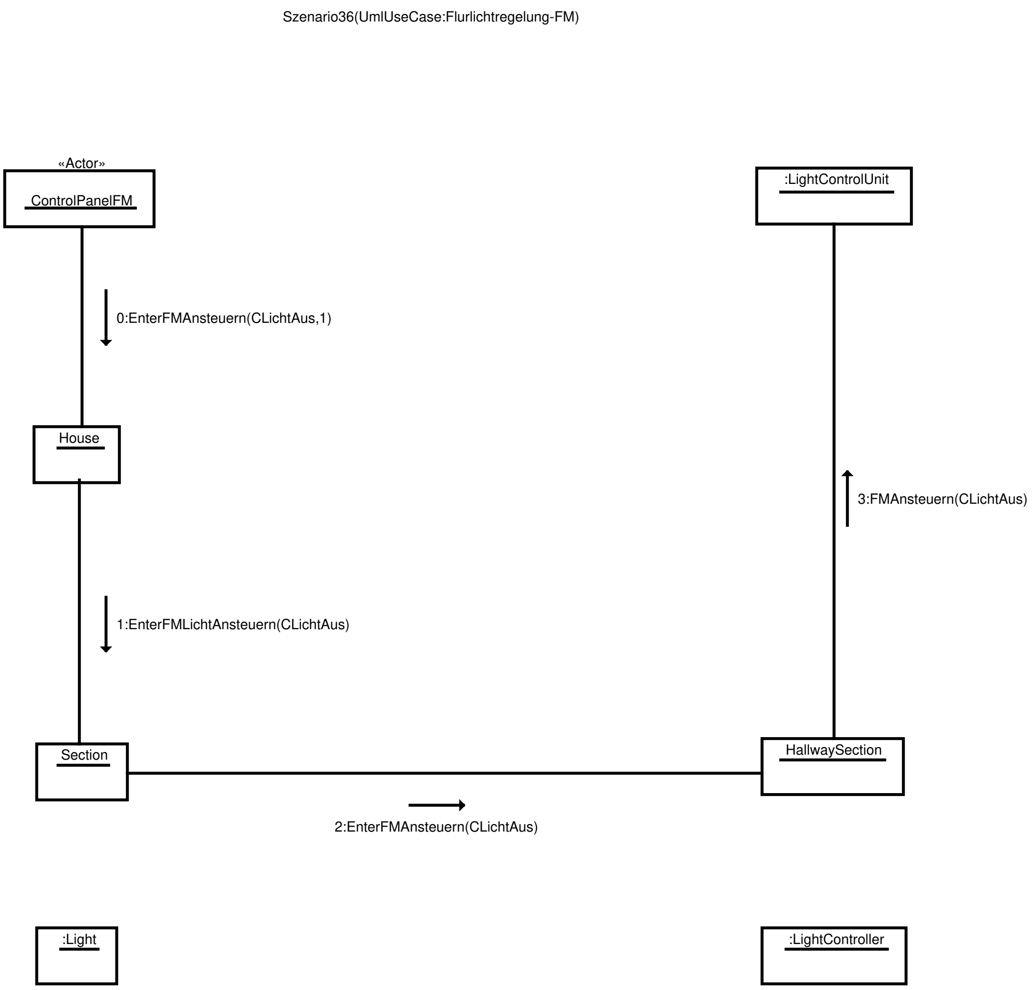
Szenario52(UmlUseCase:EnterSensorDefaultValue)



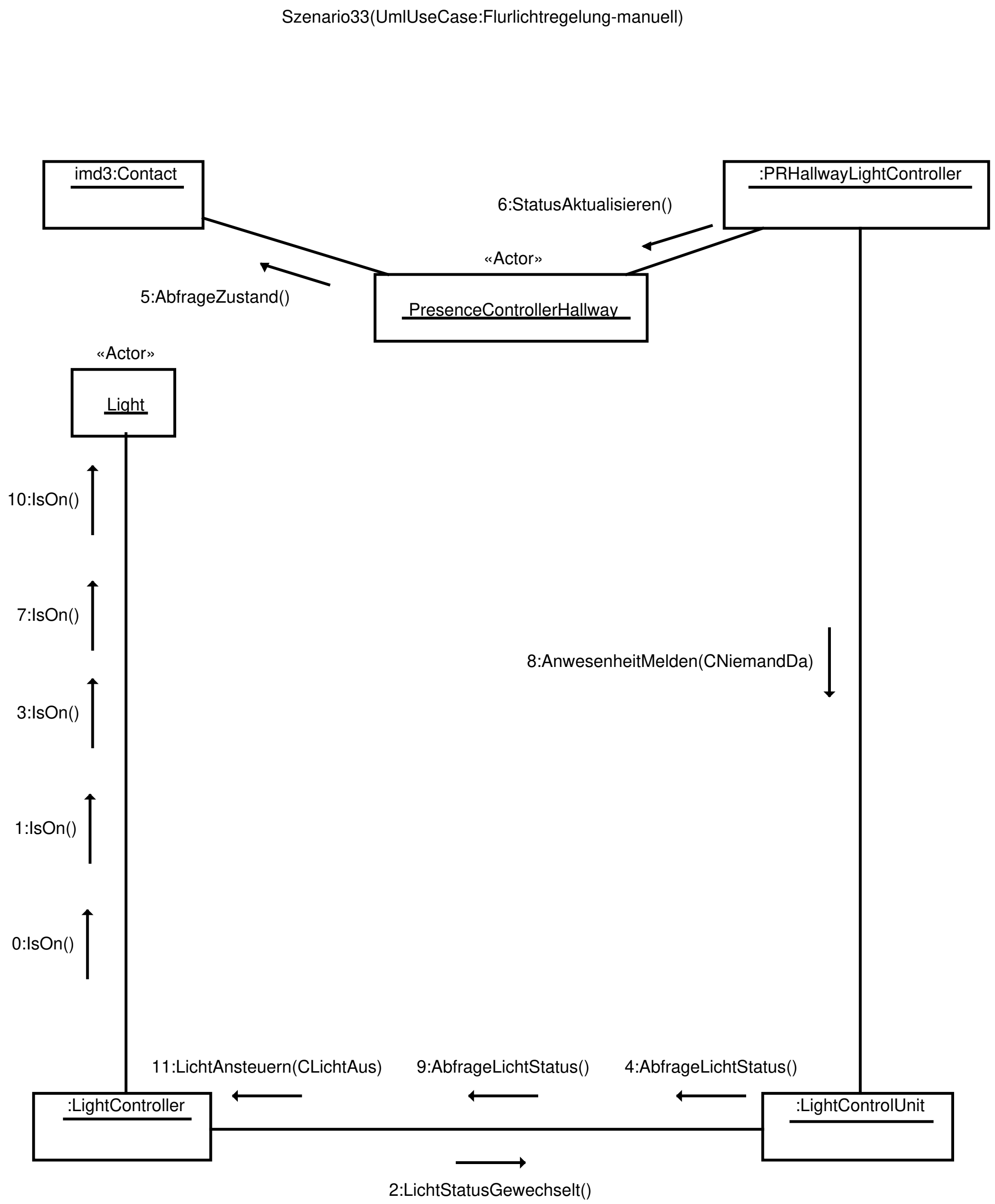
2.2.6.8 Flurlichtregelung-FM



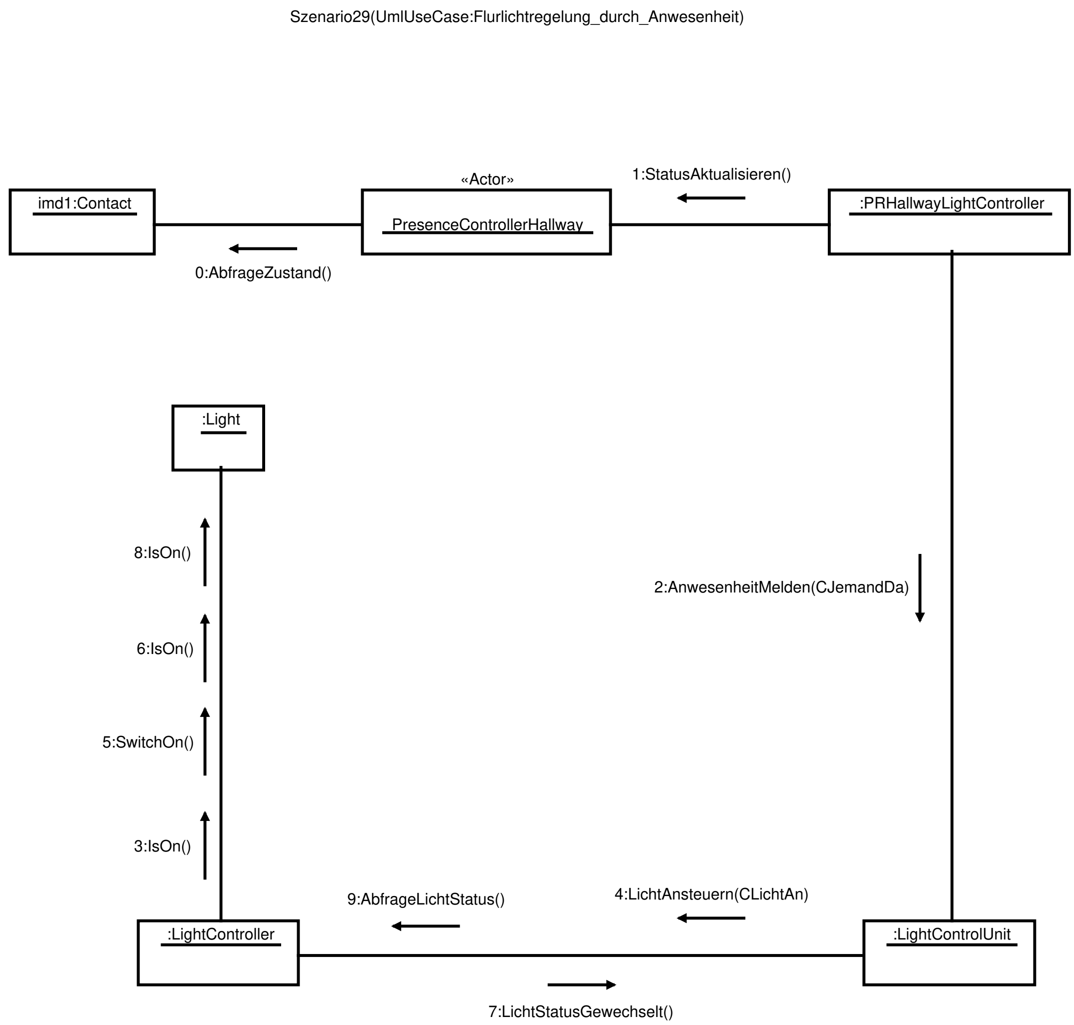

Szenario35(UmlUseCase:Flurlichtregelung-FM)
«Actor»
ControlPanelFM
:LightControlUnit
0:EnterFMLichtAnsteuern(CLichtAn,1)
3:FMLichtAnsteuern(CLichtAn)
1:EnterFMLichtAnsteuern(CLichtAn)
House
Section
HallwaySection
2:EnterFMLichtAnsteuern(CLichtAn)
:Light
:LightController


Szenario36(UmlUseCase:Flurlichtregelung-FM)
«Actor»
ControlPanelFM
:LightControlUnit
0:EnterFMAnsteuern(CLichtAus,1)
House
3:FMAnsteuern(CLichtAus)
1:EnterFMLichtAnsteuern(CLichtAus)
Section
HallwaySection
2:EnterFMAnsteuern(CLichtAus)
:Light
:LightController

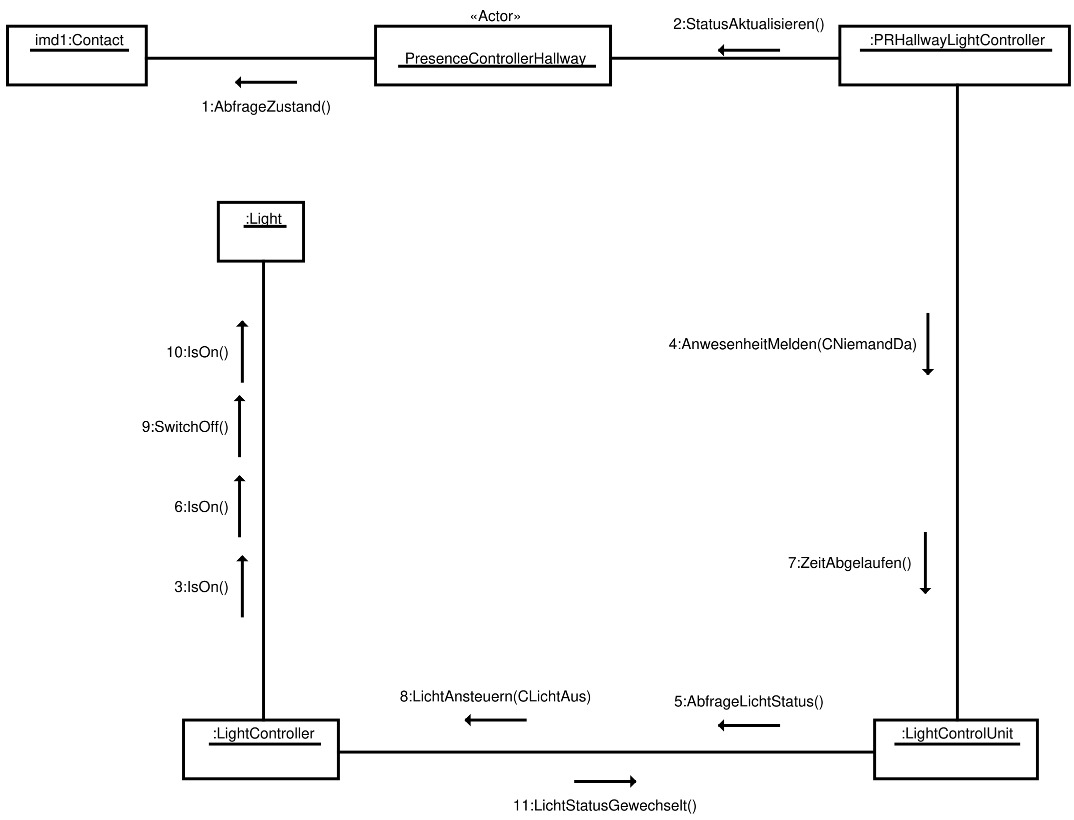
2.2.6.9 Flurlichtregelung-manuell



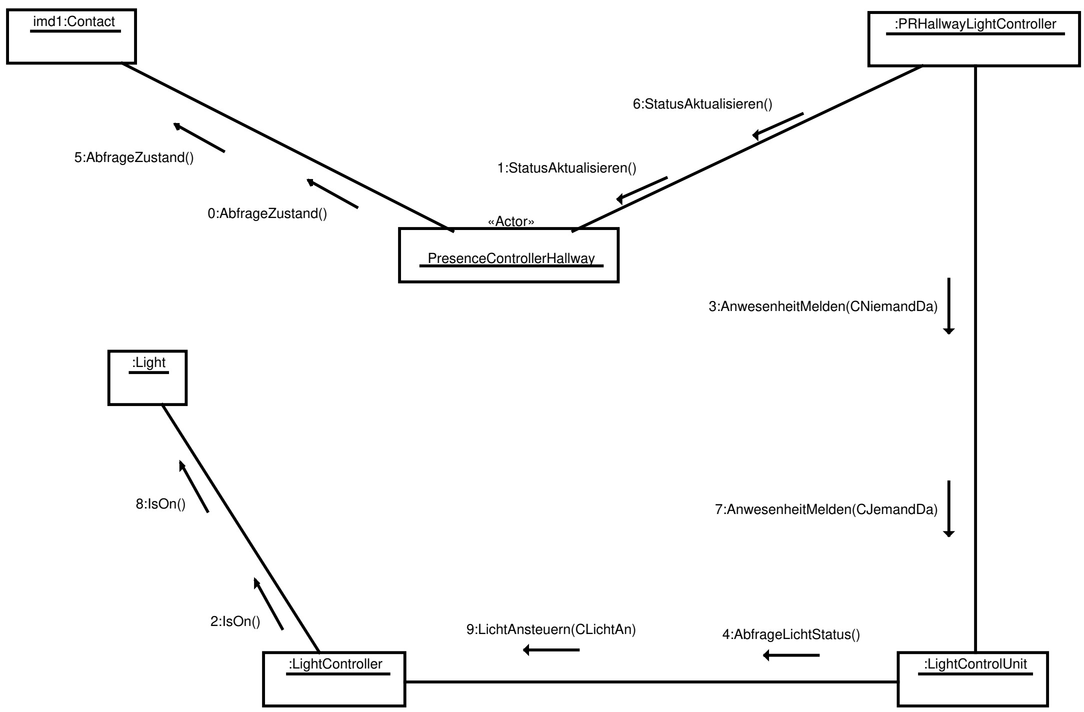
2.2.6.10 Flurlichtregelung_durch_Anwesenheit



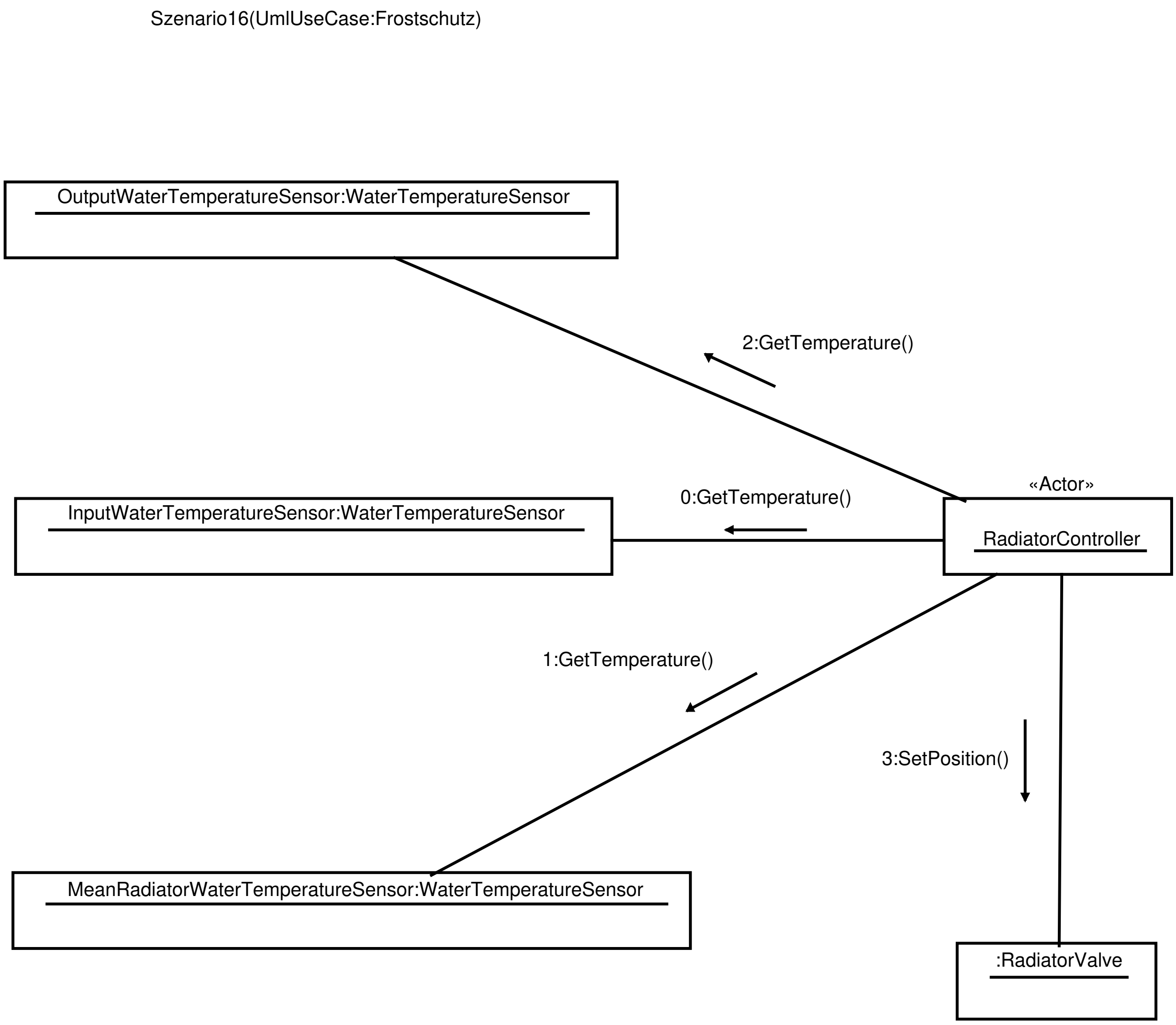
Szenario30(UmlUseCase:Flurlichtregelung_durch_Anwesenheit)



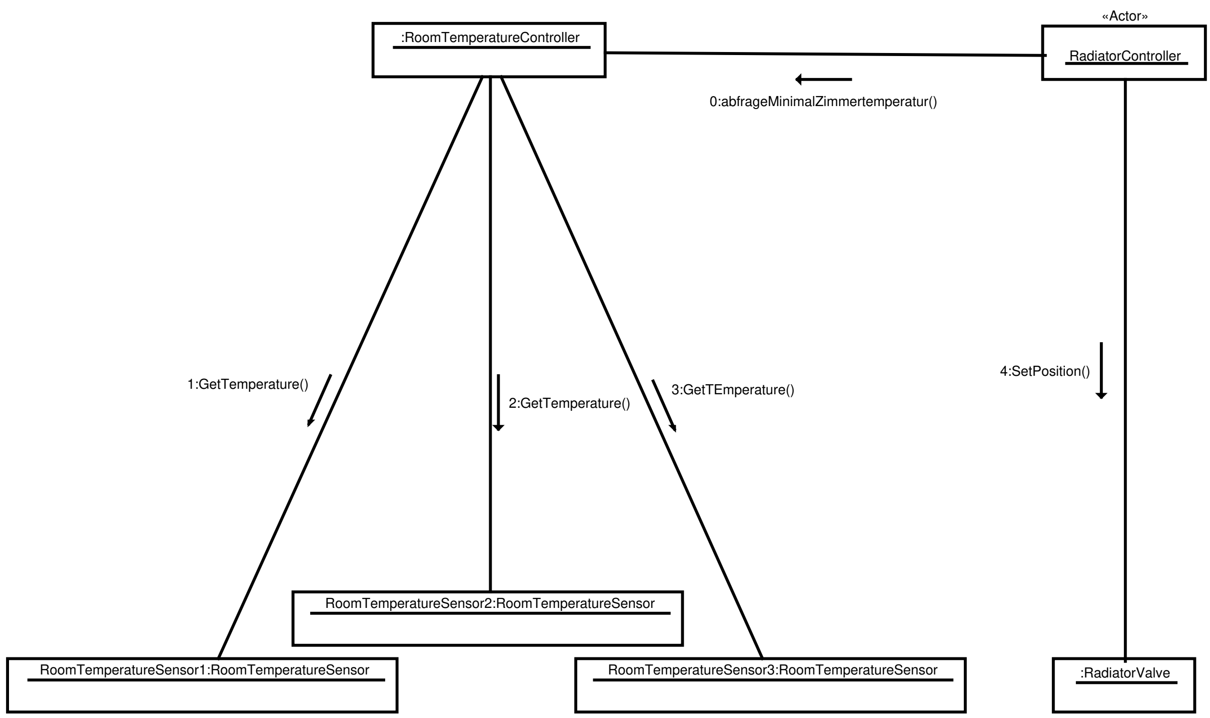
Szenario31(UmlUseCase:Flurlichtregelung_durch_Anwesenheit)



2.2.6.11 Frostschutz

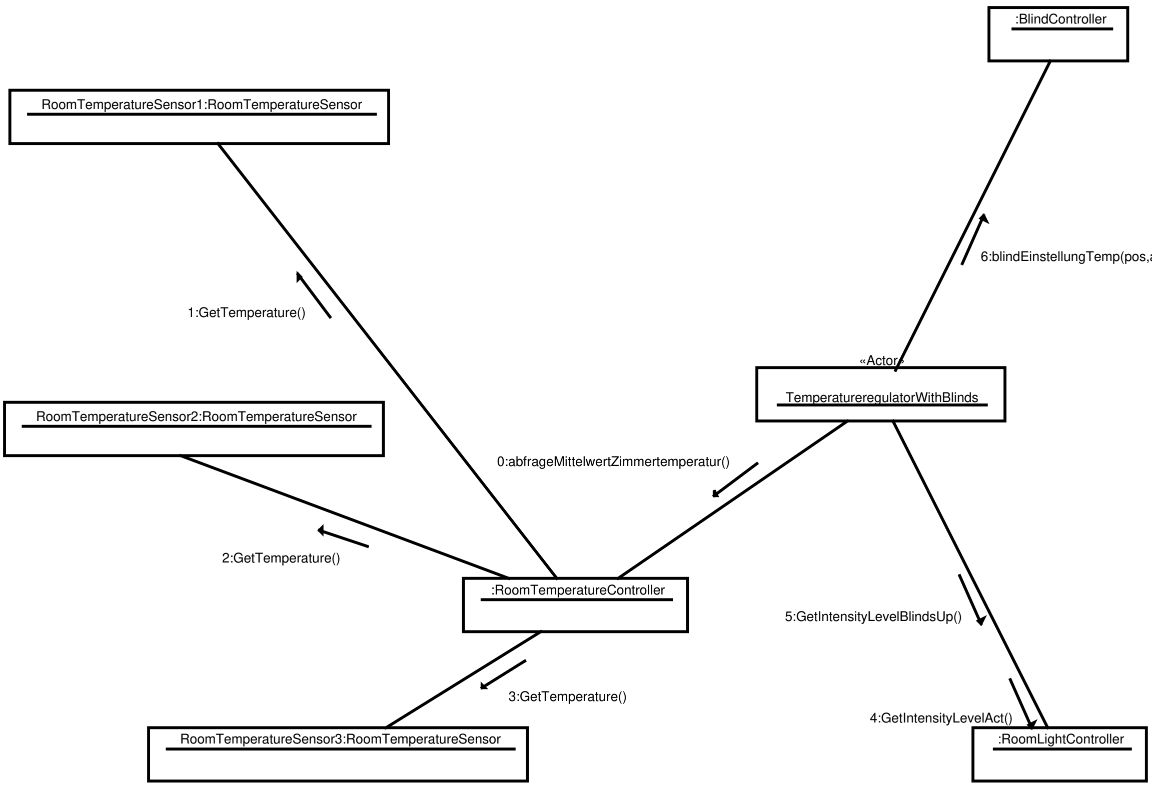


Szenario17(UmlUseCase:Frostschutz)

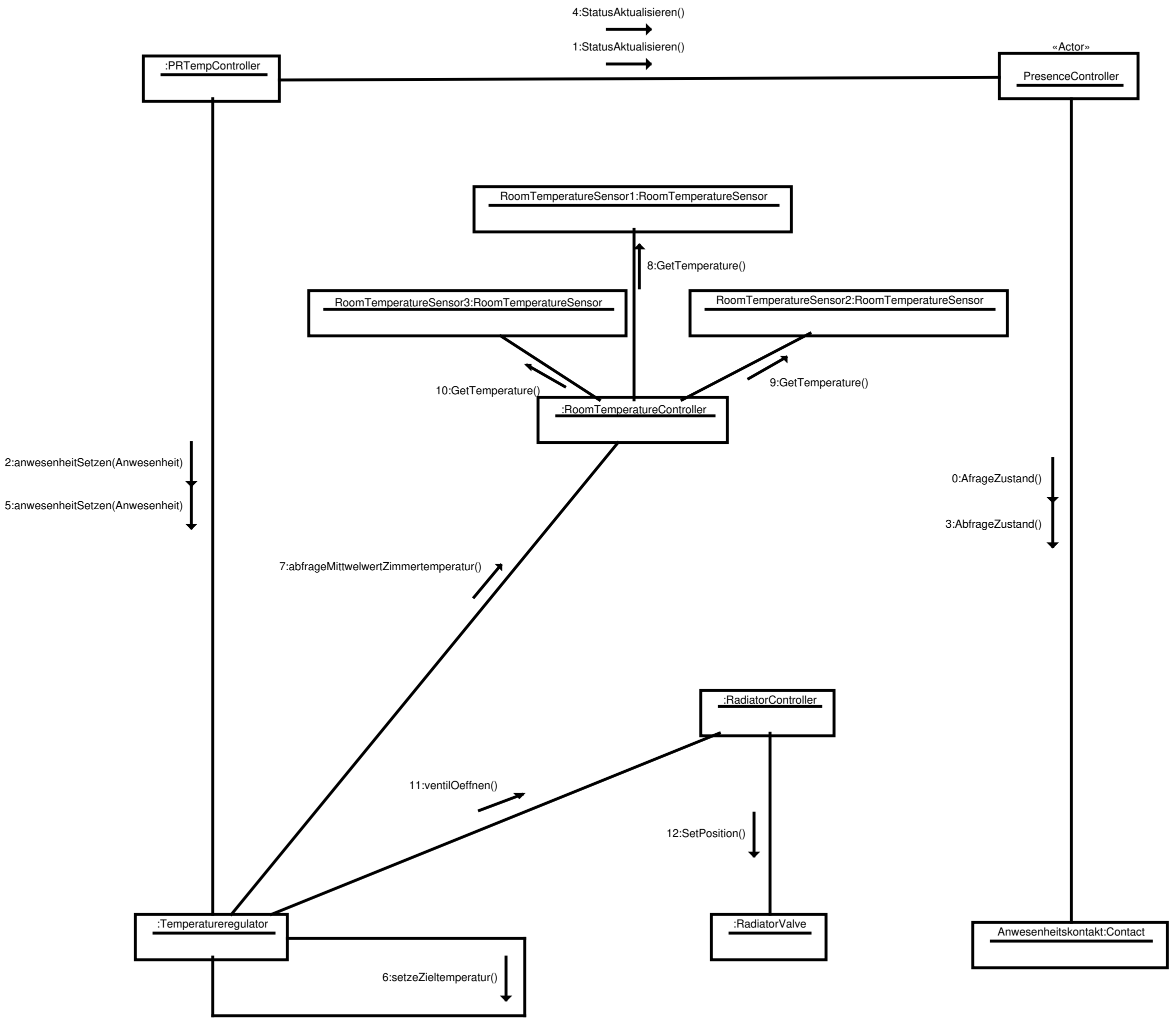


2.2.6.12 Heizen

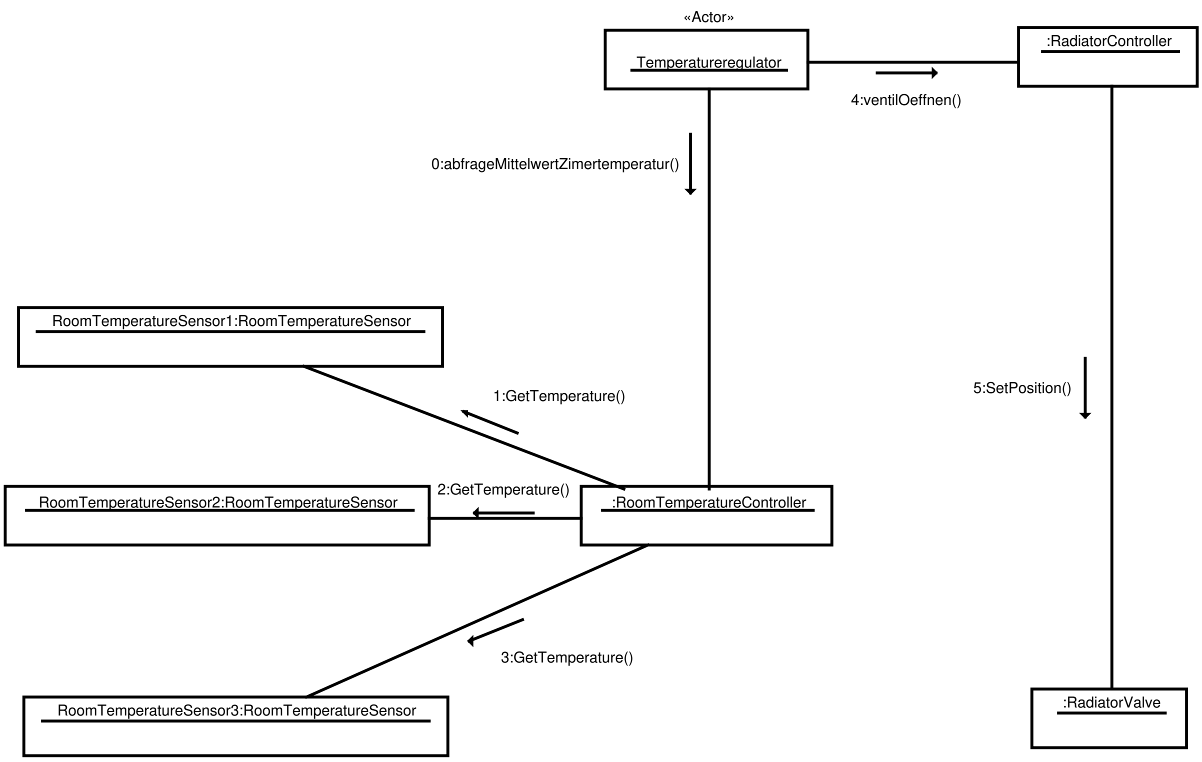
Szenario11(UmlUseCase:Heizen)



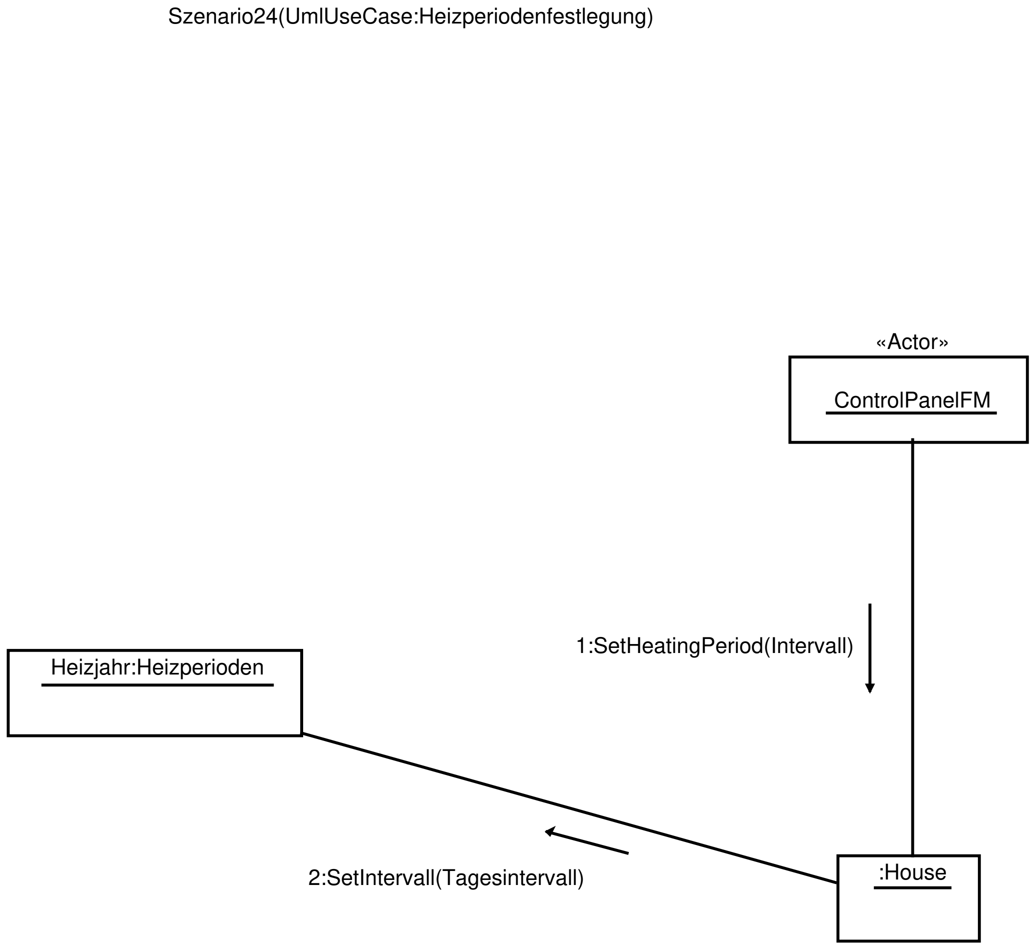
Szenario12(UmlUseCase:Heizen)



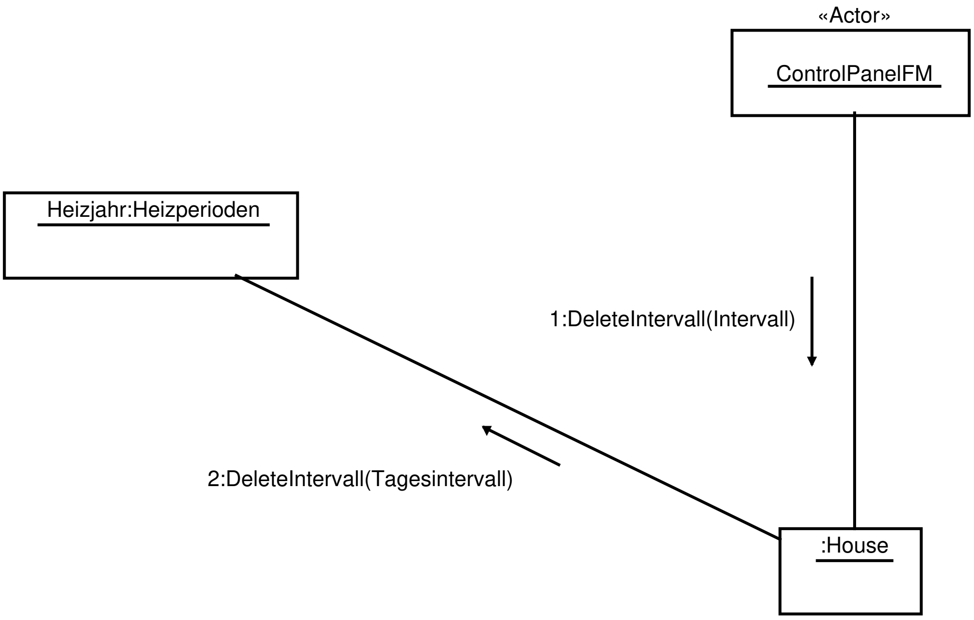
Szenario13(UmlUseCase:Heizen)



2.2.6.13 Heizperiodenfestlegung

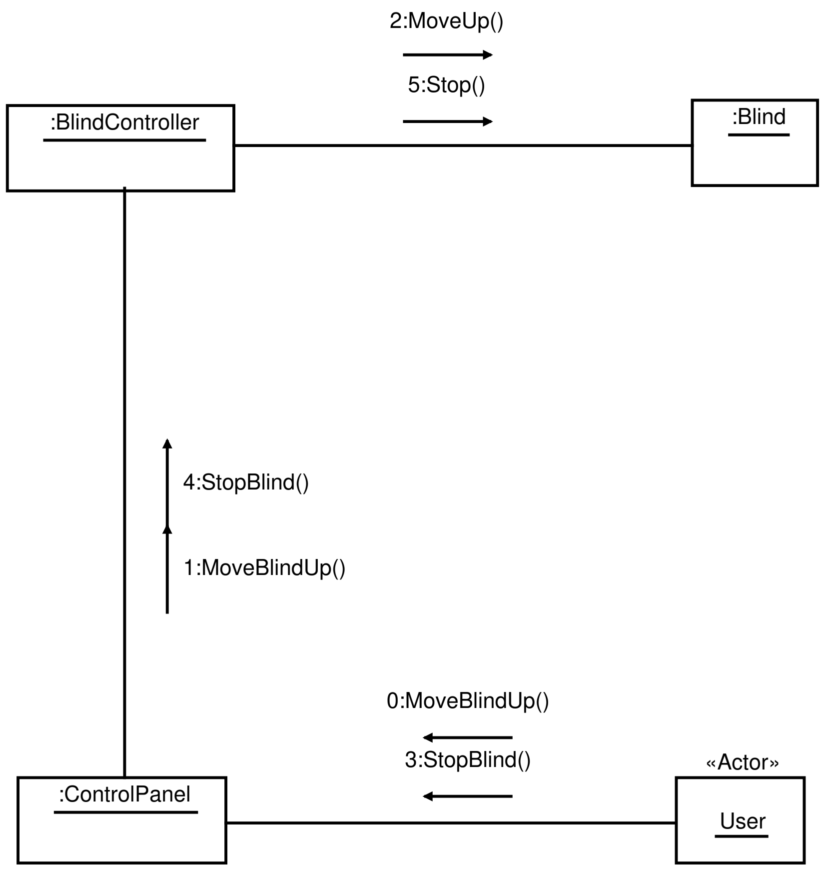


Szenario25(UmlUseCase:Heizperiodenfestlegung)

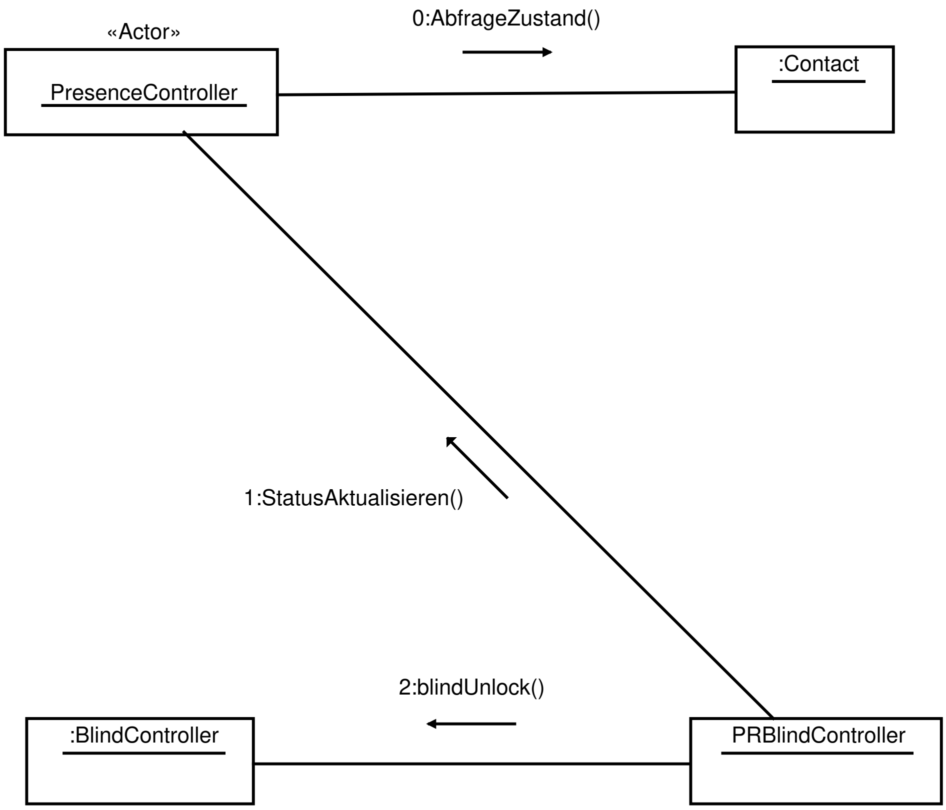


2.2.6.14 JalousienKontrolle

Szenario14(UmlUseCase:JalousienKontrolle)

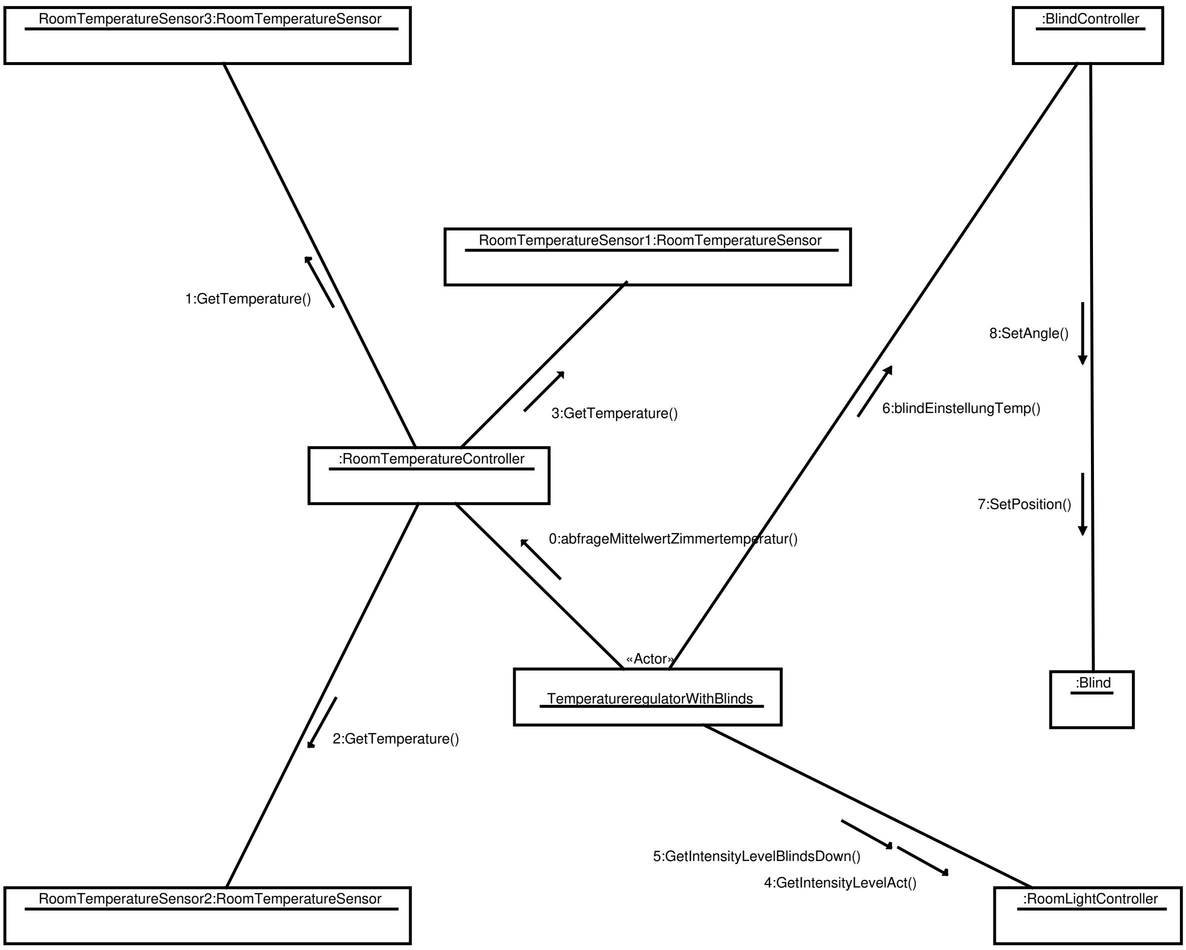


Szenario15(UmlUseCase:JalousienKontrolle)

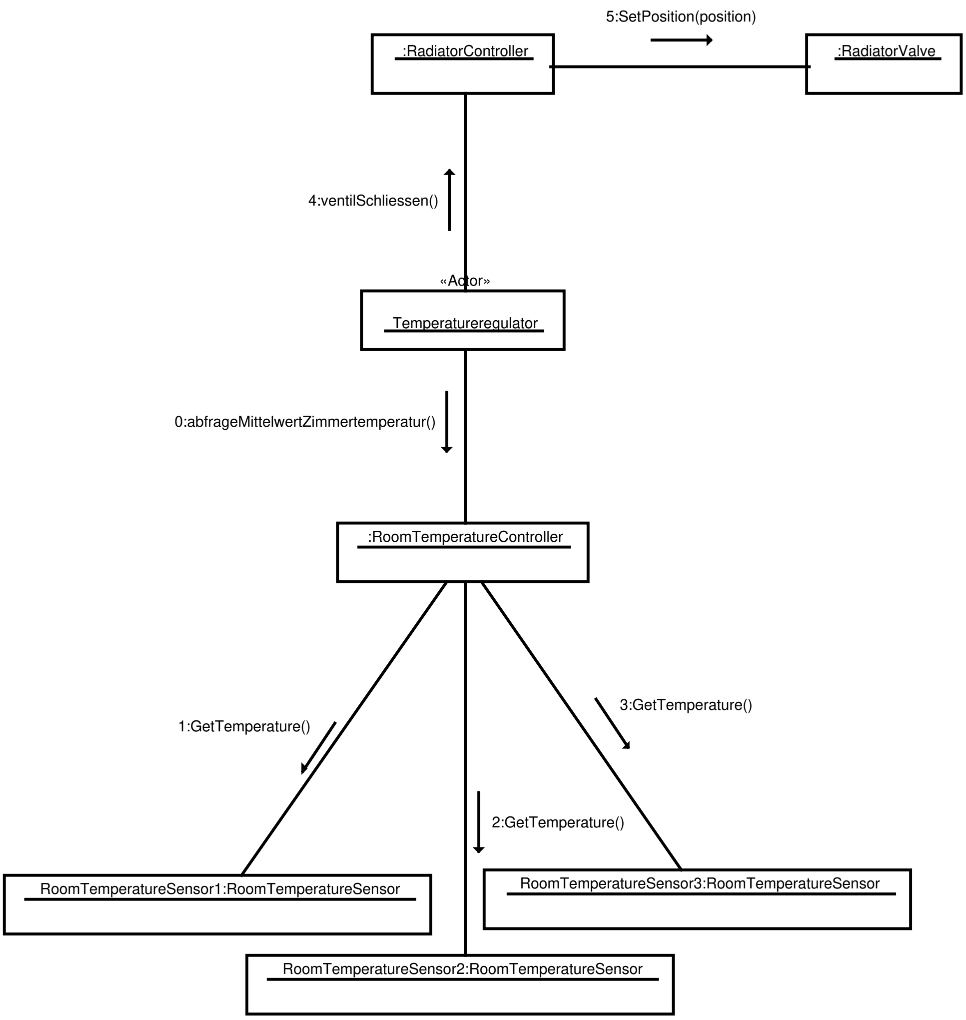


2.2.6.15 Kuehlen

Szenario9(UmlUseCase:Kuehlen)

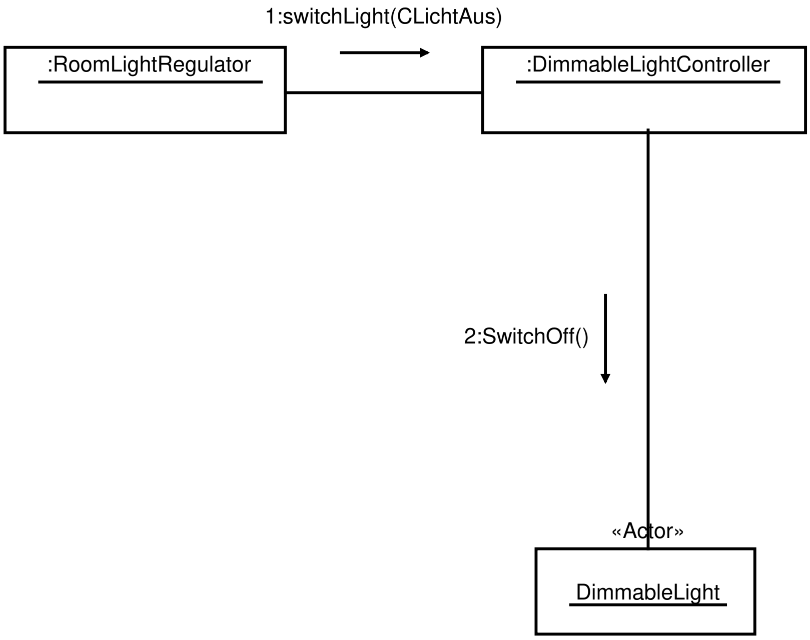


Szenario10und18(UmlUseCase:Kuehlen)

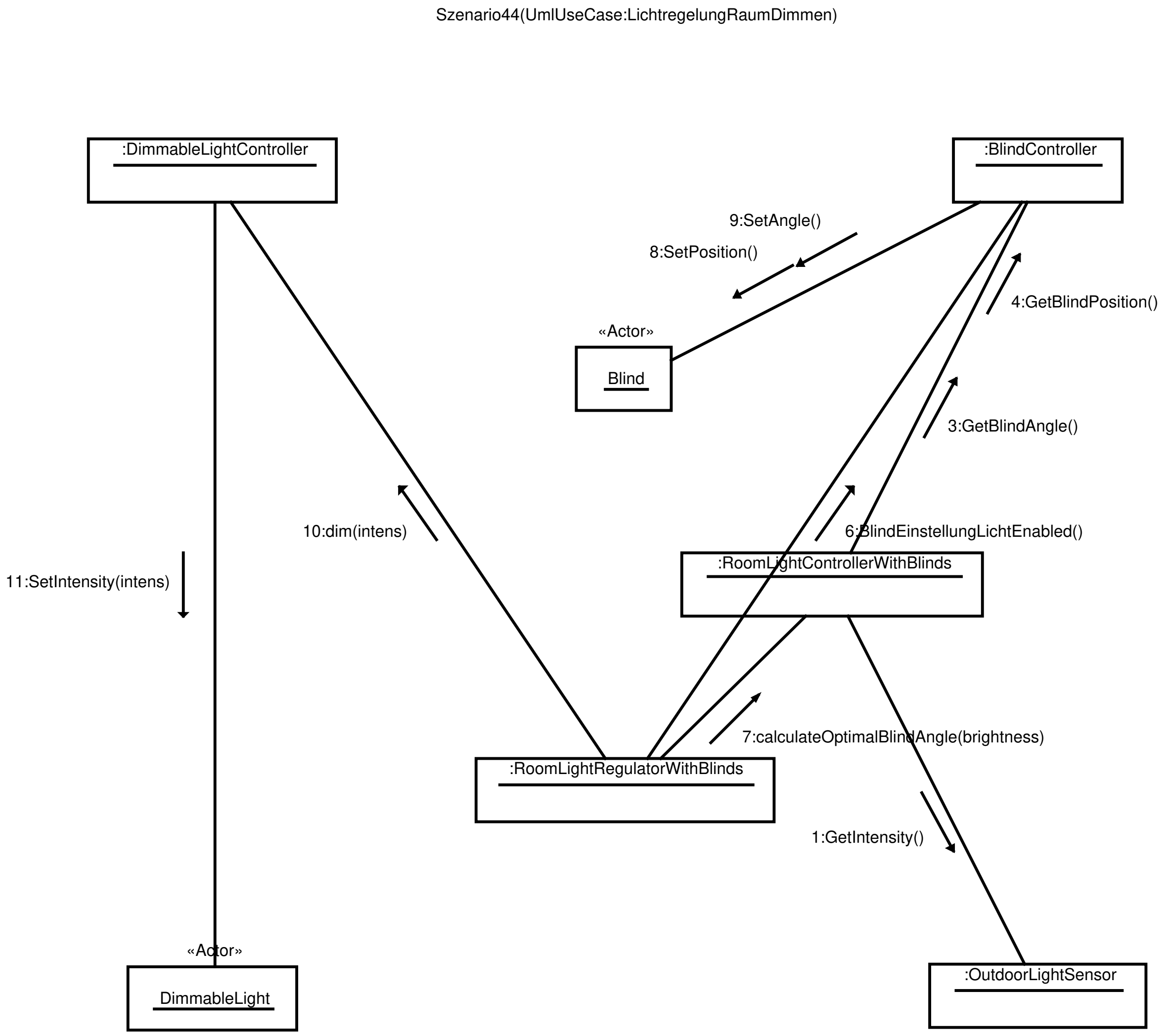


2.2.6.16 LichtregelungRaumAus

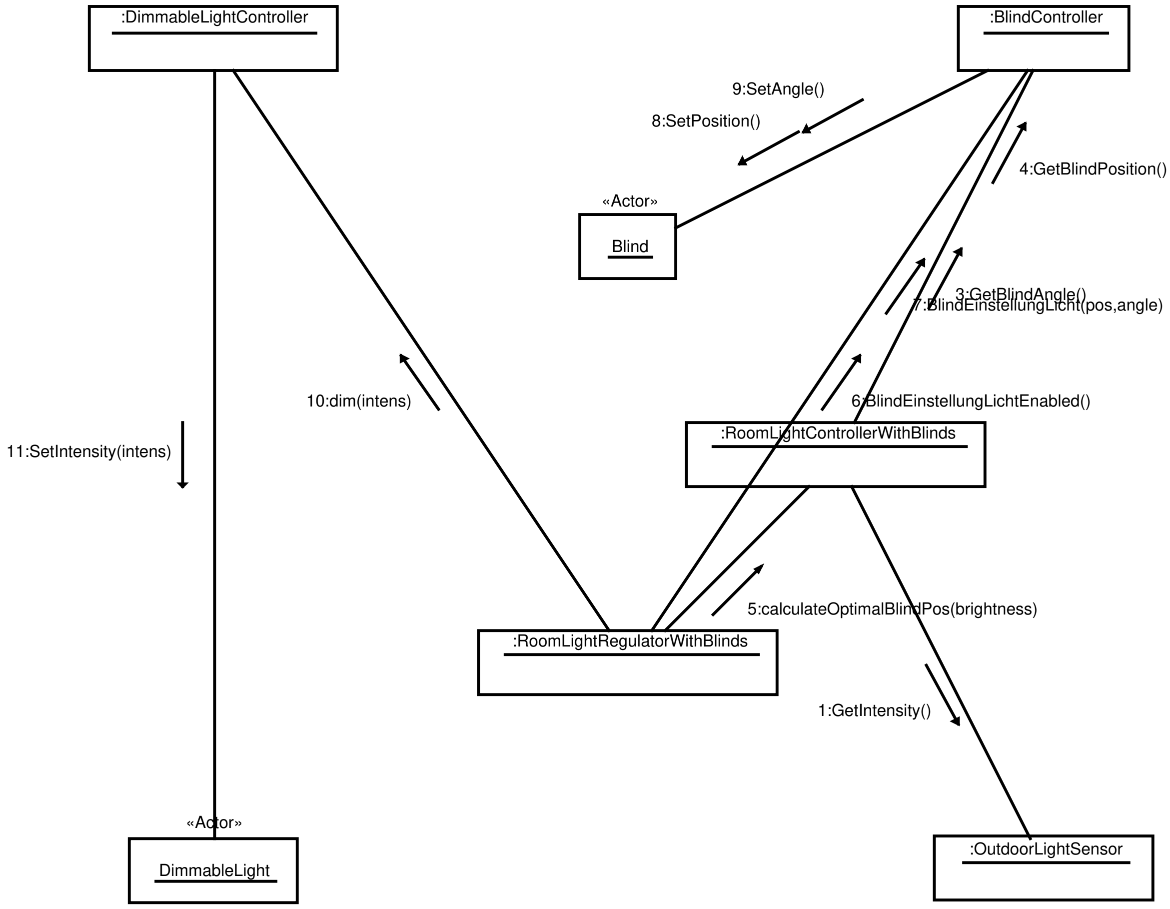
Szenario47(UmlUseCase:LichtregelungRaumAus)



2.2.6.17 LichtregelungRaumDimmen

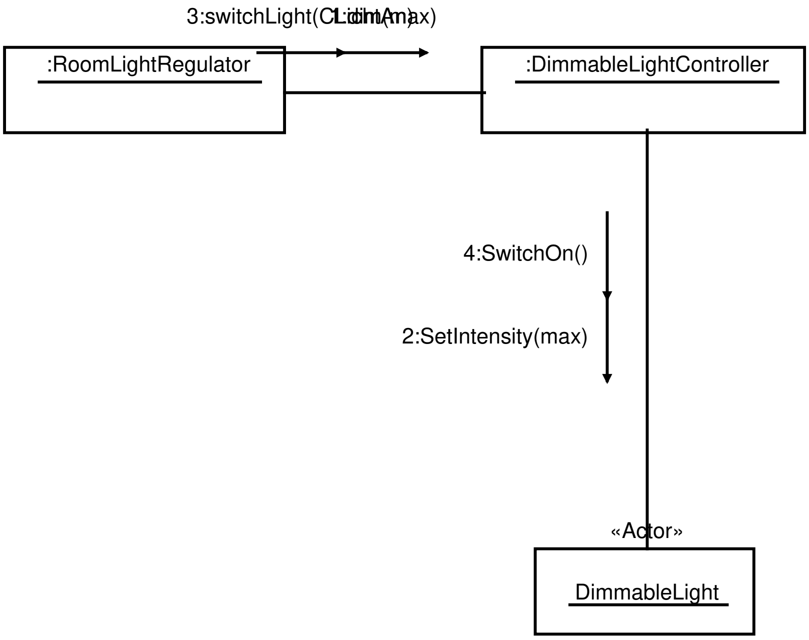


Szenario45(UmlUseCase:LichtregelungRaumDimmen)



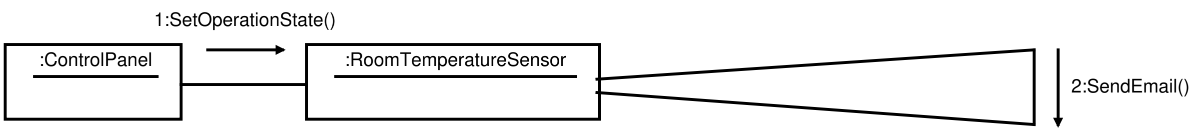
2.2.6.18 LichtregelungRaumEin

Szenario46(UmlUseCase:LichtregelungRaumEin)

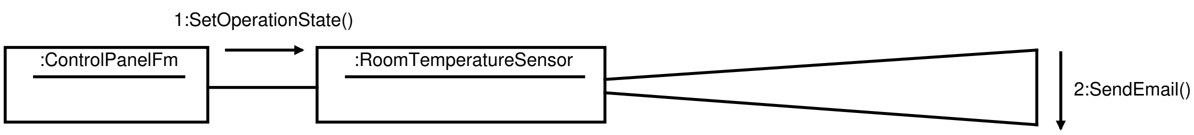


2.2.6.19 SendEMail

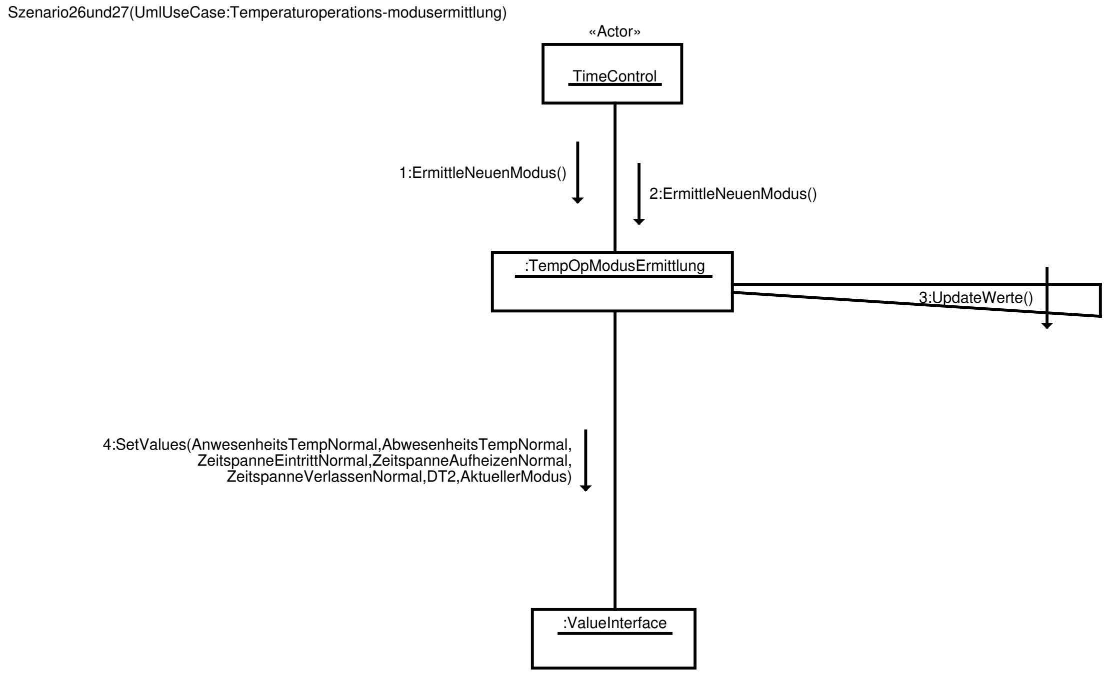
Szenario49(UmlUseCase:SendEMail)



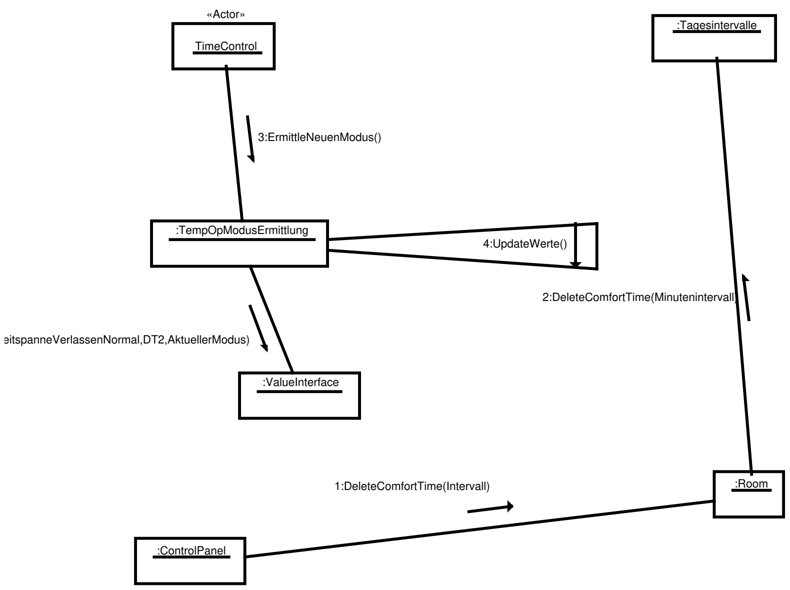
Szenario51(UmlUseCase:SendEMail)



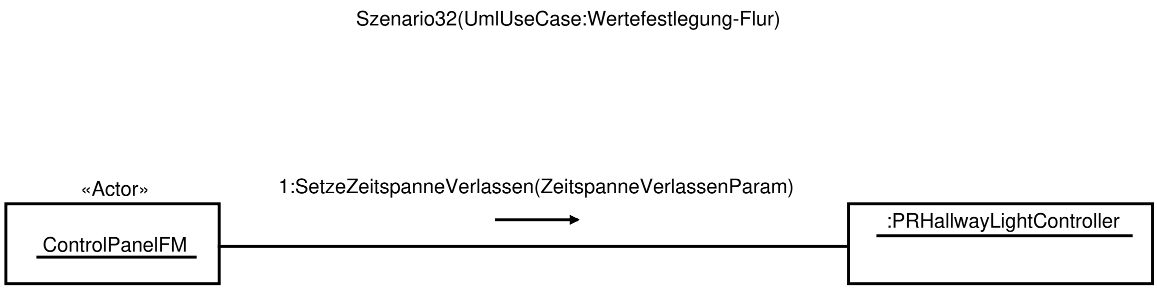
2.2.6.20 Temperaturoperations-modusermittlung



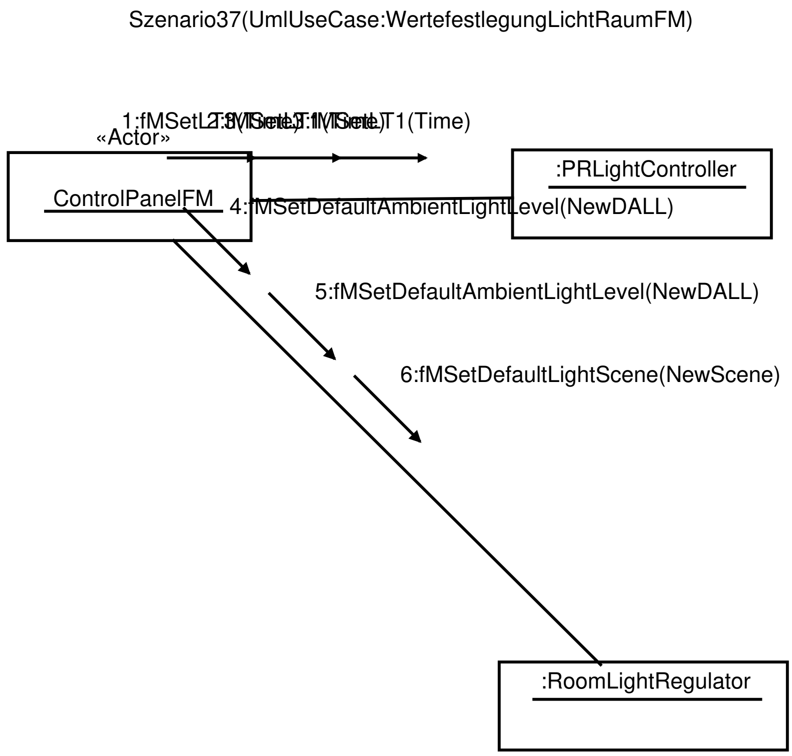
Szenario28(UmlUseCase:Temperaturoperations-modusermittlung)



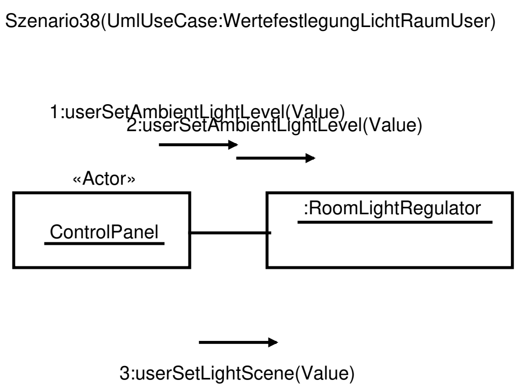
2.2.6.21 Wertefestlegung-Flur



2.2.6.22 WertefestlegungLichtRaumFM



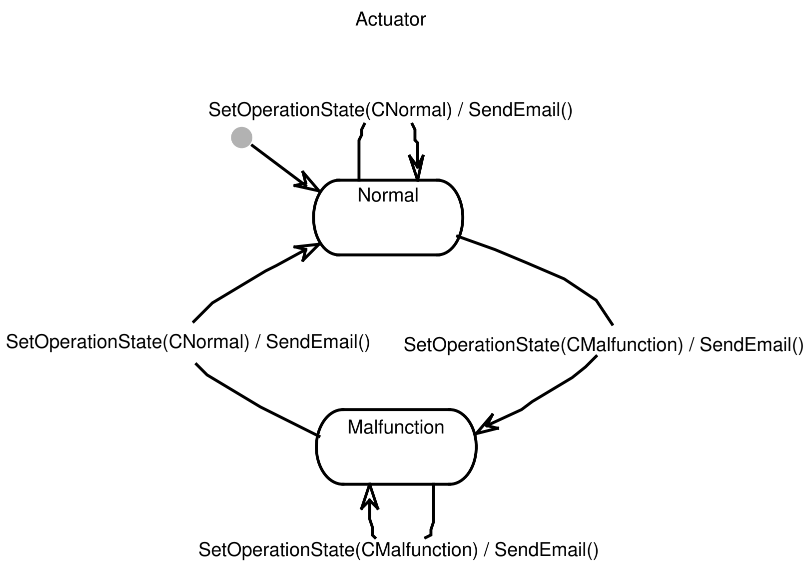
2.2.6.23 WertefestlegungLichtRaumUser



2.2.7 UML-Statediagramme

2.2.7.1 Actuator

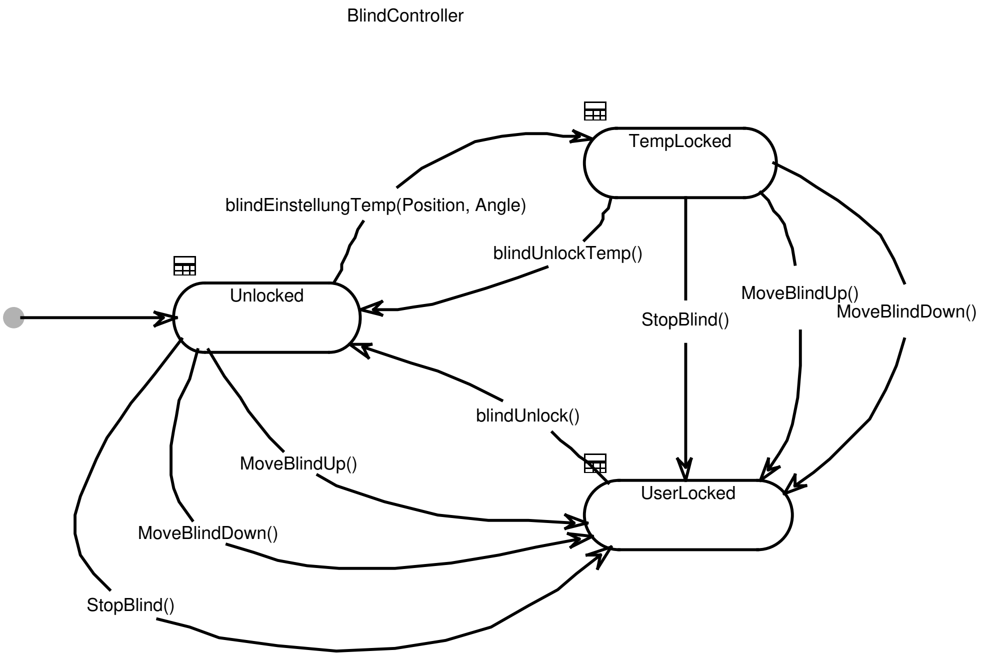
Statediagramm für die Klasse Actuator, welches stellvertretend für alle abgeleiteten Klassen die Uebergaenge zwischen den Operationszustaeenden darstellt. Diese Klasse(n) kann die Zustände "Normal" und "Malfunction", welche Normalbetrieb bzw. vom User oder Facility Manager eingegebenen "Defektbetrieb" darstellen. Die Zustände werden durch Aufruf der Methoden SetOperationState(CNormal) bzw. SetOperationState(CMalfunction) gewechselt.



2.2.7.2 Blind

2.2.7.3 BlindController

Der BlindController hat drei verschiedene Lockstati. Diese geben an, welche der drei Zugriffsmoeglichkeiten auf die Blinds (Licht, Temperatur, Anwesender) erlaubt ist.

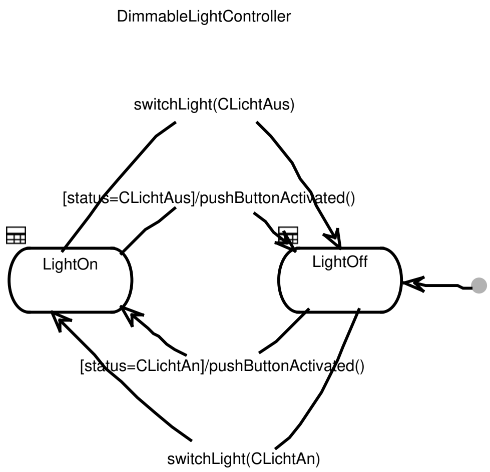


Entry-Aktion für State: Unlocked
lockStatus = CUnlocked

Entry-Aktion für State: TempLocked
statusLocked = CTempLocked

Entry-Aktion für State: UserLocked
lockStatus = CUserLocked

2.2.7.4 DimmableLightController



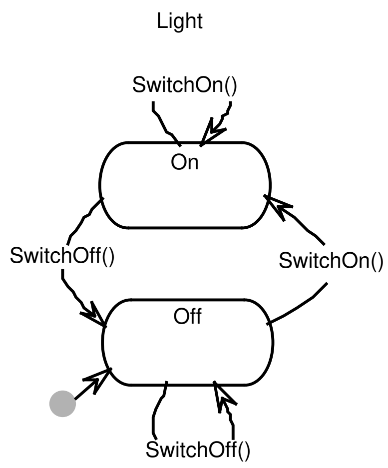
Do-Aktivität für State: LightOn
DimmableLight.IsOn()

Do-Aktivität für State: LightOff

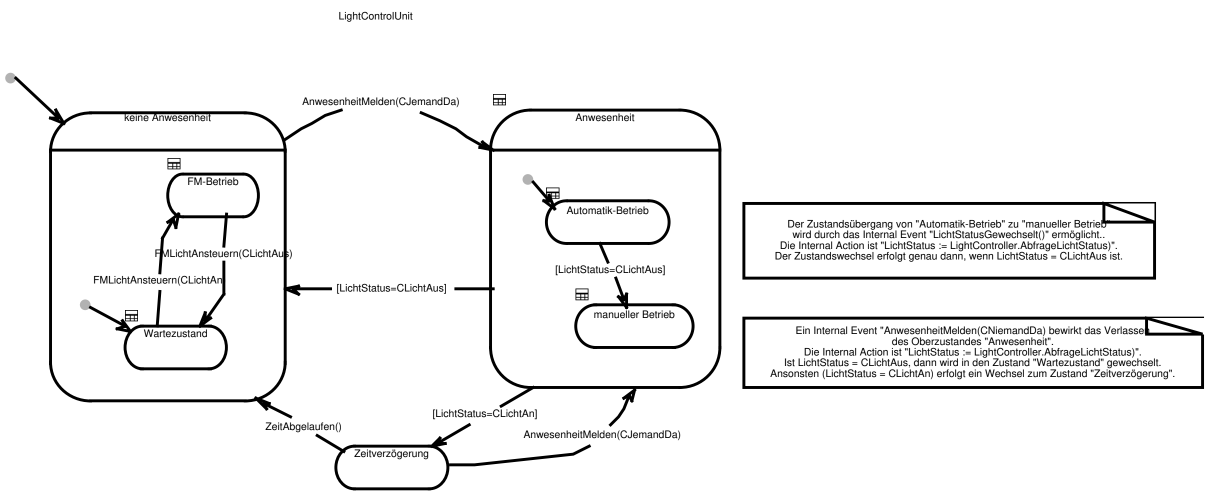
DimmableLight.IsOn()

2.2.7.5 Light

Statediagramm der Klasse Light. Diese kann die Zustände "Off"(=Licht aus) und "On"(=Licht an) annehmen. Die Zustände werden durch entsprechende Aufrufe von "SwitchOn()" bzw. "SwitchOff()" gewechselt.



2.2.7.6 LightControlUnit



Internal Action für State: Anwesenheit

Event: AnwesenheitMelden(CNiemandDa) Action: LichtStatus = LightController.AbfrageLichtStatus()

Event: AnwesenheitMelden(CNiemandDa) Action: LichtStatus := LightController.AbfrageLichtStatus()

Entry-Aktion für State: Automatik-Betrieb

LightController.LichtAnsteuern(CLichtAn)

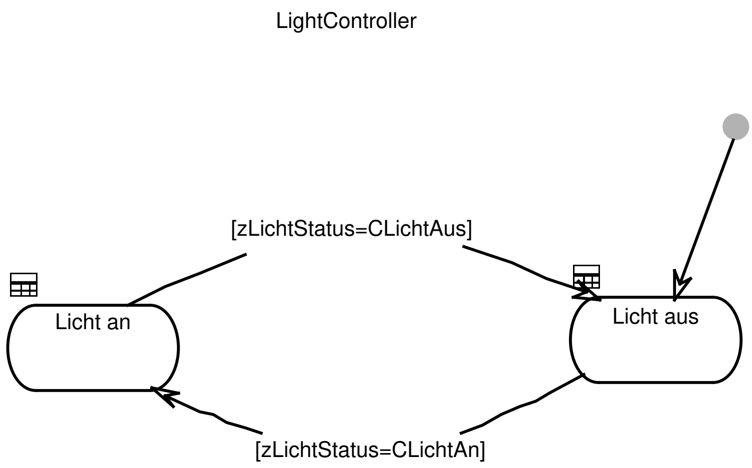
Internal Action für State: Automatik-Betrieb

Event: LichtStatusGewechselt() Action: LichtStatus := LightController.AbfrageLichtStatus()

Entry-Aktion für State: Wartezustand
LightController.LichtAnsteuern(CLichtAus)

Entry-Aktion für State: FM-Betrieb
LightController.LichtAnsteuern(CLichtAn)

2.2.7.7 LightController



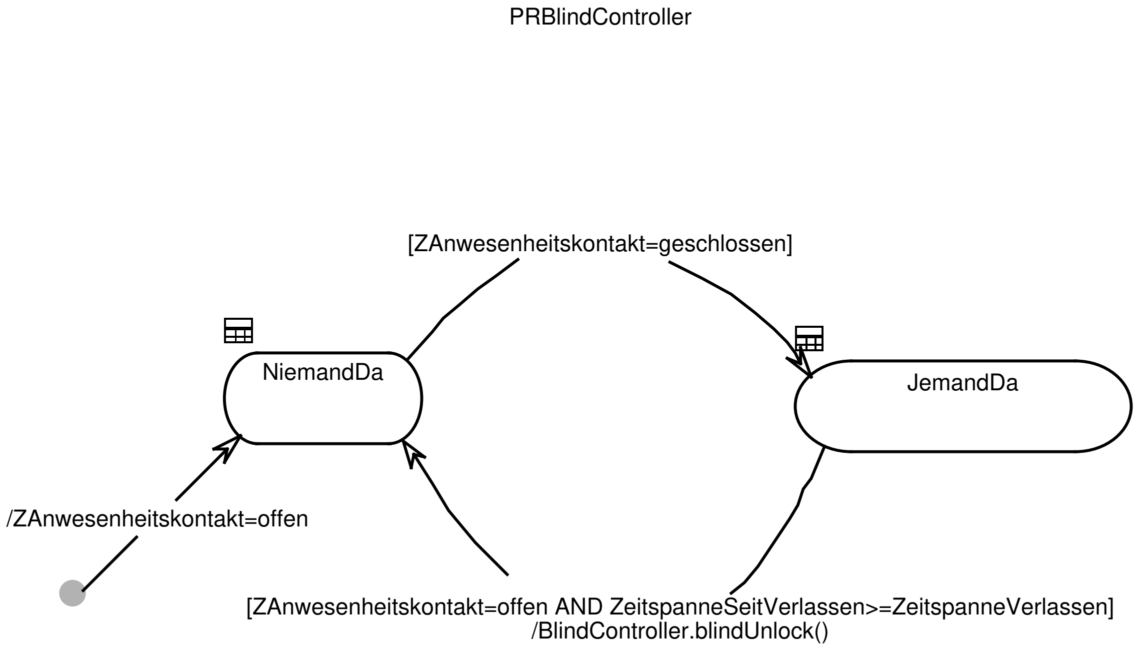
Exit-Aktion für State: Licht an
LightControlUnit.LichtStatusGewechselt()

Do-Aktivität für State: Licht an
zLichtStatus := Light.IsOn()

Exit-Aktion für State: Licht aus
LightControlUnit.LichtStatusGewechselt()

Do-Aktivität für State: Licht aus
zLichtStatus := Light.IsOn()

2.2.7.8 PRBlindController



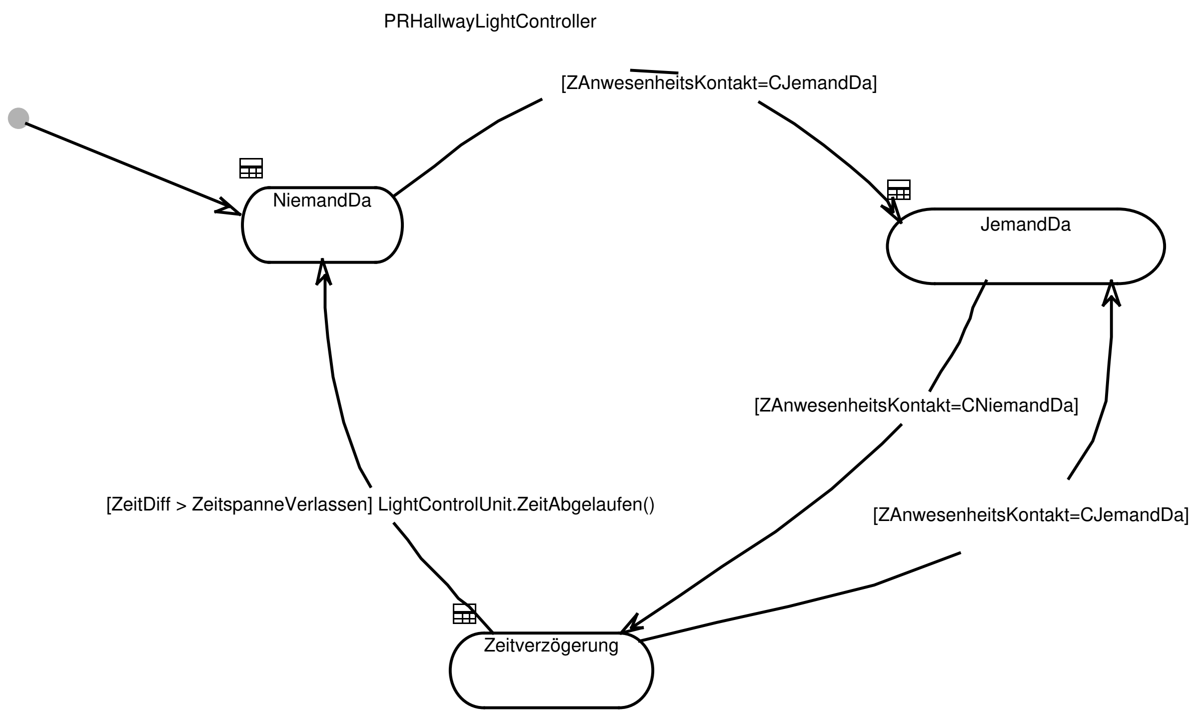
Do-Aktivität für State: NiemandDa

(ZAnwesenheitskontakt, ZeitspanneSeitVerlassen) := PresenceControllerRoom.StatusAktualisieren()

Do-Aktivität für State: JemandDa

(ZAnwesenheitskontakt, ZeitspanneSeitVerlassen) := PresenceControllerRoom.StatusAktualisieren()

2.2.7.9 PRHallwayLightController



Do-Aktivität für State: NiemandDa

ZAnwesenheitsKontakt := PresenceControllerHallway.StatusAktualisieren().ZAnwesenheitsKontakt

Entry-Aktion für State: JemandDa

LightControlUnit.AnwesenheitMelden(CJemandDa)

Do-Aktivität für State: JemandDa

ZAnwesenheitsKontakt := PresenceControllerHallway.StatusAktualisieren().ZAnwesenheitsKontakt

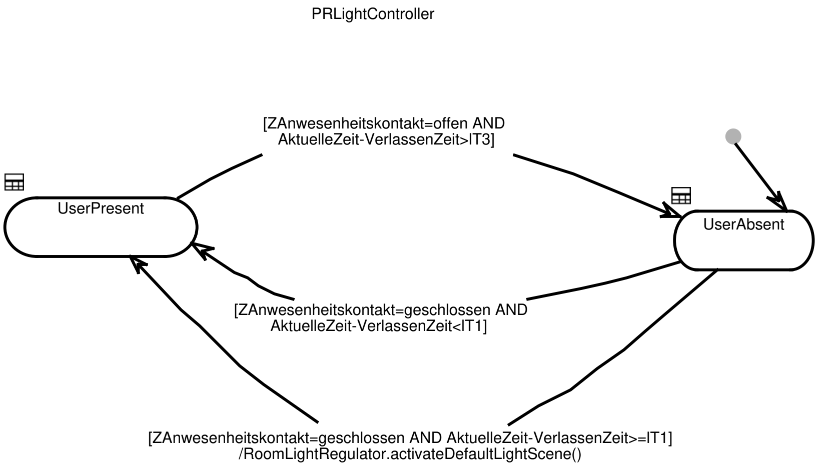
Entry-Aktion für State: Zeitverzögerung

LightControlUnit.AnwesenheitMelden(CNiemandDa)

Do-Aktivität für State: Zeitverzögerung

(ZAnwesenheitsKontakt, ZeitDiff) := PresenceControllerHallway.StatusAktualisieren()

2.2.7.10 PRLightController



Entry-Aktion für State: UserPresent

RoomLightRegulator.updateUserPresence(CJemandDa)

Do-Aktivität für State: UserPresent

ZAnwesenheitskontakt := PresenceController.ErmittleAnwesenheit()

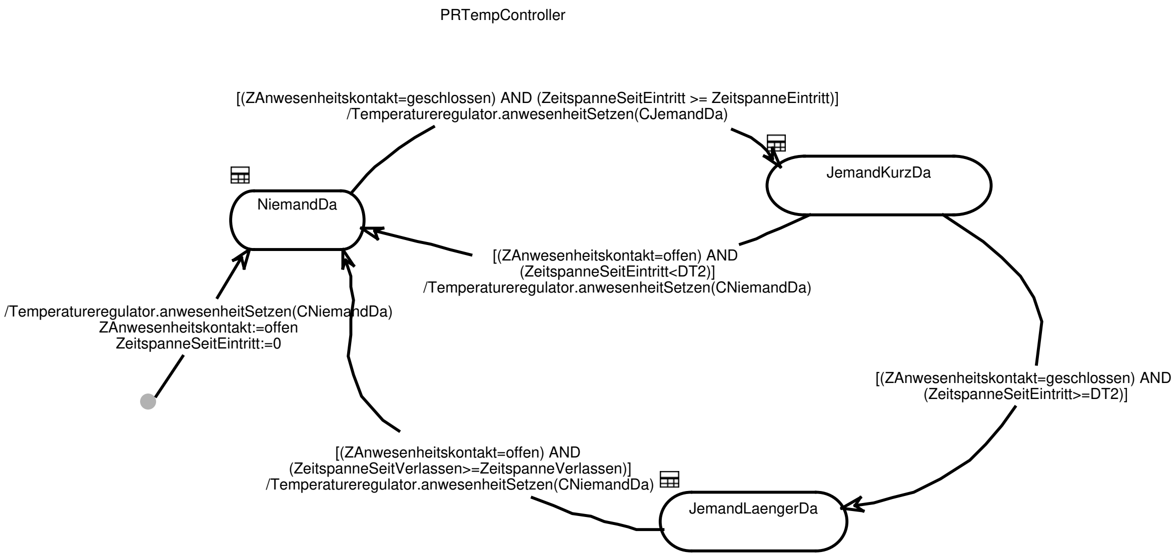
Entry-Aktion für State: UserAbsent

RoomLightRegulator.updateUserPresence(CNiemandDa)

Do-Aktivität für State: UserAbsent

ZAnwesenheitskontakt := PresenceController.ErmittleAnwesenheit()

2.2.7.11 PRTempController



Do-Aktivität für State: JemandKurzDa

(ZAnwesenheitskontakt, ZeitspanneSeitEintritt) := PresenceControllerRoom.StatusAktualisieren()

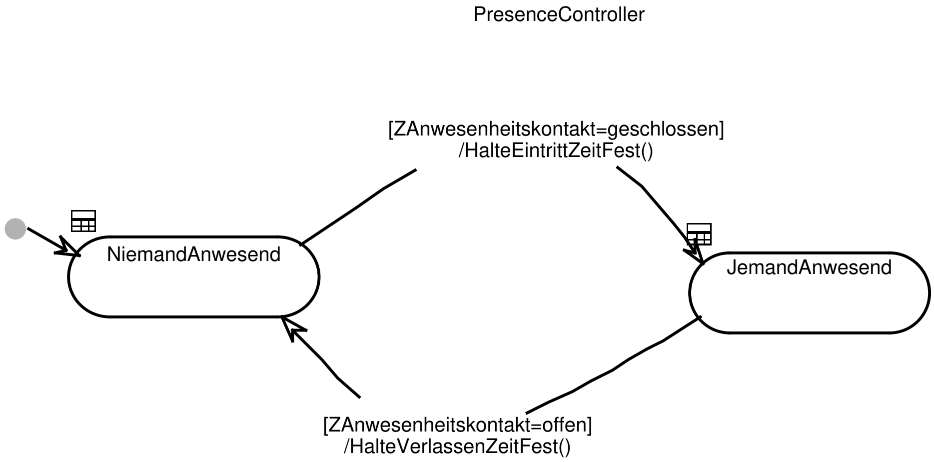
Do-Aktivität für State: NiemandDa

(ZAnwesenheitskontakt, ZeitspanneSeitEintritt) := PresenceControllerRoom.StatusAktualisieren()

Do-Aktivität für State: JemandLaengerDa

(ZAnwesenheitskontakt, ZeitspanneSeitVerlassen) := PresenceControllerRoom.StatusAktualisieren()

2.2.7.12 PresenceController



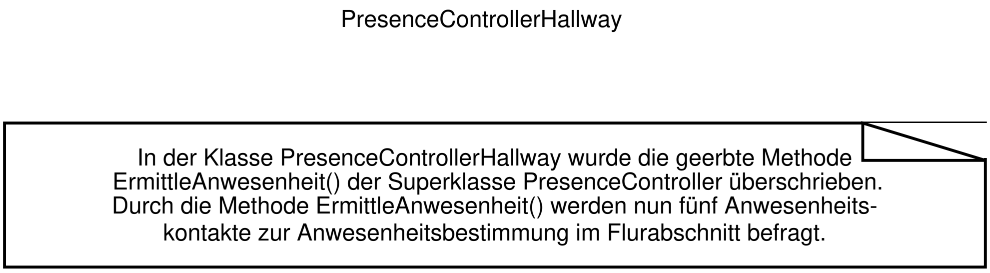
Do-Aktivität für State: NiemandAnwesend

ZAnwesenheitskontakt := ErmittleAnwesenheit()

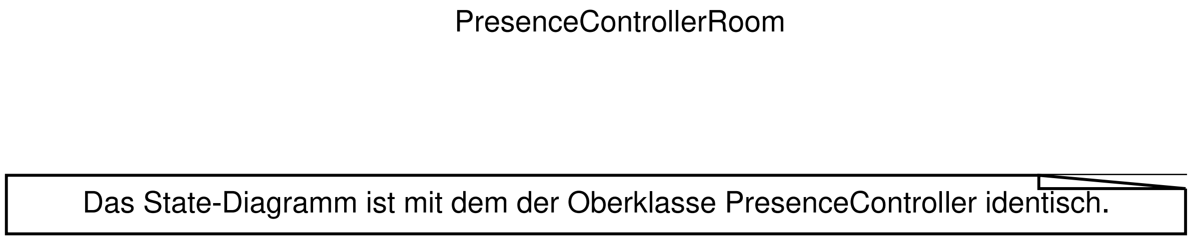
Do-Aktivität für State: JemandAnwesend

ZAnwesenheitskontakt := ErmittleAnwesenheit()

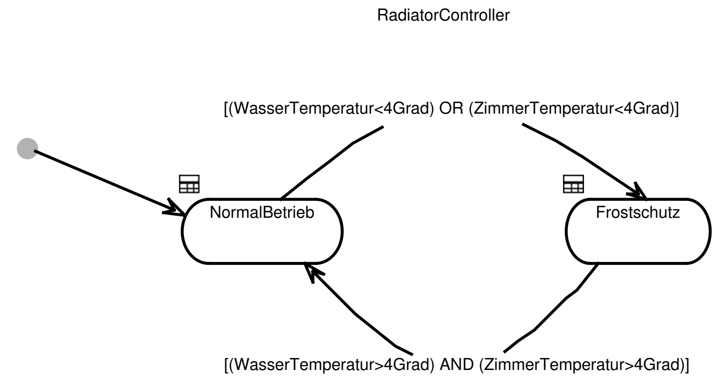
2.2.7.13 PresenceControllerHallway



2.2.7.14 PresenceControllerRoom



2.2.7.15 RadiatorController



Entry-Aktion für State: NormalBetrieb

ventilFuerNormalbetriebEinstellen()

Do-Aktivität für State: NormalBetrieb

alle 5 Minuten: WasserTemperatur:=getWaterTemperature() ZimmerTemperatur:=RoomTemperatureController.abfrageMinimalZimmerTemperatur()

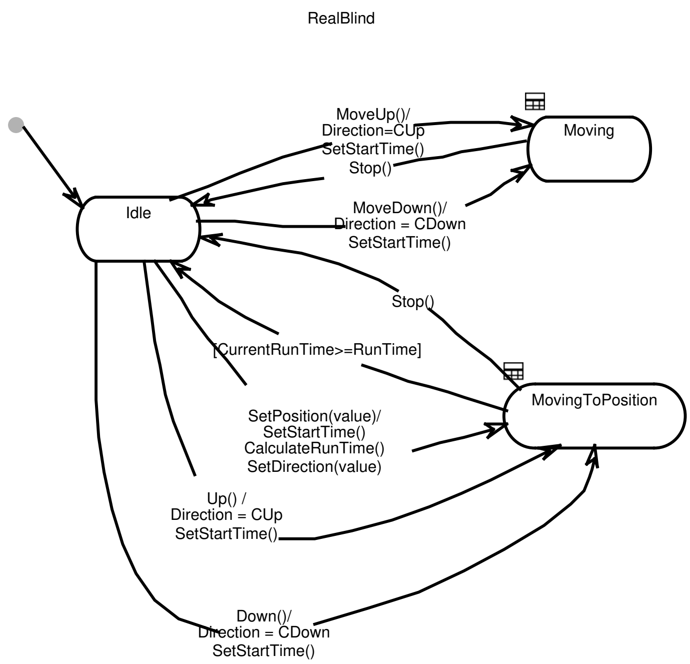
Entry-Aktion für State: Frostschutz

ventilFuerFrostschutzEinstellen()

Do-Aktivität für State: Frostschutz

alle 5 Minuten: WasserTemperatur:=getWaterTemperature() ZimmerTemperatur:=RoomTemperatureController.abfrageMinimalZimmerTemperatur()

2.2.7.16 RealBlind



Exit-Aktion für State: MovingToPosition

CalculatePosition

Do-Aktivität für State: MovingToPosition

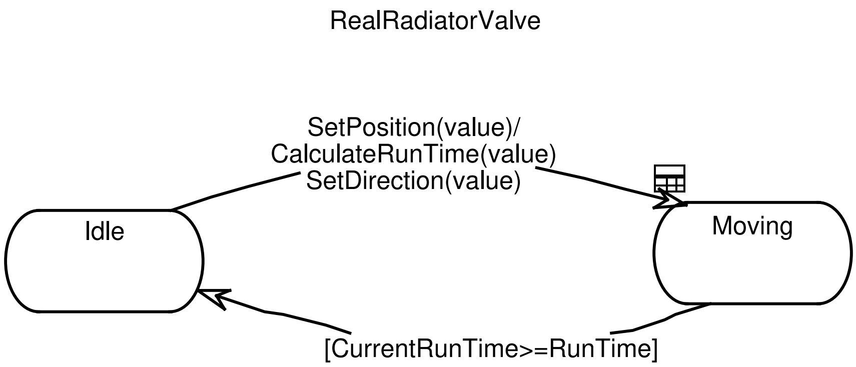
CurrentTime = getActualTime() CurrentRunTime = CurrentTime.getMinute()*60 + CurrentTime.get-Second - (StartTime.getMinute()*60+StartTime.getSecond())

Exit-Aktion für State: Moving

CalculatePosition()

2.2.7.17 RealRadiatorValve

Das Statediagramm beschreibt die Zustände der Klasse RealRadiatorValve. Diese ist im Zustand Idle, wenn keine Bewegung vorliegt. Sobald eine Positionsänderung veranlaßt wird wechselt die Klasse in den Zustand Moving. Diesen verläßt sie wieder, wenn die geforderte Position erreicht wurde.



Entry-Aktion für State: Moving

StartTime = getActualTime()

Exit-Aktion für State: Moving

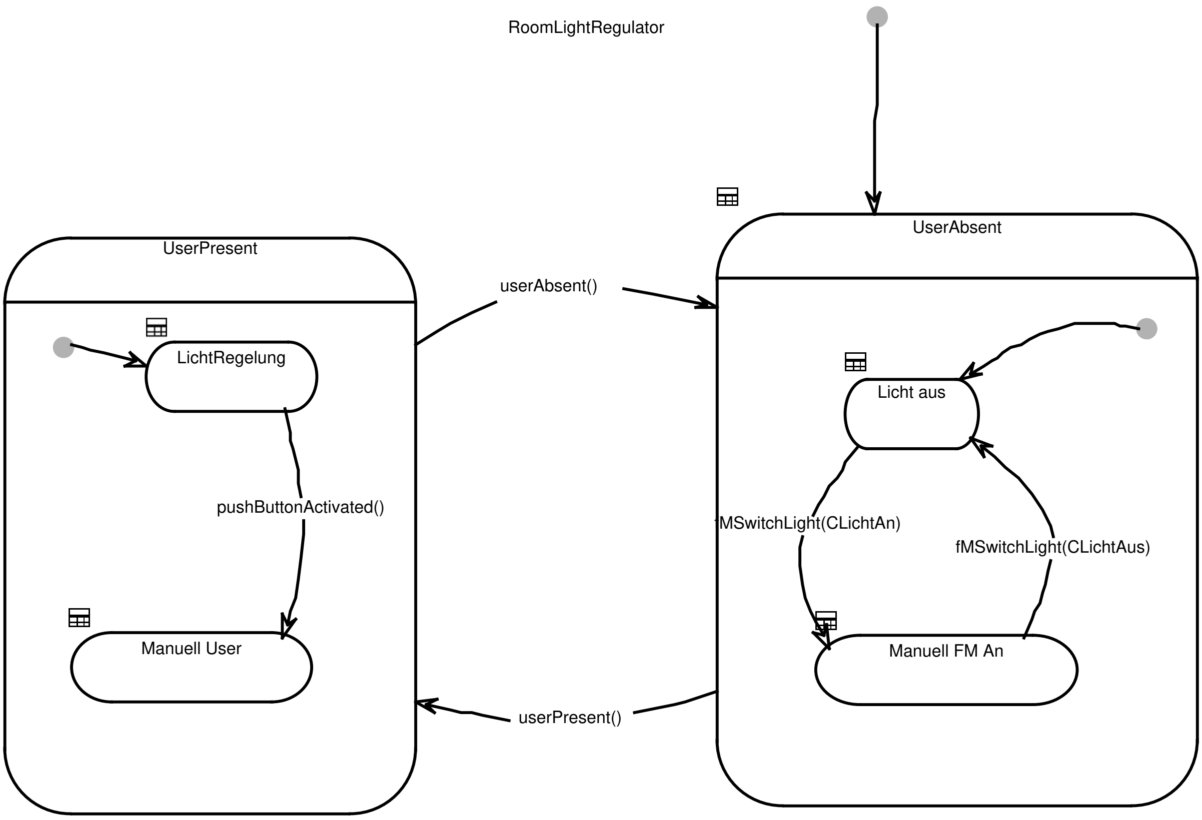
CalculatePosition()

Do-Aktivität für State: Moving

CurrentTime = getActualTime() CurrentRunTime = CurrentTime.getMinute()*60 + CurrentTime.getSecond() - (StartTime.getMinute()*60+StartTime.getSecond())

2.2.7.18 RoomLightController

2.2.7.19 RoomLightRegulator



Entry-Aktion für State: LichtRegelung

DimmableLightController.switchOn()
mode = automatic

Do-Aktivität für State: LichtRegelung

updateLight()

Entry-Aktion für State: Manuell User

mode = manual

Entry-Aktion für State: UserAbsent

mode = manual

Entry-Aktion für State: Licht aus

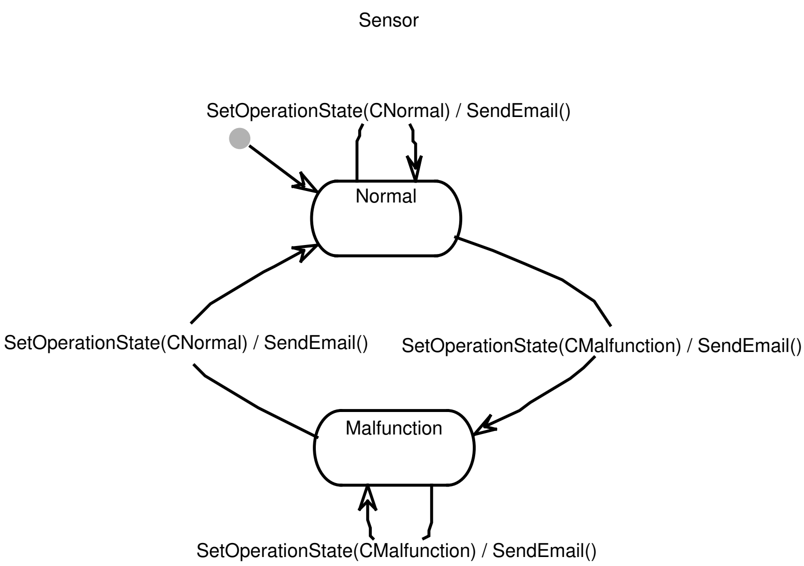
DimmableLightController.dim(0%)
DimmableLightController.switchOff()

Entry-Aktion für State: Manuell FM An

DimmableLightController.dim(100%)
DimmableLightController.switchOn()

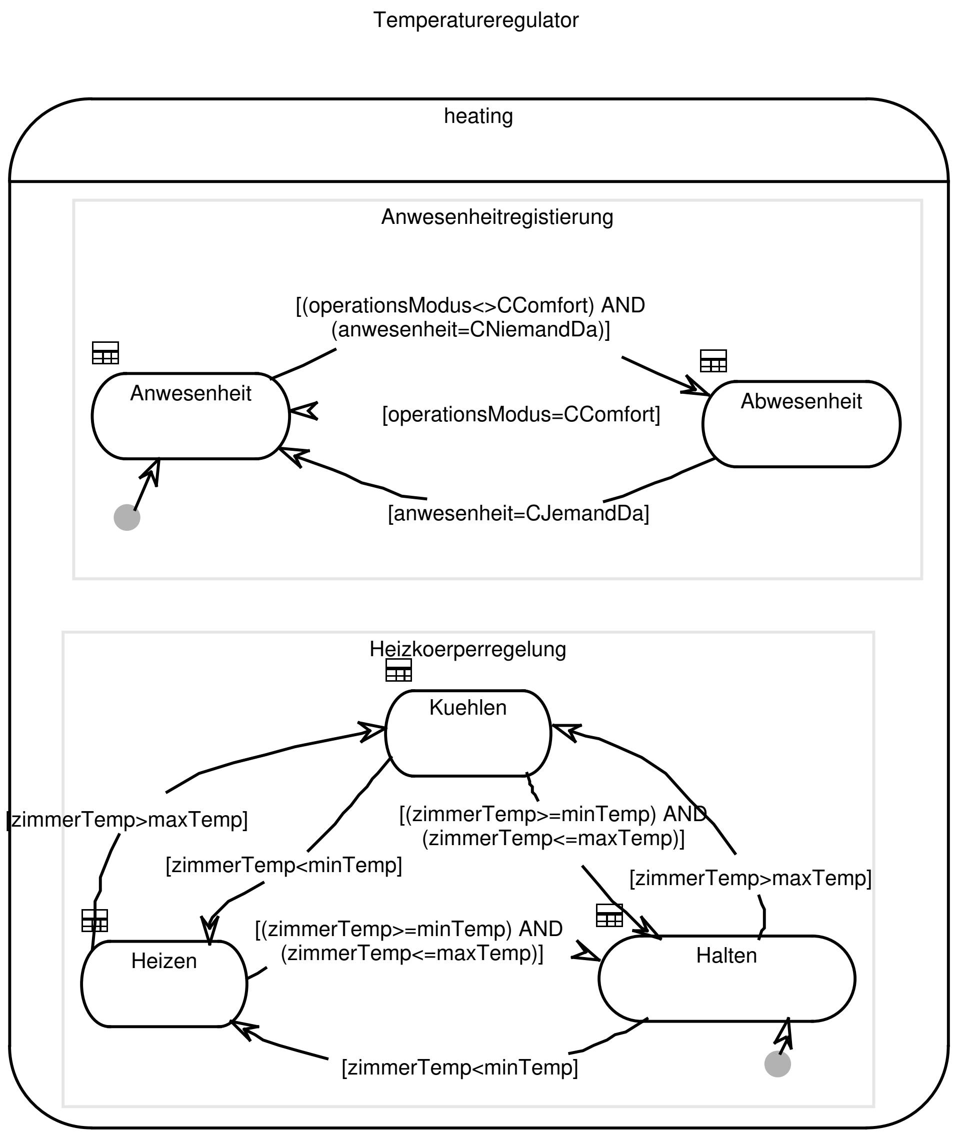
2.2.7.20 Sensor

Statediagramm für die Klasse Sensor, welches stellvertretend für alle abgeleiteten Klassen (z.B. OutdoorLightSensor) steht.
Diese Klasse(n) kann die Zustände "Normal" und "Malfunction", welche Normalbetrieb bzw. vom User oder Facility Manager eingegebenen "Defektbetrieb" darstellen. Die Zustände werden durch Aufruf der Methoden SetOperationState(CNormal) bzw. SetOperationState(CMalfunction) gewechselt.



2.2.7.21 Temperatureregulator

Das Statediagramm der Klasse Temperaturregulator zeigt auf der obersten Ebene zwei Zustände:
non-heating: Der Zustand, in dem keine Temperaturen definiert sind, und somit auch keine Blind- oder Heizeinstellung zur Temperaturregulierung stattfindet.
heating: Der Zustand, in dem die Temperaturregelung durchgeführt wird. Dieser wird aufgeteilt in Anwesenheitsregulierung, Blindsregelung und Heizkoerperregelung.



Entry-Aktion für State: Kuehlen

RadiatorController.ventilSchliessen()

Do-Aktivität für State: Kuehlen

alle 60s zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur()

Entry-Aktion für State: Anwesenheit

setzeZieltemperatur()

Entry-Aktion für State: Abwesenheit

setzeZieltemperatur()

Entry-Aktion für State: Heizen

RadiatorController.ventilOeffnen()

Do-Aktivität für State: Heizen

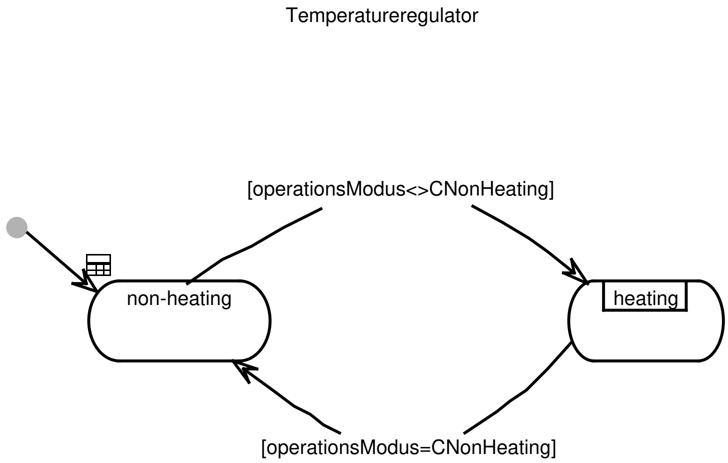
alle 60s zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur()

Entry-Aktion für State: Halten

RadiatorController.ventilRegelungVorbereiten()

Do-Aktivität für State: Halten

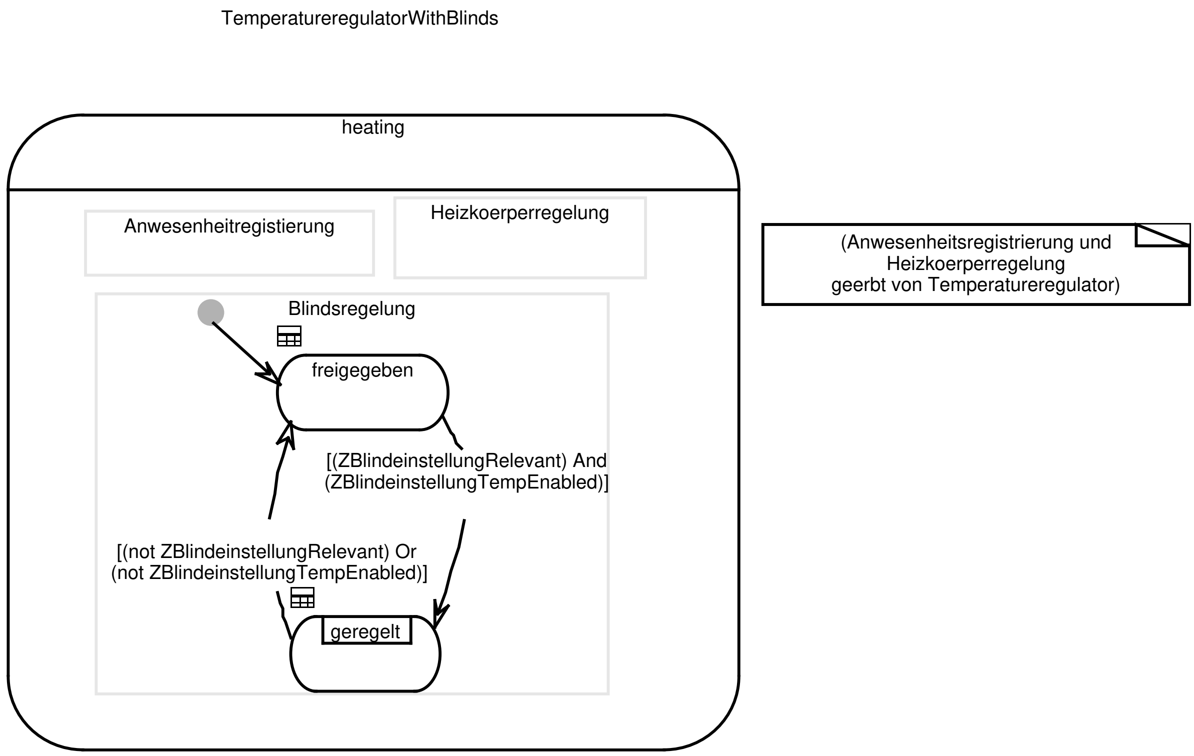
alle 60s zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur(),RadiatorController .ventilRegelung(zielTemp,zimmerTemp)



Entry-Aktion für State: non-heating

RadiatorController.ventilSchliessen()

2.2.7.22 TemperatureregulatorWithBlinds



Entry-Aktion für State: freigegeben

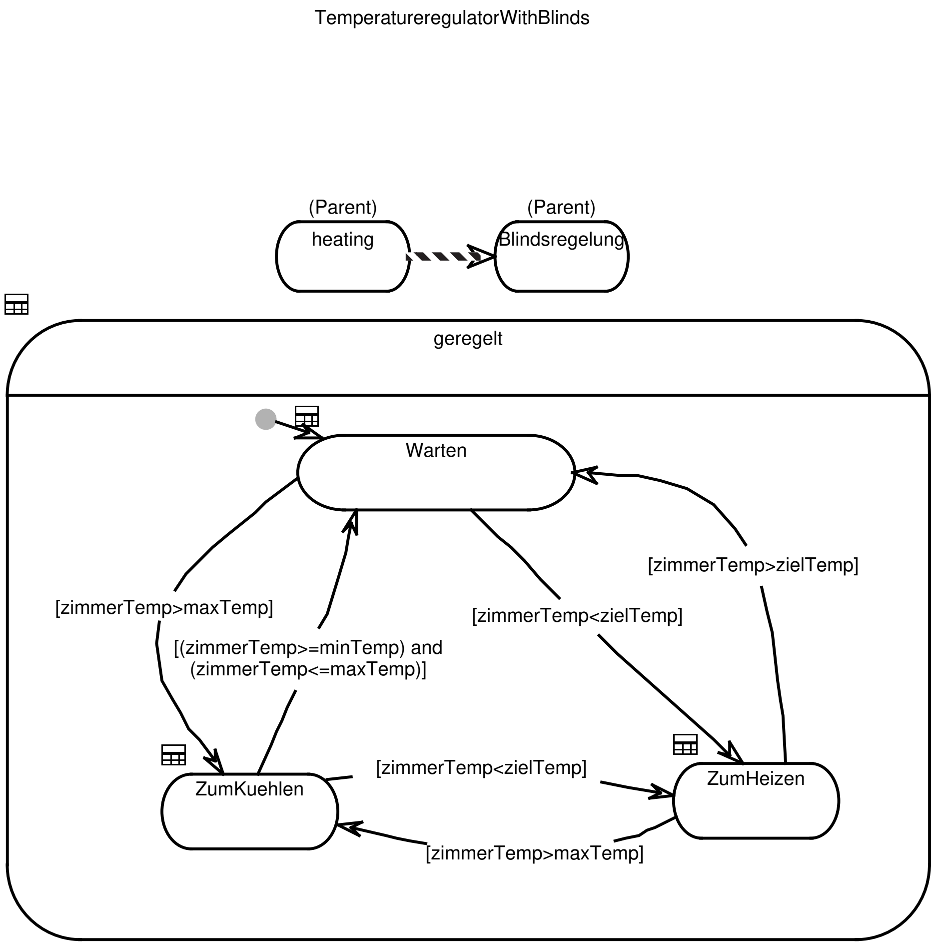
BlindController.BlindUnlockTemp()

Do-Aktivität für State: freigegeben

alle 5 Minuten ZBlindeinstellungRelevant, ZBlindEinstellungTempEnabled:=istBlindeinstellungRelevant(), BlindController.blindEinstellungTempEnabled()

Do-Aktivität für State: geregelt

alle 5 Minuten ZBlindeinstellungRelevant, ZBlindEinstellungTempEnabled:=istBlindeinstellungRelevant(), BlindController.blindEinstellungTempEnabled()



Do-Aktivität für State: geregelt

alle 5 Minuten ZBlindeinstellungRelevant, ZBlindEinstellungTempEnabled:=istBlindeinstellungRelevant(), BlindController.blindEinstellungTempEnabled()

Do-Aktivität für State: Warten

alle 5 Minuten: zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur()

Entry-Aktion für State: ZumHeizen

BlindController.blindEinstellungTemp(0, 0)

Do-Aktivität für State: ZumHeizen

alle 5 Minuten: zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur()

Entry-Aktion für State: ZumKuehlen

BlindController.blindEinstellungTemp(100,0)

Do-Aktivität für State: ZumKuehlen

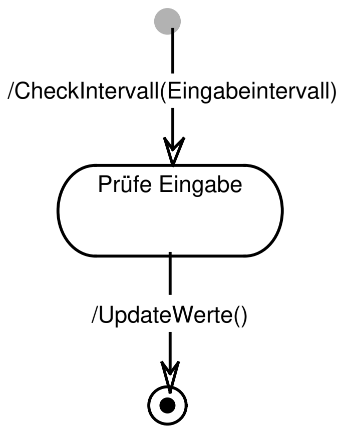
alle 5 Minuten: zimmerTemp:=RoomTemperatureController.abfrageMittelwertZimmertemperatur()

2.2.7.23 UML-Aktivitätsdiagramme

Die Aktivitätsdiagramme stellen komplexere Funktionalitäten einzelner Methoden dar. Im folgenden wird für jede Methode mit komplexerer Funktionalität ein Aktivitätsdiagramm angegeben. Nicht berücksichtigt werden Methoden, die lediglich zum Setzen oder Abfragen von Werten dienen.

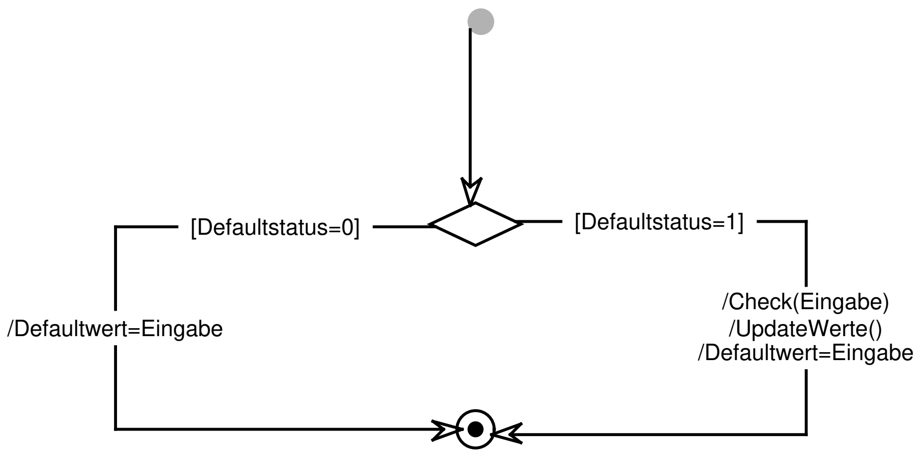
2.2.7.23.1 Benutzerdaten::SetAkzeptanzintervall()

Benutzerdaten::SetAkzeptanzintervall()



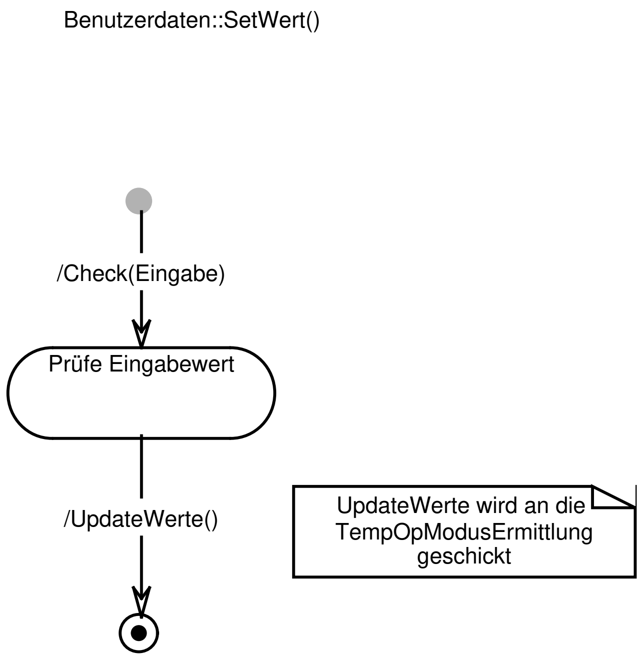
2.2.7.23.2 Benutzerdaten::SetDefaultwert()

Benutzerdaten::SetDefaultwert()

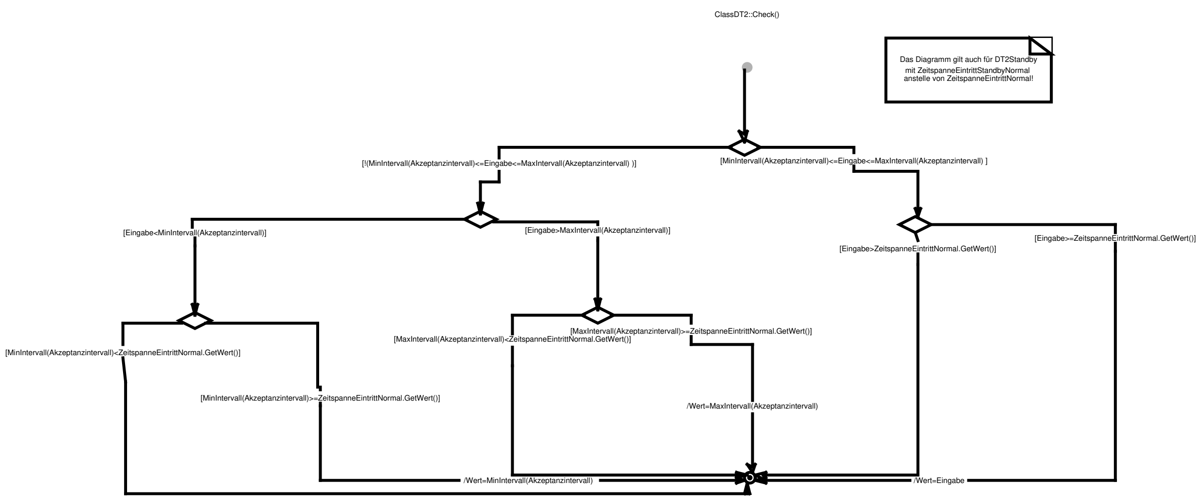


2.2.7.23.3 Benutzerdaten::SetWert()

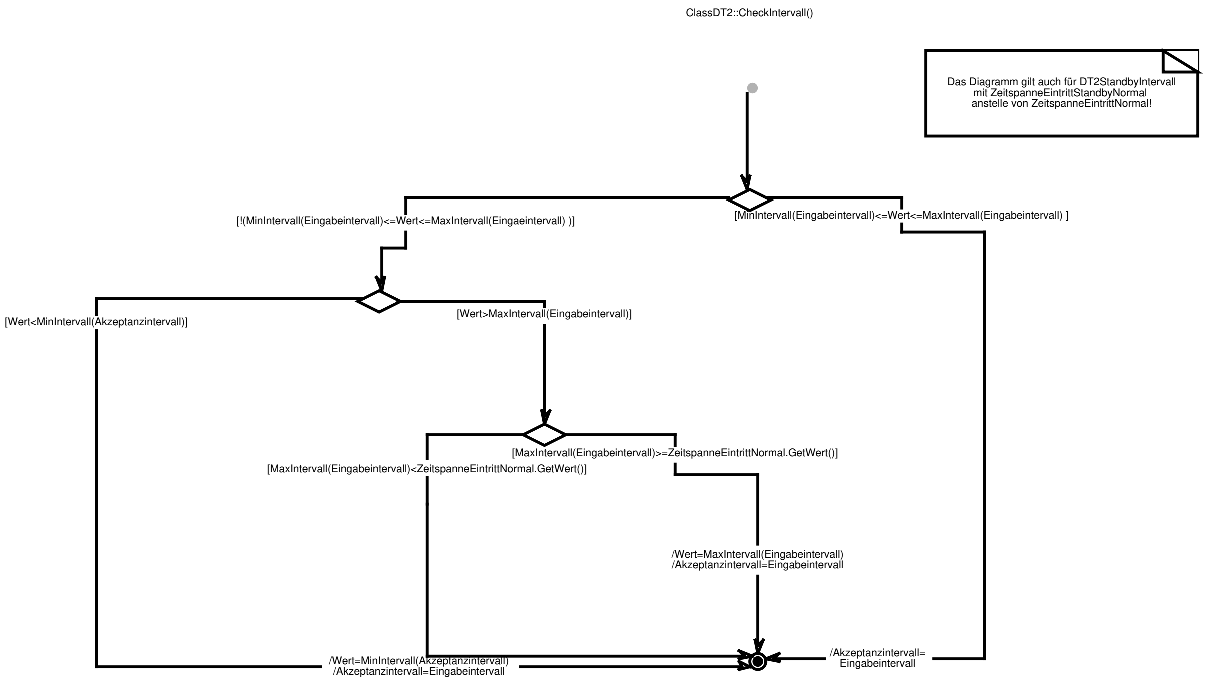
Bei einem Aufruf von SetWert wird, je nach dem, von welcher Klasse die Instanz gebildet wurde, der Wert au f verschiedene Eigenschaften geprueft und anschliessend ein UpdateWerte an die TempOpModusErmittlung geschickt.



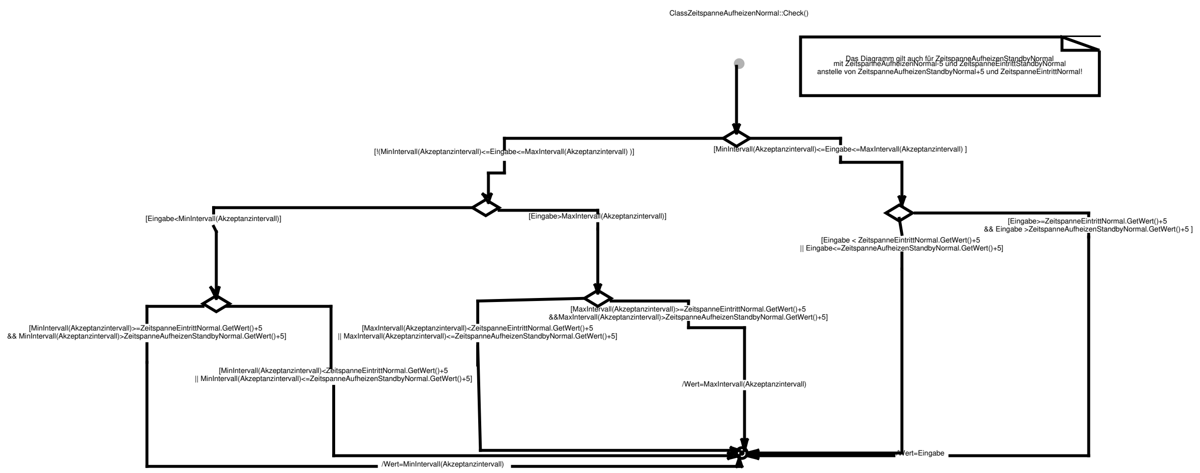
2.2.7.23.4 ClassDT2::Check()



2.2.7.23.5 ClassDT2::CheckIntervall()

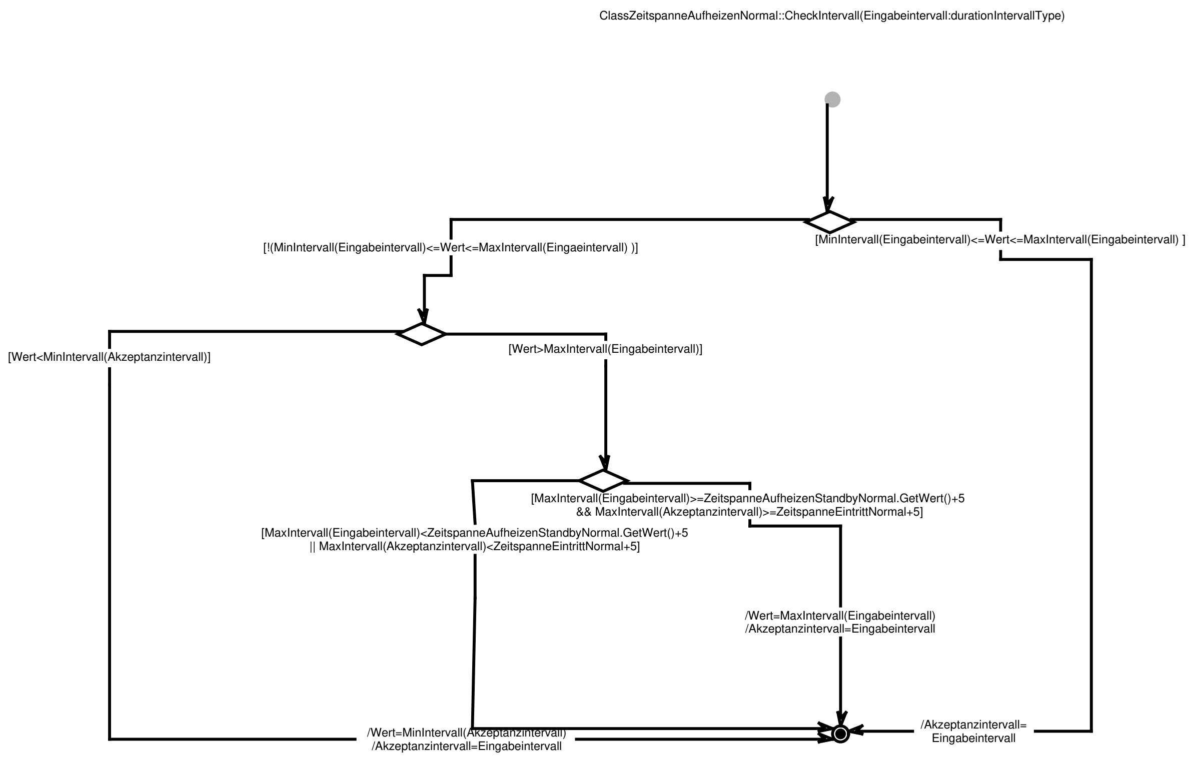


2.2.7.23.6 ClassZeitspanneAufheizenNormal::Check()

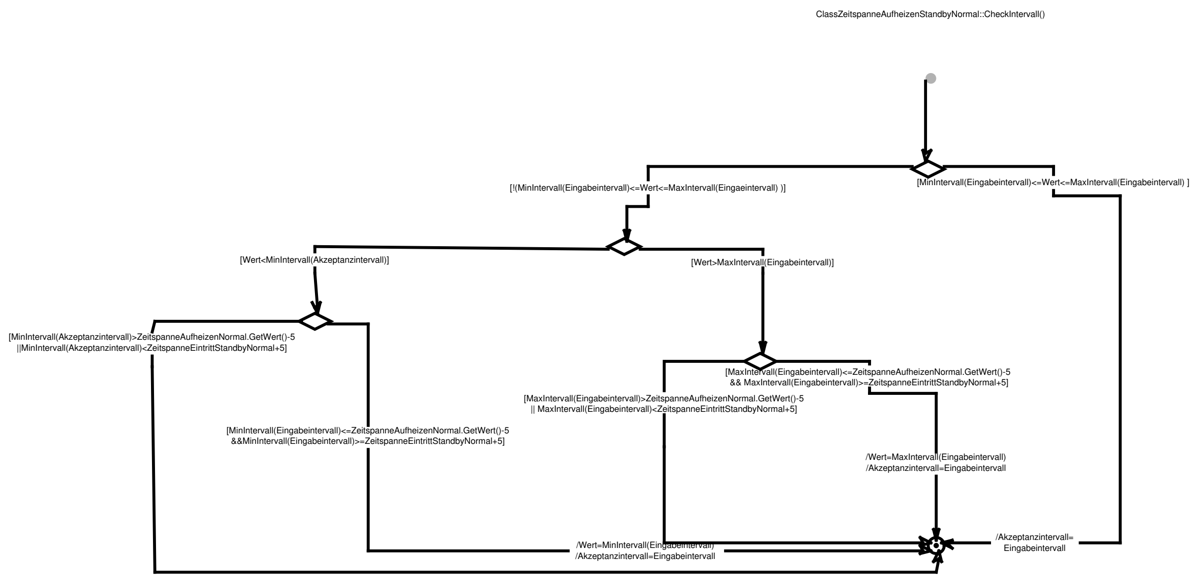


2.2.7.23.7 ClassZeitspanneAufheizenNormal::CheckIntervall(Eingabeintervall:dura-

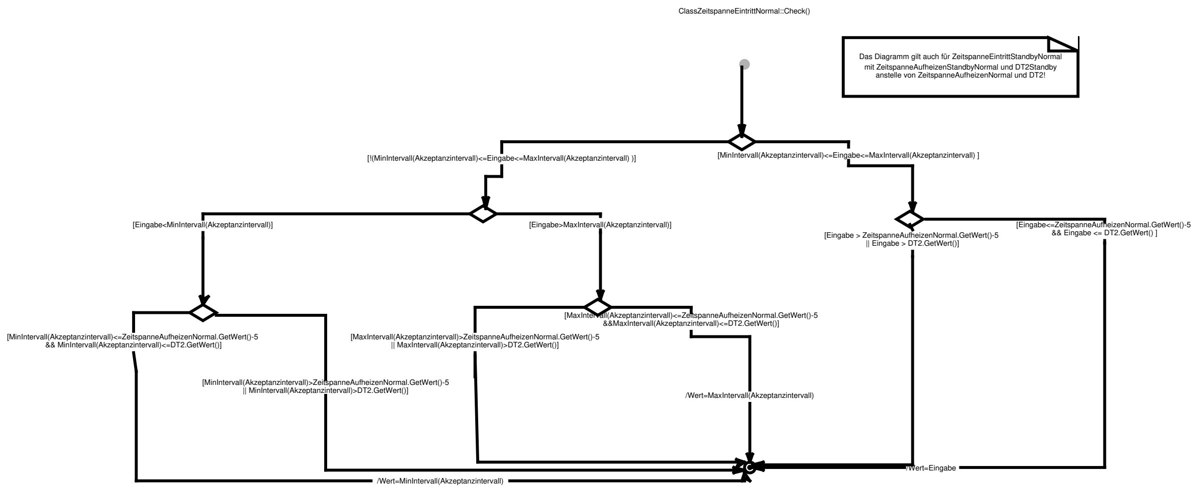
tionIntervallType)



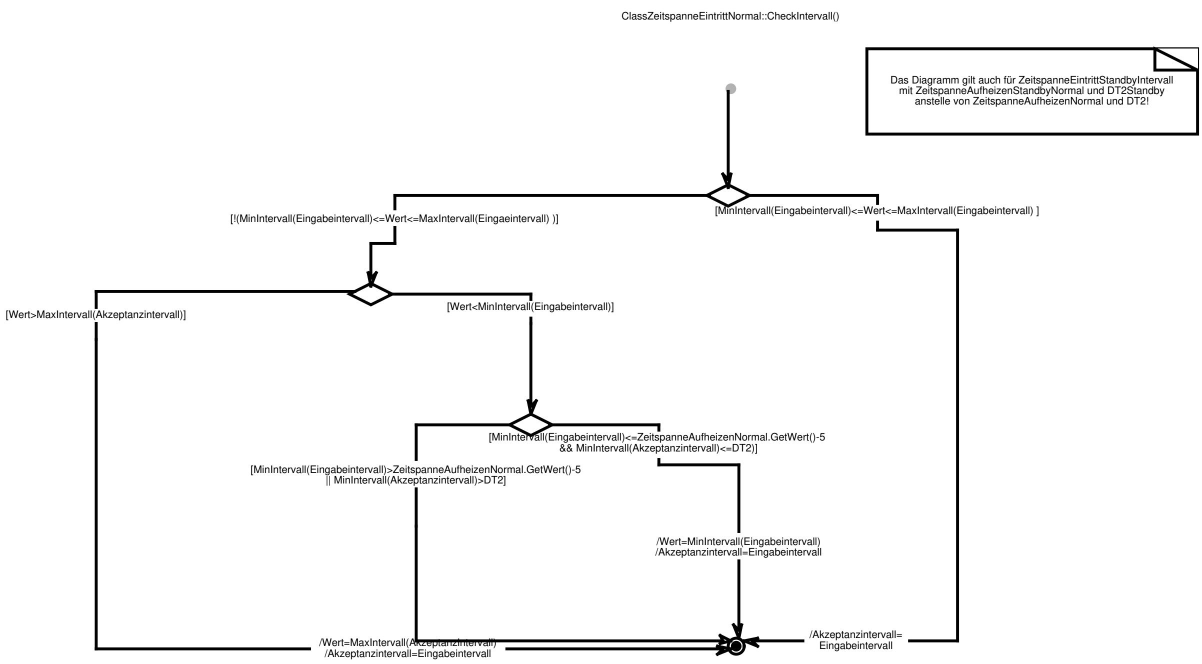
2.2.7.23.8 ClassZeitspanneAufheizenStandbyNormal::CheckIntervall()



2.2.7.23.9 ClassZeitspanneEintrittNormal::Check()



2.2.7.23.10 ClassZeitspanneEintrittNormal::CheckIntervall()



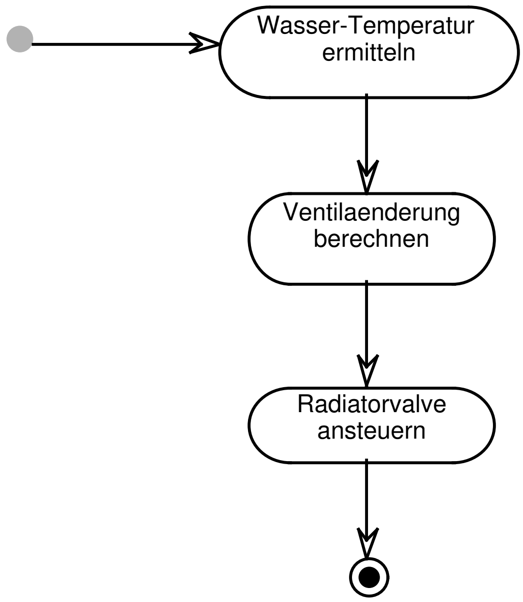
2.2.7.23.11 RadiatorController::ventilRegelung(zielTemp:tempType, sollTemp:tem-

pType)

RadiatorController::ventilRegelung(zielTemp:tempType, sollTemp:tempType)

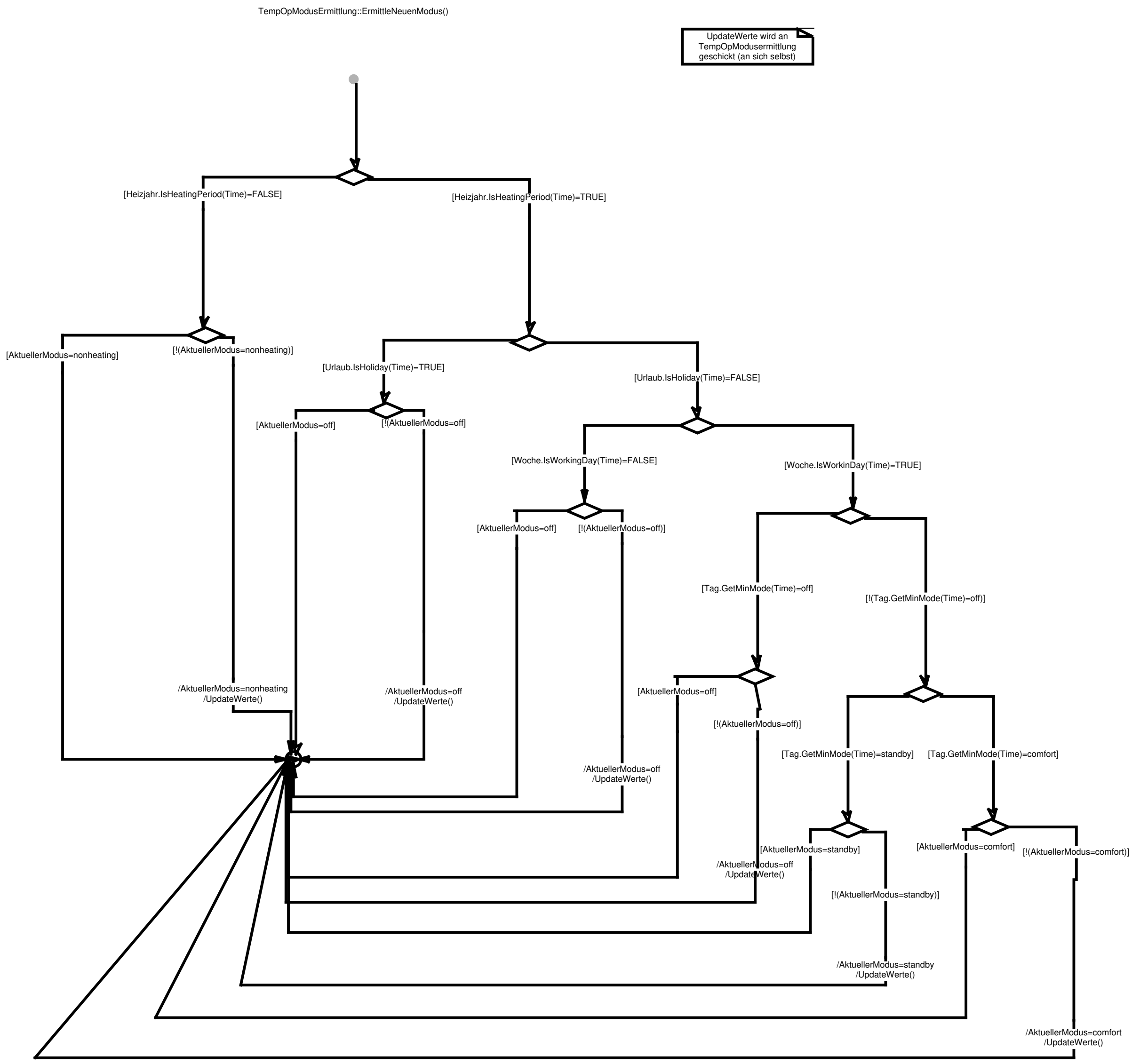
2.2.7.23.12 RadiatorController::ventilRegelung(zielTemp:tempType, zimmerTemp:tempType)

RadiatorController::ventilRegelung(zielTemp:tempType, zimmerTemp:tempType)



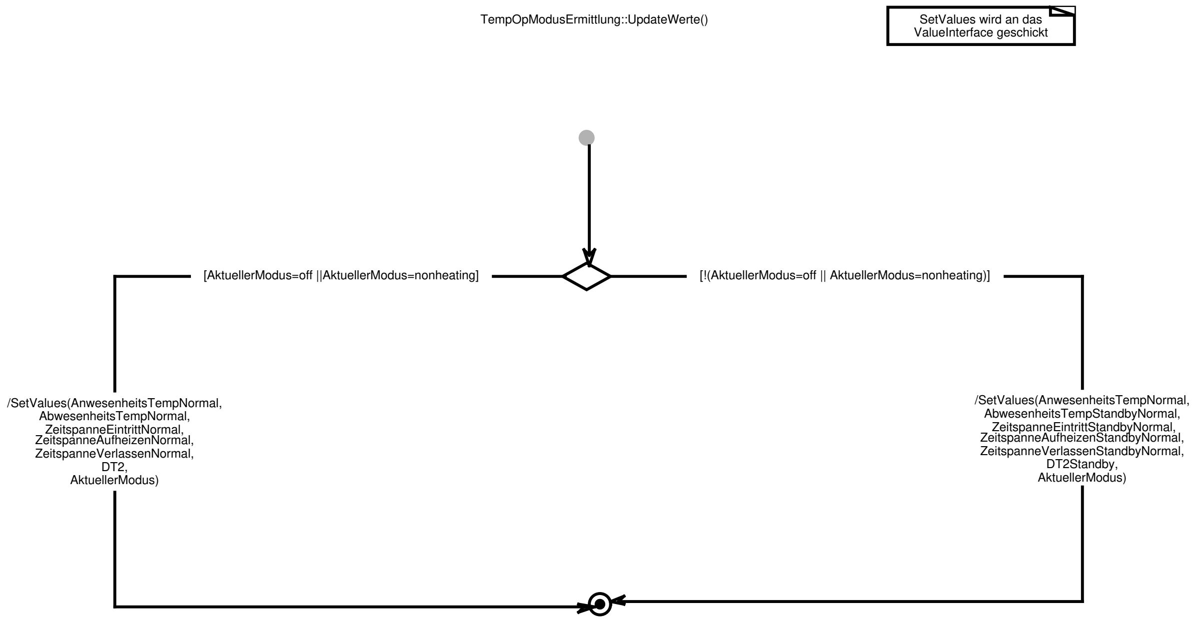
2.2.7.23.13 TempOpModusErmittlung::ErmittleNeuenModus()

Bei einem Aufruf von ErmittleNeuenModus wird überprüft, ob sich wirklich ein neuer Modus ergeben hat und gegebenenfalls UpdateWerte verschickt.

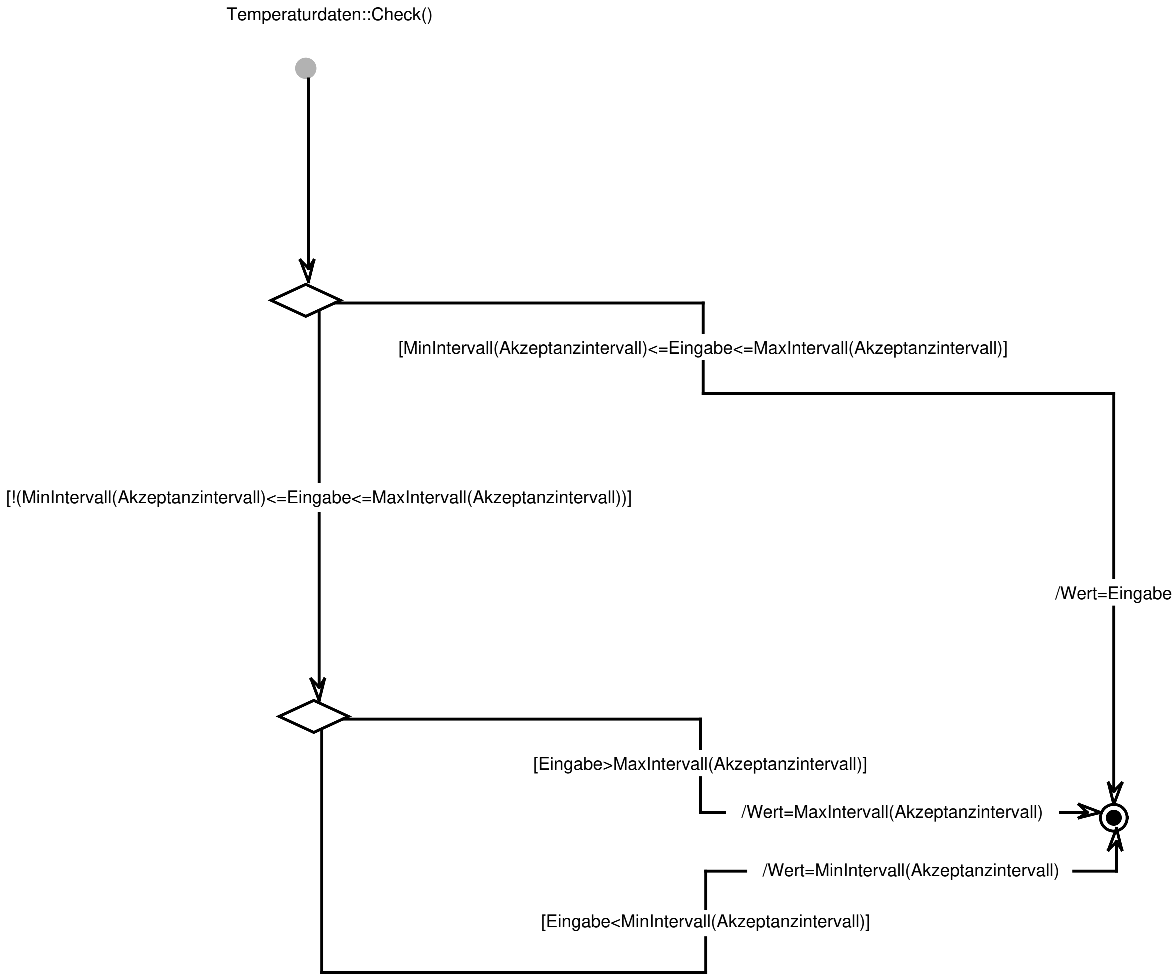


2.2.7.23.14 TempOpModusErmittlung::UpdateWerte()

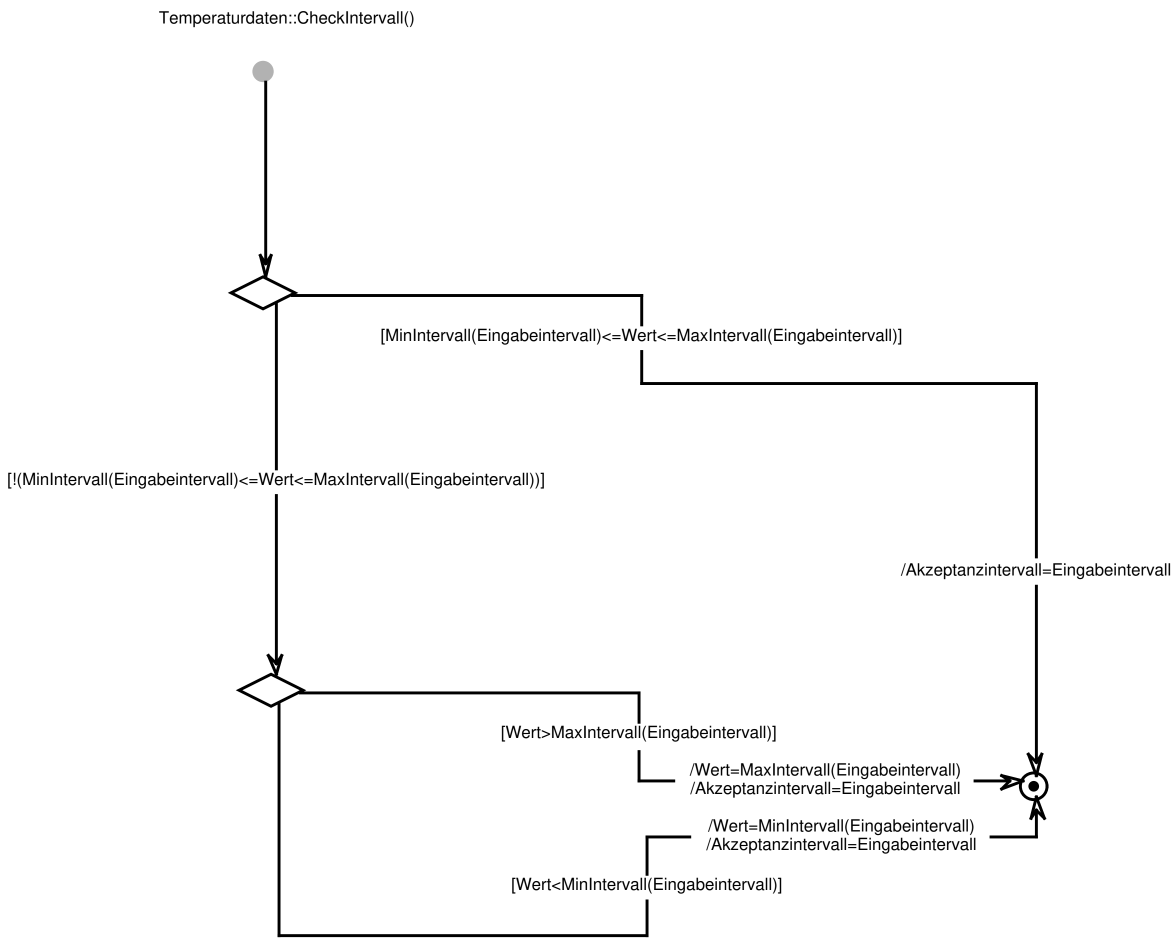
Bei einem Aufruf von UpdateWerte werden alle Werte, die an das ValueInterface geschickt werden sollen von den jeweiligen Instanzen abgefragt (HoleBenötigteWerte) und dann mittels SetValues verschickt.



2.2.7.23.15 Temperaturdaten::Check()



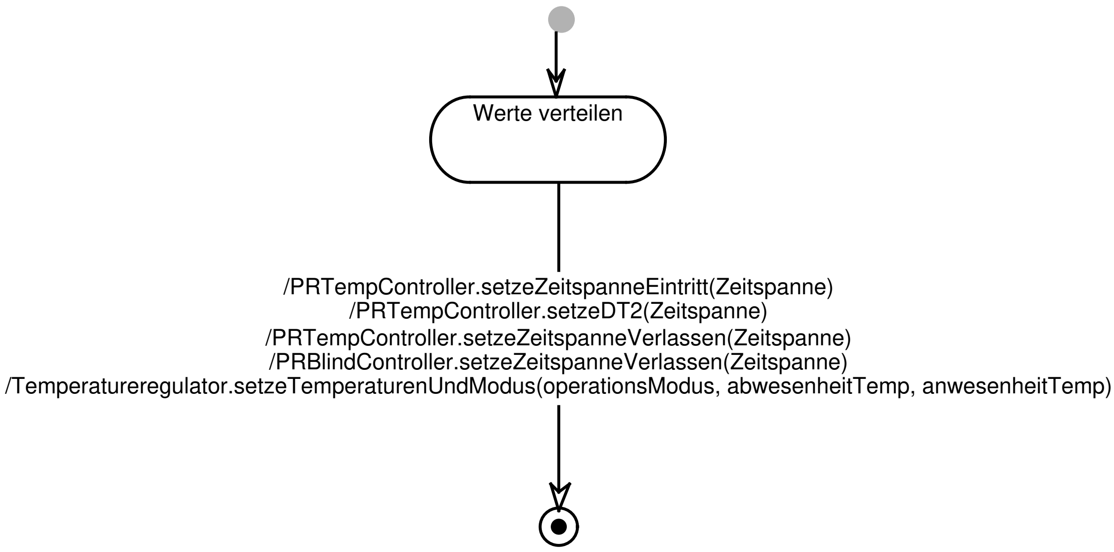
2.2.7.23.16 Temperaturdaten::CheckIntervall()



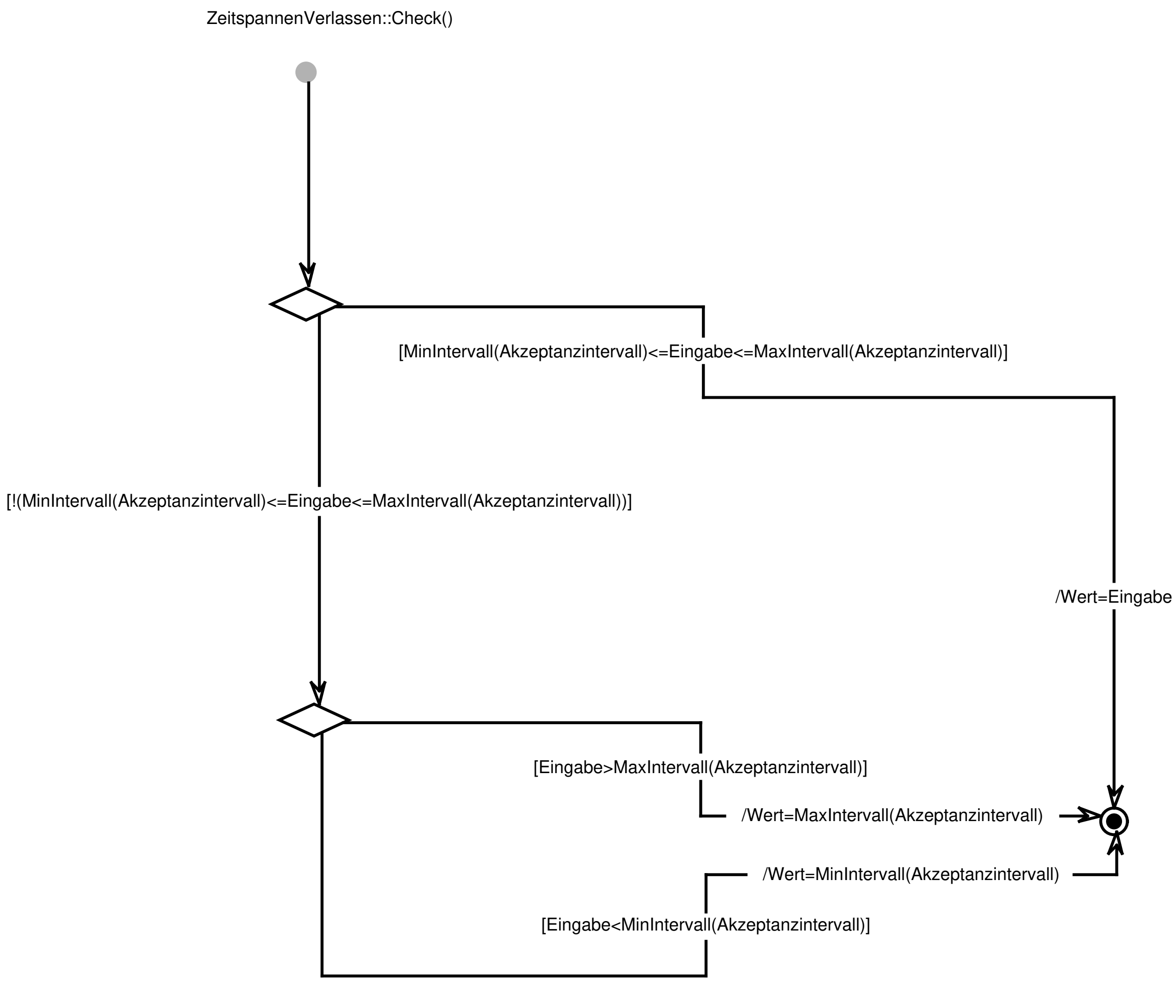
2.2.7.23.17 ValueInterface::SetValues()

Beschreibt den internen Ablauf der Werteweiterlegung von ValueInterface zu den entsprechenden Klassen

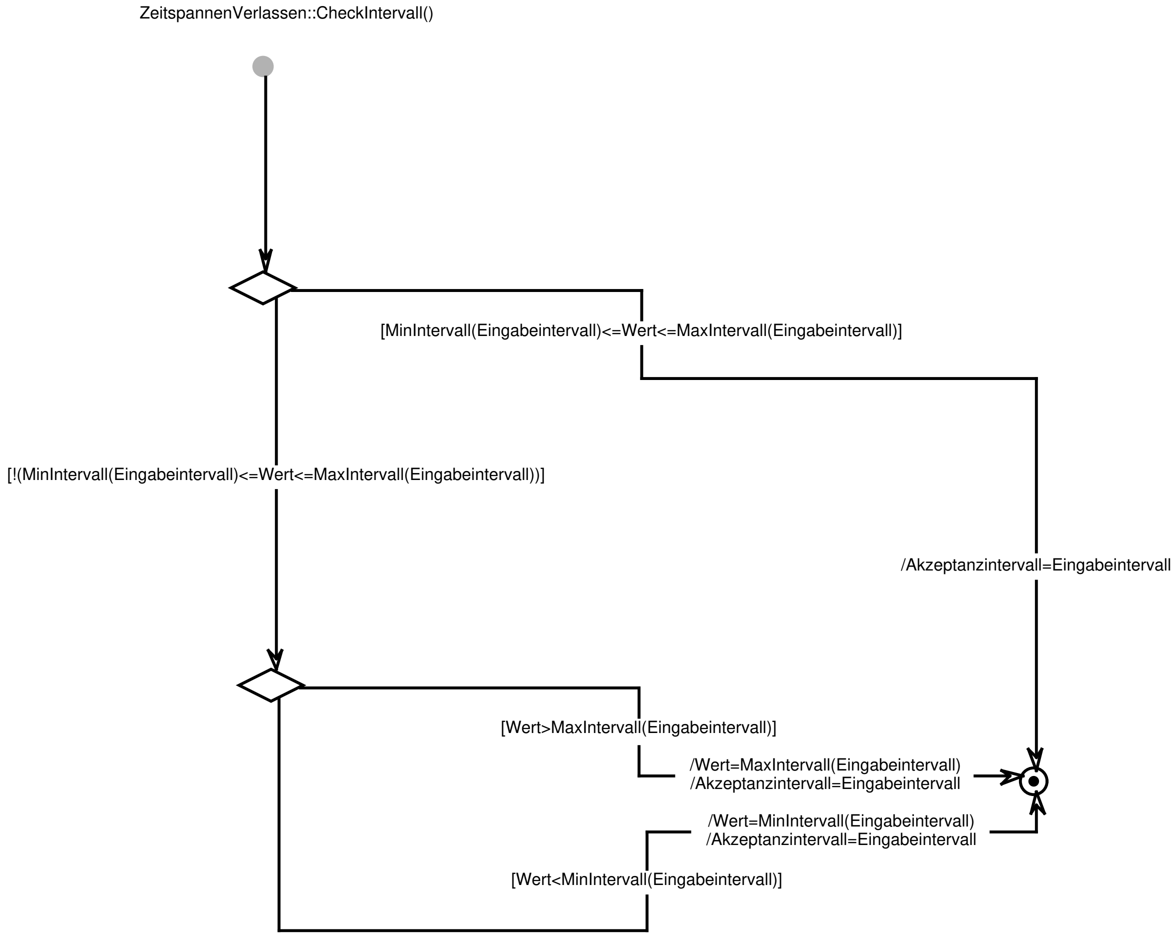
ValueInterface::SetValues()



2.2.7.23.18 ZeitspannenVerlassen::Check()



2.2.7.23.19 ZeitspannenVerlassen::CheckIntervall()



2.2.8 Verfolgbarkeitsmatrizen Analyse <-> Entwurf

Nachfolgend für jede Klasse der Analyse angegeben, durch welche Attribute und Methoden des Entwurfs die jeweiligen Attribute und Methoden der Analyse repräsentiert werden.

Die Methoden der Klassen *CentralHeatingSite*, *Contact*, *Motor*, *Radiator* und *Temperaturesensor* haben sich gegenüber der Analyse nicht geändert. Sie werden deshalb im folgenden nicht aufgeführt.

2.2.8.1 Klasse BlindController

	BlindController									
	lockStatus	blindUnlock()	blindEinstellungTemp()	blindUnlockTemp()	getBlindPosition()	getBlindAngle()	blindEinstellungTempEnabled()	MoveBlindUp()	MoveBlindDown()	StopBlind()
LockStatus	X									
BlindEinstellungTemp()			X							
BlindUnlock()		X								
BlindUnlockTemp()				X						

2.2.8.2 Klasse House

	House			
	MaxBenoetigteKesseltemperatur	Aussentemperatur	RegenAnSechtion()	callBack()
KesselTempAktualisieren()				X
RegenHaus()			X	

2.2.8.3 Klasse PRBlindController

	PRBlindController	
	ZeitspanneVerlassen	setzeZeitspanneVerlassen()
ZeitSpanneVerlassen	X	
SetValues()		X

2.2.8.4 Klasse PRTempController

	PRTempController					
	ZeitspanneEintritt	DT2	ZeitspanneVerlassen	setzeZeitspanneEintritt()	setzeZeitspanneVerlassen()	setzeDT2()
ZeitspanneEintritt	X					
DT2		X				
ZeitspanneVerlassen			X			
SetValues()				X	X	X

2.2.8.5 Klasse PresenceController

	PresenceControllerRoom					
	VerlassenZeit	EintrittZeit	HoleAktZeit()	HalteEintrittZeitFest()	HalteVerlassenZeitFest()	StatusAktualisieren()
VerlassenZeit	X					
EintrittZeit		X				
HoleAktZeit()			X			
HalteEintrittZeitFest()				X		
HalteVerlassenZeitFest()					X	
StatusAktualisieren()						X

2.2.8.6 Klasse RadiatorController

	RadiatorController						
	regelungsVentilEinstellung	ventilRegelung()	ventilOeffnen()	ventilSchliessen()	ventilFuerFrostschutzEinstellen()	ventilFuerNormalbetriebEinstellen()	getWaterTemperature()
RegelungsVentilEinstellung	X						
VentilEinstellungVeraendern()		X	X	X			
VentilFuerFrostschutzEinstellen()					X		
VentilFuerNormalbetriebEin-stellen()						X	
GetWaterTemperature()							X

2.2.8.7 Klasse Room

	Room		
	Flaeche	BenoetigteKesselTempAbfragen()	RegenZimmer()
Flaeche	X		
BenoetigteKesselTempAbfragen()		X	
RegenZimmer()			X

2.2.8.8 Klasse RoomtemperatureController

	RTC		**Temperatureregulator**			
	abfrageMinimalZimmertemperatur()	abfrageMittelwertZimmertemperatur()	minTemp	maxTemp	zielTemp	setzeZieltemperatur()
MinTemp			X			
MaxTemp				X		
ZielTemp					X	
AbfrageZimmerTemperatur()	X	X				
ZieltemperaturSetzen()						X

2.2.8.9 Klasse Section

	Section	
	BenoetigteKesselTempAbfragen()	RegenAnZimmer()
BenoetigteKesselTempAbfragen()	X	
RegenAbschnitt()		X

2.2.8.10 Klasse Temperatureregulator

	Temperatureregulator							**RadiatorController**			**TemperatureregulatorWithBlinds**
	abwesenheitTemp	anwesenheitTemp	anwesenheit	operationsModus	anwesenheitSetzen()	setzeTemperaturenUndModus()	ventilNachregeln()	ventilOeffnen()	ventilSchliessen()	ventilRegelung()	istBlindeinstellungRelevant()
AbwesenheitTemp	X										
AnwesenheitTemp		X									
Anwesenheit			X								
OperationsModus				X							
AnwesenheitSetzen()					X						
SetValues()						X					
VentilNachregeln()							X				
ZimmerTempGut()										X	
ZimmerTempZuHoch()									X		
ZimmerTempZuTief()								X			
ZimmerTempUeberZiel()										X	
ZimmerTempUnterZiel()										X	
istBlindeinstellungRelevant()											X

2.2.8.11 Klasse ValueInterface

	ValueInterface
	SetValues()
SetValues()	X

Die Methode der Klasse *Jahresintervalle* hat sich gegenüber der Analyse nicht geändert. Sie wird deshalb im folgenden nicht aufgeführt.

2.2.8.12 Klasse Benutzerdaten

	Benutzerdaten													
	Defaultstatus	Defaulttwert	Wert	Akjzeptanzintervall	SetDefaultwert()	SetWert()	SetAkzeptanzintervall()	GetWert()	GetAkzeptanzintervall()ll	GetDefaulwert	MinIntervall()	MaxIntervall()	Check()	CheckIntervall()
Defaultstatus	X													
Defaultwert		X												
Wert			X											
Akzeptanzintervall				X										
SetDefaultwert()					X									
SetWert						X					X	X	X	
SetAkzeptanzintervall()							X							X
GetWert								X						
GetAkzeptanzintervall()									X					
GetDefaultwert()										X				

	Temperaturdaten	ZeitspannenVerlassen	ClassDT2	ClassDT2Standby	ClassZeitspanneAufheizenNormal	ClassZeitspanneAufheizenStandbyNormal	ClassZeitspanneEintrittNormal	ClassZeitspanneEintrittStandbyNormal	Temperaturdaten	ZeitspannenVerlassen	ClassDT2	ClassDT2Standby	ClassZeitspanneAufheizenNormal	ClassZeitspanneAufheizenStandbyNormal	ClassZeitspanneEintrittNormal	ClassZeitspanneEintrittStandbyNormal
	Check()	Check()	Check()	Check()	Check()	Check()	Check()	Check()	CheckIntervall()	CheckIntervall()	CheckIntervall()	CheckIntervall()	CheckIntervall()	CheckIntervall()	CheckIntervall()	CheckIntervall()
SetWert	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X
SetAkzeptanzintervall()	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X	X

2.2.8.13 Klasse TempOpModusErmittlung

	TempOpModusErmittlung			Heizperioden	Urlaubszeiten	Wochenintervalle	Tagesintervalle
	AktuellerModus	UpdateWerte()	ErmittleNeuenModus()	IsHeatingperiod()	IsHoliday()	IsWorkingday()	GetMinMode()
AktuellerModus	X						
UpdateWerte()		X					
ErmittleNeuenModus			X	X	X	X	X

2.2.8.14 Klasse ControlPanelFM

In der Klasse ControlPanelFM wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt.

2.2.8.15 Klasse House

In der Klasse ControlPanelFM wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt.

2.2.8.16 Klasse Section

In der Klasse ControlPanelFM wurden das Attribut SectionNr und die Methoden SetZeitspanneVerlassen(), EnterFMLichtAnsteuern(), GetSectionNr() eingefügt.

2.2.8.17 Klasse HallwaySection

In der Klasse ControlPanelFM wurden die Methoden SetZeitspanneVerlassen() und EnterFMLichtAnsteuern() eingefügt.

2.2.8.18 Klasse PresenceControllerHallway

Die Klasse PresenceControllerHallway war in der Analyse nicht vorhanden.

2.2.8.19 Klasse PresenceController

In der Klasse PresenceController wurden die Attribute SectionNr und RoomNr neu eingefügt.

2.2.8.20 Klasse PRHallwayLightController

Die Methoden und die Attribute der Klasse PRHallwayLightController haben sich gegenüber der Analyse nicht geändert.

2.2.8.21 Klasse LightControlUnit

	LightControlUnit			
	AnwesenheitMelden()	LichtStatusGewechselt()	FMLichtAnsteuerenn()	ZeitAbgelaufen()
JemandDa()	X			
NiemandDa()	X			
LichtStatusGewechselt()		X		
FM_SchaltetEin()			X	
FM_SchaltetAus()			X	
ZeitAbgelaufen()				X

2.2.8.22 Klasse LightController

	LightController				
	SectinNr	RoomNr	LichtAnsteuern()	AbfrageLichtStatus()	SetLightControlUnit()
LichtStatus					
SchalteLichtEin()			X		
SchalteLichtAus()			X		
AbfrageLichtStatus()				X	

Die Methode der Klasse *Jahresintervalle* hat sich gegenüber der Analyse nicht geändert. Sie wird deshalb im folgenden nicht aufgeführt.

2.2.8.23 Klasse RoomLightRegulator

	RoomLightRegulator										
	defaultLightScene	userLightScene	mode	userPresent()	userAbsent()	userSetAmbientLightLevel()	activateDefaultLightScerne()	userSetLightScene()	pushButtonActivated()	fMSwitchOn()	fMSwitchOff()
DefaultLightScene	X										
UserLightScene		X									
Mode			X								
UserPresent()				X							
UserAbsent()					X						
UpdateLight()											
UserSetAmbientLightLevel()						X					
ActivateDefaultLightScene()							X				
UserSetLightScene()								X			
PushButtonActivated()									X		
FMSwitchOn()										X	
FMSwitchOff()											X

2.2.8.24 Klasse PRLightController

	PRLight Controll er			
	lT3	lT1	fMSetLT3()	fMSetLT1()
LT3	X			
LT1		X		
FMSetLT3()			X	
FMSetLT1()				X

2.2.8.25 Klasse DimmableLightController

	DimmableLightController			
	status	switchOn()	switchOff()	dim()
Status	X			
Dim()				X
SwitchOn()		X		
SwitchOff()			X	

2.2.8.26 Klasse BlindController

	BlindController			
	blindEinstellungLicht()	blindEinstellungLichtgEnabled()	getBlindPosition()	getBlindAngle()
BlindEinstellungLicht()	X			
BlindEinstellungLichtEnabled()		X		
GetBlindPosition()			X	
GetBlindAngle()				X

2.2.8.27 RoomLightController

	BlindController		RoomLightControllerWithBlinds			
	getRoomLightIntensity()	getOutdoorSensorValue()	getIntensityLevelBindsUp()	getIntensityLevelBlindsDown()	calculateOptmalBlindPos()	calculateOptimalBlindAngle()
getIntensityLevelAct()	X					
CalculateOptimalBlindPos()					X	
GetIntensityLevelBlindsUp()			X			
GetIntensityLevelBlindsDown				X		

Gegenueber der Analyse hat sich im HardwareWrapper nur die Klasse RadiatorValve geaendert, deshalb wird nur diese dargestellt..

2.2.8.28 Klasse RadiatorValve

	RadiatorValve		
	GetPosition()	Position	SetPosition()
GetPosition()	X	X	
Position		X	X
SetPosition()		X	X

2.3 Komponenten-Anforderungen

2.3.1 Allgemeine Datentypen

2.3.1.1 Wiederverwendete Klassen

Dieser Abschnitt listet die Datantypen und Klassen auf, die im Rahmen früherer Projekte erstellt worden sind und hier wiederverwendet werden.

C++ Klassen und Typdefinitionen
typedef unsigned int timeCompType
durationType timeCompType

C++ Klassen und Typdefinitionen

```
Klasse Clock:
class Clock
{
  friend ostream& operator<< (ostream& os, const Clock &c);
  friend istream& operator>> (istream& is, Clock &c);
   // these operators must be friends because they need direct access to
the
 // data members or base classes

private:
  static const int ChoursPerDay;
  // this const stores the number of hours per day;
  // this value is used in some calculations in the class
  static const int CminutesPerHour;
  // this const stores the number of minutes per hour;
 // this value is used in some calculations in the class
  timeCompType hour, minute;
  // these members represent the main data components of this class,
  // storing a certain point in time

public:
  Clock (const timeCompType h, const timeCompType m);
  Clock (const Clock &c);
  Clock (const durationType dur);
  Clock (void);
  // ~Clock: no members need to be handled by destructor

  Clock& operator= (const Clock &c);
  // memberwise assignment

  bool operator< (const Clock &c) const;
  // c1 < c2 iff
 // c1.hour < c2.hour || (c1.hour == c2.hour && c1.minute < c2.minute)

  bool operator<= (const Clock &c) const;
  // c1 <= c2 iff c1 < c2 || c1 == c2

  bool operator> (const Clock &c) const;
  // c1 > c2 iff !(c1 <= c2)

  bool operator>= (const Clock &c) const;
  // c1 >= c2  iff !(c1 < c2)

  bool operator== (const Clock &c) const;
  // c1 == c2 iff c1.hour == c2.hour && c1.minute == c2.minute
```

C++ Klassen und Typdefinitionen

```
bool operator!= (const Clock &c) const;
  // c1 != c2 iff !(c1== c2)

  Clock operator+ (const Clock &c) const;
  // c3 = c1 + c2:
  //    c3.hour = c1.hour + c2.hour + minuteOverflow;
 //    c3.minute = (c1.minute + c2.minute) % CminutesPerHour

  Clock operator- (const Clock &c) const;
  // c3 = c1 - c2 computes time span from object c1 to object c2, e.g.:
  // 21:00 - 22:30 = 1:30
  // attention: 23:00 - 1:15 = 2:1;
  // so, this operator doesn't really implement would usually be expected
  // from the minus operator

  timeCompType getMinute (void) const;
  // returns value of member minute

  timeCompType getHour (void)const ;
  // returns returns value of member hour

  void getClock (timeCompType &h, timeCompType &m) const;
  // returns values of members minute and hour through the reference
  // parameters

  void setMinute (const timeCompType m);
  // sets member minute to new value

  void setHour (const timeCompType h);
  // sets member hour to new value

  void setClock (const timeCompType h, const timeCompType m);
  // sets members hour and minute to new values

  Clock getClock (void) const;
  // returns a copy of this object

  durationType toDuration (void) const;
  // converts the stored point in time to a duration;
  // therefore the following value is returned:
  //   hour * CminutesPerHour + minutes

  operator durationType () const;
  // type cast from Clock to durationType;
}; // class Clock
```

C++ Klassen und Typdefinitionen

```
Klasse Day:
class Day
{
  friend ostream& operator<< (ostream& os, const Day &d);
  friend istream& operator>> (istream& is, Day &d);
   // these operators must be friends because they need direct access to
the
  // data members or base classes

private:
  static const int daysPerMonthArray[2][13];
  // this array contains the number of days of every month in a year;
  // daysPerMonthArray[0] contains the values for non-leap-years and
  // daysPerMonthArray[1] for leap-years;
  // daysPerMonthArray[?][0] is not used; for the sake of simpler access
  // the values are stored in daysPerMonthArray[?][1] ... [12]

  timeCompType dayNo, month, year;
  // main data members of this class which represent a specific date

public:
  Day (const timeCompType d, const timeCompType m, const timeCompType y);
  Day (const Day &d);
  Day (void);
  // ~Day: no members need to be handled by destructor

  Day& operator= (const Day &d);
  // memberwise assignment

  bool operator< (const Day &d) const;
  // d1 < d2 iff
  // d1.year < d2.year
  // || ((d1.year == d2.year) && (d1.month < d2.month))
  // || ((d1.year == d2.year) && (d1.month == d2.month)
  // && (d1.dayNo < d2.dayNo))

  bool operator<= (const Day &d) const;
  // d1 <= d2 iff d1 < d2 || d1 == d2

  bool operator> (const Day &d) const;
  // d1 > d2 iff !(d1 <= d2)
  bool operator>= (const Day &d) const;
  // d1 >= d2 iff !(d1 < d2)

  bool operator== (const Day &d) const;
  // d1 == d2 iff d1.year == d2.year && d1.month == d2.month
  // && d1.dayNo == d2.dayNo
```

C++ Klassen und Typdefinitionen

```
bool operator!= (const Day &d) const;
// d1 != d2 iff !(d1 == d2)

Day operator++ (int);
// postfix version of operator ++; computes next day and stores
// it in the object; delivers a copy of the object as value

timeCompType getDay (void) const;
// returns value of member dayNo

timeCompType getMonth (void) const;
// returns value of member month

timeCompType getYear (void) const;
// returns value of member year

void getDayMonth (timeCompType &d, timeCompType &m) const;
// returns values of members dayNo and month through the reference
// parameters

void getDay (timeCompType &d, timeCompType &m, timeCompType &y) const;
// returns values of members dayNo, month and year through the reference
// parameters

void setDay (const timeCompType d);
// sets member dayNo to new value

void setMonth (const timeCompType m);
// sets member month to new value

void setYear (const timeCompType y);
// sets member year to new value

void setDay (const timeCompType d, const timeCompType m, const
             timeCompType y);
// sets members dayNo, month and year to new values

void setDayFromDayIndex (const int d, const timeCompType y);
// the day index is the number of a day in a year (1.1. is day 1 and
// 31.12. is day 365 or 366 in a leap year); the parameter year is
// necessary to be able to tell wether February has 28 or 29 days;
// the methods set the members dayNo, month and year according to the
// date specified by the parameters of the method
```

C++ Klassen und Typdefinitionen

```
  Day getDayObject (void) const;
  // return a copy of the object

  int getDayInYear (void) const;
  // computes and returns the day index of the date stored in the object

  bool isLeapYear (const timeCompType y) const;
  // true iff the year specified by the parameter is a leap year

  bool isLeapYear (void) const;
  // true iff the member year specifies a leap year

  int daysInMonth (const timeCompType m, const timeCompType y) const;
  // computes and returns the number of days the month specified by the
  // parameters m(onth) and y(ear) has; the parameter y is necessary in
  // order to take care of leap years

  int daysInMonth () const;
  // computes and returns the number of days the month specified by the
  // members month and year has
}; // class Day
```

C++ Klassen und Typdefinitionen

```
Klasse SListBase:

class SListBase {
        slink *last;
        slink *current;
        int  endofsearch;
        int length;

    public:
        void insert(slink *a);
        void append(slink *a);
        slink *get();

        void clear() { last = 0; } // nothing to delete ?
        SListBase()  { last = 0; length = 0; }
        SListBase( slink* a )  { last = a->next = a; length = 1;}
                        // stupid constructor ??

        void reset();
        slink *next();
        slink *replacecurr( slink* a );
        slink *getcurr();
        int isEmpty() const
                { return (last == 0); }
        void insBeforeCurr(slink *a);

        int getLength() { return length; }

}; // class SListBase
```

C++ Klassen und Typdefinitionen

```
Klasse Slist:
 template <class T> class SList : public SListBase {
  public:
  SList(){} // constructor
  SList(SList<T>& Source)
       // copy constructor
       // necessary if lists are member of lists
  {
    T tmp;
    this->clear();
    Source.reset();
    while (Source.next(tmp) == notempty) {
      append(tmp);
    } }

  ~SList() // destructor
  {
    T tmp;
    // first of all, destroy list-entries
    while (get(tmp) == notempty);
  }

  void insert(const T& a)
       // Insert at head of list
  { SListBase::insert(new Tlink<T>(a)) ; }

  void insertBeforeCurr(const T& a)
       // Insert at head of list
  { SListBase::insBeforeCurr(new Tlink<T>(a)) ; }

  SList<T>& operator=(SList<T>& Source)
       // Assignment of Lists
       // initially destroys destination list elements
       // assigns a list of copied(!) source list elements
  {
    T tmp;
    if (this != &Source) {
      // no self assignment
      // first of all, destroy list-entries
      while (get(tmp) == notempty);
      // Copy Source into destination
      Source.reset();
      while (Source.next(tmp) == notempty) {
        append(tmp);
      }
   }
    return *this;
  }
```

C++ Klassen und Typdefinitionen

```
void append(const T& a)
     // append to tail
{
  SListBase::append(new Tlink<T>(a));
}

SListError get(T& ret)
     // gets element and deletes it from List
     // returns empty if lidt is empty and notempty otherwise
{
  Tlink<T> *link;
  SListError RetVal;
  (slink *)link = SListBase::get();
  if (link) {
    ret = link->info;
    RetVal = notempty;
    delete link;
  } else {
    RetVal = empty;
  }
  return RetVal;
}

void reset()
     // to be called before list can be traversed using next()
     // („const‟ is actually a lie, inserted to specify that
     // traversing the list does not really modify it)
{ SListBase::reset(); }


int getLength()
     // returns length of list
{ return SListBase::getLength(); }

SListError next(T& ret)
     // gets element and from list without deleting them
     // returns empty if list is empty and notempty otherwise
     // („const‟ is actually a lie, inserted to specify that
     // traversing the list does not really modify it)
{
  Tlink<T> *link;
  SListError RetVal;
  (slink *)link = SListBase::next();
  if (link) {
    ret = link->info;
    RetVal = notempty;
  } else {
    RetVal = empty;
  }
  return RetVal;
}
```

C++ Klassen und Typdefinitionen

```
  void deletecurr()
       // deletes list element, that was
       // reached using next
  {
    Tlink<T> *tmp;
    (slink *)tmp = SListBase::getcurr();
    if (tmp) {
      delete tmp;
    } else {
      cerr << „deletecurr: attempt to empty list\n“;
    }
  }

  void replacecurr(T& a)
       // replaces list element, that was
       // reached using next, by a copy(!)
       // of element a
  {
    Tlink<T> *tmp;
    (slink *)tmp = SListBase::replacecurr(new Tlink<T>(a));
    if (tmp) delete tmp;
  }

}; // class Slist
```

C++ Klassen und Typdefinitionen

```
Klasse Time:
class Time : public Clock, public Day
{
  friend ostream& operator<< (ostream& os, const Time &t);
  friend istream& operator>> (istream& is, Time &t);
   // these operators must be friends because they need direct access to
the
  // data members or base classes

public:
  Time (const timeCompType h, const timeCompType mi,
        const timeCompType d, const timeCompType mo, const timeCompType y);
  Time (const Time &t);
  Time (void);
  // ~Time: no members need to be handled by destructor

  Time& operator= (const Time &t);
  // memberwise assignment

  bool operator< (const Time &t) const;
  // t1 < t2 iff t1.day < t2.day || (t1.day == t2.day && t1.time < t2.time)

  bool operator<= (const Time &t) const;
  // t1 <= t2 iff t1 < t2 || t1 == t2

  bool operator> (const Time &t) const;
  // t1 > t2 iff !(t1 <= t2)

  bool operator>= (const Time &t) const;
  // t1 >= t2 iff !(t1 < t2)

  bool operator== (const Time &t) const;
  // t1 == t2 iff t1.day == t2.day && t1.time == t2.time

  bool operator!= (const Time &t) const;
  // t1 != t2 iff !(t1 == t2)

  Time operator++ (int);
  // postfix version of operator ++; computes next minute; in case of an
  // overflow hours, days, .... are updated as well;
  // the changes are stored in the object and a copy of the unchanged
  // object is returned as the operator‘s value

  void getTime (timeCompType &h, timeCompType &mi,
  timeCompType &d, timeCompType &mo, timeCompType &y) const;
   // returns all time components (from the day and time)in the value
result
```

C++ Klassen und Typdefinitionen

<pre> // parameters
 	
  void setTime (const timeCompType h, const timeCompType mi,
                const timeCompType d, const timeCompType mo, const
                timeCompType y);
  // sets new values for the day and time
}; // class Time</pre>

2.3.1.2 Datentypen aus dem UML-Entwurf

Die folgenden Datentypen basieren auf Typen und Klassen, die im UML-Entwurf verwendet werden.

C++ Typdefinition	Beschreibung
`typedef enum {CWertUngueltig, CKeinFehler} fehlerType`	Mögliche Rückgabewerte bei Fehlern.
`typedef enum {CJemandDa, CNiemandDa } zAnwesenheitType`	Mögliche Werte bei Benutzung von Anwesenheitswerten.

<pre>typedef struct{
 zAnwesenheitType ZAnwesenheitsKontakt;
 durationType   ZeitDiff;
} ZAnwesenheitZeitType</pre>
	Rückgabetyp des PresenceControllers bei Statusabfrage.

`typedef enum {CFensterGekippt, CFensterOffen, CFensterGeschlossen } zFensterType`	Mögliche Werte bei Benutzung des Fensterzustands.
`typedef enum {CZimmerTempZuTief, CZimmerTempGut, CZimmerTempZuHoch} zZimmerTempType`	Mögliche Werte für den Zimmertemperaturzustand.
`typedef int tempType`	Typ für Temperaturwerte. Wird zur Übersichtlichkeit und leichteren Änderbarkeit definiert.
`typedef int percentageType`	Typ für Prozentwerte. Wird zur Über sichtlichkeit und leichteren Änderbarkeit definiert.
`typedef int degreeType`	Typ für Winkelangaben. Wird zur Über sichtlichkeit und leichteren Änderbarkeit definiert.
`typedef enum {CNormal, CMalfunction} malfunctionType`	Mögliche Werte zur Eingabe von Enter-Malfunction.
`typedef enum {CUp, CDown} directionType`	Dieser Typ gibt die möglichen Richtungen an, in die sich die Blinds oder Radiator-Valves bewegen können.
`typedef enum {comfort,standby,off, nonheating} modeType`	Mögliche Modi in denen das System sein kann.

C++ Typdefinition	Beschreibung
`typedef struct{ durationType min; durationType max; } durationIntervallType`	Ein Zeitintervall mit seinem Anfang (min) und seinem Ende(max).
`typedef struct{ tempType min; tempType max; } tempIntervallType`	Ein Temperaturintervall mit seinem Anfang (min) und seinem Ende(max).
`typedef struct{ timeCompType min; timeCompType max; }minuteIntervallType`	Ein Minuten-Zeitintervall mit seinem Anfang (min) und seinem Ende(max).
`typedef struct{ timeCompType min; timeCompType max; }yearIntervallType`	Ein Jahrestage-Zeitintervall mit seinem Anfang (min) und seinem Ende(max).
`typedef enum {working,nonworking} week-DayType`	Mögliche Tagesmodi zur Festlegung der Arbeitstage.
`typedef enum {is,non} yearDayType`	Mögliche Tagesmodi zur Festlegung von Urlaubszeiten als auch Heizperioden.
`typedef enum {Mo, Di, Mi, Do, Fr, Sa, So} weekType`	Mögliche Tage der Woche.
`typedef struct { weekType min; weekType max; }weekIntervallType`	Struktur zum speichern der vom Benutzer eingegebenen Arbeitstage dh. ein Wochenintervall mit seinem Anfang min und seinem Ende max.
`typedef unsigned int dayOfYearType`	Beschreibt den aktuellen Tag im Jahr
`typedef unsigned int dayOfWeekType`	Beschreibt den aktuellen Tag in der Woche
`typedef unsigned int minuteOfDayType`	Beschreibt die aktuelle Minute des Tages
`typedef enum {CLichtAus, CLichtAn} zLichtStatusType`	Mögliche Werte für die Flurlichtanlage.

2.3.2 Klassendefinitionen

Im folgenden werden zu jeder Klasse definiert:

- Assoziationen zu Objekten anderer Klassen

  Dabei werden Assoziationen nur in die Richtung, in die eine Klasse Methoden einer anderen aufruft, durch einen Zeiger auf diese modelliert.

- Aggregationen von Objekten zu anderen Klassen
- Methodenschnittstellen von Konstruktoren, Destruktoren oder privaten Methoden. Die public-Methoden der Klassen wurden bereits im Kapitel 2.2.3 beschrieben.
- Imports von anderen Klassen

2.3.2.1 Klasse Actuator

2.3.2.1.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.1.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.1.3 Methodenschnittstellen

Signatur: **Actuator()**

Signatur: **void CallBitOutputPort(value:int)**

Beschreibung: Diese Methode wird den abgeleiteten Klassen als Schnittstelle zu den entsprechenden Methoden des Portbausteins zur Verfügung gestellt. Dadurch wird erreicht, daß die Attribute SectionNr, RoomNr und SensorName nur innerhalb der Basisklasse sichtbar sein muessen.

Signatur: **void CallIntOutputPort(value:int)**

Beschreibung: Diese Methode wird den abgeleiteten Klassen als Schnittstelle zu den entsprechenden Methoden des Portbausteins zur Verfügung gestellt. Dadurch wird erreicht, daß die Attribute SectionNr, RoomNr und SensorName nur innerhalb der Basisklasse sichtbar sein muessen.

2.3.2.1.4 Imports

Keine.

2.3.2.2 Klasse Benutzerdaten

2.3.2.2.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.2.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.2.3 Methodenschnittstellen

Signatur: **Benutzerdaten()**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Abstrakte Methode, wird in den Unterklassen konkretisiert.

Signatur: **int MinIntervall(Akzeptanzintervall:intervallType)**

Beschreibung: Liefert das Minimum des übergebenen Intevalls.

Signatur: **int MaxIntervall(Akzeptanzintervall:intervallType)**

Beschreibung: Liefert das Maximum des übergebenen Intervalls.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Abstrakte Methode zur Prüfung ob der aktuelle Benutzerwert in dem vom Facility Manager neu eigegebenen Akzeptanzintervall liegt.

2.3.2.2.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.3 Klasse Blind

2.3.2.3.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.3.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.3.3 Methodenschnittstellen

Signatur: **Blind(SectionNr:int, RoomNr:int, ActuatorName:char*)**

2.3.2.3.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- degreeType
- percentageType

2.3.2.4 Klasse BlindController

2.3.2.4.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.4.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
blindPtr	Blind

2.3.2.4.3 Methodenschnittstellen

Signatur: **BlindController(InitialSectionNr:int, InitialRoomNr:int)**

2.3.2.4.4 Imports

Importierte Methoden

Herkunft: Blind

- SetAngle
- SetPosition
- Stop
- MoveUp

Importierte Datentypen

Herkunft: SimpleTypes

- lockStatusType
- angleType
- percentageType

2.3.2.5 Klasse CentralHeatingSite

2.3.2.5.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.5.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.5.3 Methodenschnittstellen

2.3.2.5.4 Imports

Keine.

2.3.2.6 Klasse ClassDT2

2.3.2.6.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.6.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.6.3 Methodenschnittstellen

Signatur: **ClassDT2(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneEintrittNormalPtr:ClassZeitspanneEintrittNormal*)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzeptanzintervall liegt und ob Wert nicht kleiner ist als ZeitspanneEintrittNormal.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für DT2 im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.6.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.7 Klasse ClassDT2Standby

2.3.2.7.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
TempOpModusErmittlungPtr	TempOpModusErmittlung

2.3.2.7.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.7.3 Methodenschnittstellen

Signatur: **ClassDT2Standby(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzeptanzintervall liegt und ob Wert nicht kleiner ist als ZeitspanneEintrittStandbyNormal.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für DT2Standby im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.7.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.8 Klasse ClassZeitspanneAufheizenNormal

2.3.2.8.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
TempOpModusErmittlungPtr	TempOpModusErmittlung

2.3.2.8.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.8.3 Methodenschnittstellen

Signatur: **ClassZeitspanneAufheizenNormal(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneAufheizenStandbyNormalPtr:ClassZeitspanneAufheizenStandbyNormal*)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzektanzintervall liegt und ob Wert >= ZeitspanneEintrittNormal+5 und ob Wert > ZeitspanneAufheizenStandbyNormal+5.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für ZeitspanneAufheizenNormal im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.8.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.9 Klasse ClassZeitspanneAufheizenStandbyNormal

2.3.2.9.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
TempOpModusErmittlungPtr	TempOpModusErmittlung
ClassZeitspanneAufheizen-NormalPtr	ClassZeitspanneAufheizenNormal

2.3.2.9.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.9.3 Methodenschnittstellen

Signatur: **ClassZeitspanneAufheizenStandbyNormal(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneEintrittStandbyNormalPtr:ClassZeitspanneEintrittNormal*)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzektanzintervall liegt und ob Wert >= ZeitspanneEintrittStandbyNormal+5 und ob Wert < ZeitspanneAufheizenNormal-5.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für ZeitspanneAufheizenStandbyNormal im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.9.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.10 Klasse ClassZeitspanneEintrittNormal

2.3.2.10.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
ClassDT2Ptr	ClassDT2
ClassZeitspanneAufheizen-NormalPtr	ClassZeitspanneAufheizenNormal

2.3.2.10.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.10.3 Methodenschnittstellen

Signatur: **ClassZeitspanneEintrittNormal(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassZeitspanneAufheizenNormalPtr:ClassZeitspanneAufheizenNormal*)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzeptanzintervall liegt und ob Wert <= ZeitspanneAufheizenNormal-5 und ob Wert <= DT2.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für ZeitspanneEintrittNormal im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.10.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.11 Klasse ClassZeitspanneEintrittStandbyNormal

2.3.2.11.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
ClassZeitspanneAufheizen-StandbyNormalPtr	ClassZeitspanneAufheizenStandbyNormal
ClassDT2StandbyPtr	ClassDT2Standby

2.3.2.11.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.11.3 Methodenschnittstellen

Signatur: **ClassZeitspanneEintrittStandbyNormal(TempOpModusErmittlungPtr:TempOp-ModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType, ClassDT2StandbyPtr:ClassDT2Standby*)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzeptanzintervall liegt und ob Wert <= ZeitspanneAufheizenStandbyNormal-5 und ob Wert <= DT2Standby.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für ZeitspanneEintrittStandbyNormal im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.11.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.3.2.12 Klasse Clock

2.3.2.12.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.12.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.12.3 Methodenschnittstellen

2.3.2.12.4 Imports

Keine.

2.3.2.13 Klasse Contact

2.3.2.13.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.13.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.13.3 Methodenschnittstellen

Signatur: **Contact(SectionNr:int, RoomNr:int, SensorName:char*)**

2.3.2.13.4 Imports

Keine.

2.3.2.14 Klasse ControlPanel

2.3.2.14.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
BlindControllerPtr	BlindController

2.3.2.14.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.14.3 Methodenschnittstellen

Signatur: **ControlPanel(BlindControllerPtr:BlindController)**

2.3.2.14.4 Imports

Importierte Methoden

Herkunft: BlindController

- StopBlind
- MoveBlindUp

Herkunft: Room

- DeleteComfortTime
- DeleteWorkingDay
- DeleteStandbyTime
- SetWorkingDay
- SetComfortTime
- SetStandbyTime
- SetIntervall
- SetZeitspanneEintrittNormal
- SetZeitspanneVerlassenNormal

Herkunft: RoomTemperatureSensor

- SetOperationState

Herkunft: Temperatureregulator

- ZeitspanneAufheizenNormalSetzen
- AnwesenheitsTempNormalSetzen
- ZeitspanneAufheizenAufDefault
- AbwesenheitsTempAufDefault
- AnwesenheitsTempAufDefault

Herkunft: RoomLightRegulator

- userSetAmbientLightLevel
- userSetLightScene

Herkunft: PRTempController

- ZeitspanneEintrittNormalSetzen
- ZeitspanneVerlassenAufDefault
- ZeitspanneEintrittAufDefault

Importierte Datentypen

Herkunft: SimpleTypes

- malfunctionType

2.3.2.15 Klasse ControlPanelFm

2.3.2.15.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.15.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.15.3 Methodenschnittstellen

2.3.2.15.4 Imports

Importierte Methoden

Herkunft: RoomTemperatureSensor

- SetOperationState
- SetDefaultValue

Herkunft: Sensor

- SetDefaultValue

Importierte Datentypen

Herkunft: SimpleTypes

- lightSceneType
- intensityType
- malfunctionType

2.3.2.16 Klasse Day

2.3.2.16.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.16.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.16.3 Methodenschnittstellen

2.3.2.16.4 Imports

Keine.

2.3.2.17 Klasse DimmableLight

2.3.2.17.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.17.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.17.3 Methodenschnittstellen

Signatur: **DimmableLight(SectionNr:int, RoomNr:int, ActuatorName:char*)**

Beschreibung: Methode zum setzen der Lichtintensitaet, also der Stellung des Dimmers (0-100%).

2.3.2.17.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intensityType

2.3.2.18 Klasse DimmableLightController

2.3.2.18.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.18.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
dimmableLightPtr	DimmableLight

2.3.2.18.3 Methodenschnittstellen

Signatur: **DimmableLightController(SectionNr:int, RoomNr:int)**

2.3.2.18.4 Imports

Importierte Methoden

Herkunft: RoomLightRegulator

- pushButtonActivated

Herkunft: DimmableLight

- IsOn
- SetIntensity
- SwitchOff
- SwitchOn

2.3.2.19 Klasse DoorContact

2.3.2.19.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.19.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.19.3 Methodenschnittstellen

Signatur: **DoorContact(SectionNr:int, DoorNr:int, SensorName:char*)**

2.3.2.19.4 Imports

Keine.

2.3.2.20 Klasse HallwaySection

2.3.2.20.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.20.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
lightControllerPtr	LightController
lightControlUnitPtr	LightControlUnit
pRHallwayLightControl-lerPtr	PRHallwayLightController
presenceControllerHall-wayPtr	PresenceControllerHallway

2.3.2.20.3 Methodenschnittstellen

Signatur: **HallwaySection(InitialSectionNr:int, timerPtr:Timer*)**

Beschreibung: Konstruktor

2.3.2.20.4 Imports

Keine.

2.3.2.21 Klasse HallwaySector

2.3.2.21.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.21.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.21.3 Methodenschnittstellen

Signatur: **HallwaySector(SectionNr:int)**

Beschreibung: Konstruktor

2.3.2.21.4 Imports

Keine.

2.3.2.22 Klasse Heizperioden

2.3.2.22.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.22.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.22.3 Methodenschnittstellen

Signatur: **Heizperioden()**

2.3.2.22.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType
- yeardayType

2.3.2.23 Klasse House

2.3.2.23.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.23.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
outdoorLightSensorPtr1, outdoorLightSensorPtr2, outdoorLightSensorPtr3, outdoorLightSensorPtr4, outdoorLightSensorPtr5, outdoorLightSensorPtr6	OutdoorLightSensor
outdoorLightSensorPtr1, outdoorLightSensorPtr2, outdoorLightSensorPtr3, outdoorLightSensorPtr4, outdoorLightSensorPtr5, outdoorLightSensorPtr6	Heizperioden
section1Ptr, section2Ptr, section3Ptr	Section
controlPanelFMPtr	ControlPanelFm

2.3.2.23.3 Methodenschnittstellen

Signatur: **House(SectionZimmerListe:SList<double>, ZeitPtr:Timer*)**

Signatur:

Beschreibung: Ermittelt die benötigte Kesseltemperatur durch Abfrage aller Abschnitte und meldet diese an die Heizungsanlage.

2.3.2.23.4 Imports

Importierte Methoden

Herkunft: Heizperioden

- DeleteIntervall
- SetIntervall

Herkunft: Section

- SetAnwesenheitsTempIntervall
- SetAnwesenheitsTempDefault
- SetZeitspanneAufheizenIntervall

2.3.2.24 Klasse Jahresintervalle

2.3.2.24.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.24.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.24.3 Methodenschnittstellen

Signatur: **Jahresintervalle()**

2.3.2.24.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType
- yeardayType

2.3.2.25 Klasse Light

2.3.2.25.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.25.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
Pointername: StatusLinePtr	StatusLine

2.3.2.25.3 Methodenschnittstellen

Signatur: **Light(SectionNr:int, RoomNr:int, ActuatorName:char*)**

2.3.2.25.4 Imports

Keine.

2.3.2.26 Klasse LightControlUnit

2.3.2.26.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
lightControllerPtr	LightController

2.3.2.26.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.26.3 Methodenschnittstellen

Signatur: **LightControlUnit(lightControllerPtr:LightController*)**

Beschreibung: Konstruktor

2.3.2.26.4 Imports

Importierte Methoden

Herkunft: LightController

- AbfrageLichtStatus
- LichtAnsteuern

Importierte Datentypen

Herkunft: SimpleTypes

- zAnwesenheitType
- zLichtStatusType

2.3.2.27 Klasse LightController

2.3.2.27.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.27.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
lightPtr	Light

2.3.2.27.3 Methodenschnittstellen

Signatur: **LightController(SectionNr:int, InitialRoomNr:int)**

Beschreibung: Konstruktor

2.3.2.27.4 Imports

Importierte Methoden

Herkunft: Light

- IsOn
- SwitchOn
- SwitchOff

Herkunft: LightControlUnit

- LichtStatusGewechselt

Importierte Datentypen

Herkunft: SimpleTypes

- zLichtStatusType

2.3.2.28 Klasse OutdoorLightSensor

2.3.2.28.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.28.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.28.3 Methodenschnittstellen

Signatur: **OutdoorLightSensor()**

2.3.2.28.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intensityType

2.3.2.29 Klasse OutdoorTemperatureSensor

2.3.2.29.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.29.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.29.3 Methodenschnittstellen

Signatur: **OutdoorTemperatureSensor(SectionNr:int, RoomNr:int, SensorName:char*)**

2.3.2.29.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- tempType

2.3.2.30 Klasse PRBlindController

2.3.2.30.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
blindControllerPtr	BlindController
presenceControllerRoomPtr	PresenceControllerRoom

2.3.2.30.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.30.3 Methodenschnittstellen

Signatur: **PRBlindController(PresenceControllerRoomPtr:PresenceControllerRoom*, BlindControllerPtr:BlindController*)**

2.3.2.30.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- durationType

2.3.2.31 Klasse PRHallwayLightController

2.3.2.31.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
lightControlUnitPtr	LightControlUnit
presenceControllerHall-wayPtr	PresenceControllerHallway

2.3.2.31.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.31.3 Methodenschnittstellen

Signatur: **PRHallwayLightController(presenceControllerHallwayPtr:PresenceControllerHallway*, lightControlUnitPtr:LightControlUnit*)**

Beschreibung: Konstruktor

2.3.2.31.4 Imports

Importierte Methoden

Herkunft: LightControlUnit

- ZeitAbgelaufen
- AnwesenheitMelden

Herkunft: PresenceControllerHallway

- StatusAktualisieren

Importierte Datentypen

Herkunft: SimpleTypes

- zAnwesenheitZeitType
- zAnwesenheitType
- durationType

2.3.2.32 Klasse PRLightController

2.3.2.32.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
rLRegulatorPtr	RoomLightRegulator
presenceControllerRoomPtr	PresenceControllerRoom

2.3.2.32.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.32.3 Methodenschnittstellen

Signatur: **PRLightController(PresenceControllerRoomPtr:PresenceControllerRoom*, RoomLightRegulatorPtr:RoomLightRegulator*)**

2.3.2.32.4 Imports

Importierte Methoden

Herkunft: RoomLightRegulator

- activateDefaultLightScene
- updateUserPresence

Herkunft: PresenceControllerRoom

- StatusAktualisieren
- ErmittleAnwesenheit

2.3.2.33 Klasse PRTempController

2.3.2.33.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
temperatureregulatorPtr	Temperatureregulator
presenceControllerRoomPtr	PresenceControllerRoom

2.3.2.33.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.33.3 Methodenschnittstellen

Signatur: **PRTempController(PresenceControllerRoomPtr:PresenceControllerRoom*, TemperatureregulatorPtr:Temperatureregulator*)**

2.3.2.33.4 Imports

Importierte Methoden

Herkunft: PresenceController

- StatusAktualisieren

Herkunft: Temperatureregulator

- JemandDa
- anwesenheitSetzen

Herkunft: PresenceControllerRoom

- StatusAktualisieren

Importierte Datentypen

Herkunft: SimpleTypes

- durationType

2.3.2.34 Klasse Portbaustein

2.3.2.34.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.34.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.34.3 Methodenschnittstellen

Signatur: **Portbaustein()**

Signatur: **void CallBitOutputPort(SectionNr:int, RoomNr:int, ActuatorName:char*, value:int)**

Beschreibung:

Signatur: **int CallIntInputPort(SectionNr:int, RoomNr:int, SensorName:char*)**

Beschreibung:

Signatur: **void CallIntOutputPort(SectionNr:int, RoomNr:int, ActuatorName:char*, value:int)**

Beschreibung:

Signatur: **int CallBitInputPort(SectionNr:int, RoomNr:int, SensorName:char*)**

Beschreibung:

2.3.2.34.4 Imports

Keine.

2.3.2.35 Klasse PresenceController

2.3.2.35.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
timerPtr	Timer

2.3.2.35.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
	Contact

2.3.2.35.3 Methodenschnittstellen

Signatur: **PresenceController(InitialSectionNr:int, InitialRoomNr:int, ZeitPtr:Timer*)**

Signatur: **int ErmittleAnwesenheit()**

Beschreibung: Abstrakte Methode zur Abfrage der Anwesenheitskontakte.

Signatur: **void HalteEintrittZeitFest()**

Beschreibung: Merken des Zeitpunktes, wann das leere Zimmer von einer Person betreten wurde.

Signatur: **void HalteVerlassenZeitFest()**

Beschreibung: Merken des Zeitpunktes, wann die letzte Person das Zimmer verlassen hat.

2.3.2.35.4 Imports

Importierte Methoden

Herkunft: Contact

- AbfrageKontakt
- AbfrageZustand
- AfrageZustand

Importierte Datentypen

Herkunft: SimpleTypes

-

2.3.2.36 Klasse PresenceControllerHallway

2.3.2.36.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.36.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
contactPtr	Contact
doorContact1Ptr, doorContact2Ptr, doorContact3Ptr, doorContact4Ptr	DoorContact

2.3.2.36.3 Methodenschnittstellen

Signatur: **PresenceControllerHallway(InitialSectionNr:int, InitialRoomNr:int, timerPtr:Timer*)**

Beschreibung: Konstruktor

Signatur: **int ErmittleAnwesenheit()**

Beschreibung: Die von der Superklasse PresenceController geerbte Methode ErmittleAnwesenheit() ist in der Klasse PresenceControllerHallway überschrieben worden. In dieser überschriebenen Methode werden die Zustände der fünf Anwesenheitskontakte eines Flurabschnittes durch direkte Befragung dieser Kontakte ermittelt. Die fünf auf diese Weise erhaltenen Anwesenheitswerte werden durch ODER miteinander verknüpft. Das Ergebnis läßt eine eventuelle Belegung des Flurabschnitts erkennen.

2.3.2.36.4 Imports

Importierte Methoden

Herkunft: Contact

- AbfrageZustand

2.3.2.37 Klasse PresenceControllerRoom

2.3.2.37.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
timerPtr	Timer

2.3.2.37.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
contactPtr	Contact

2.3.2.37.3 Methodenschnittstellen

Signatur: **PresenceControllerRoom(InitialSectionNr:int, InitialRoomNr:int, TimerPtr:Timer*)**

2.3.2.37.4 Imports

Importierte Methoden

Herkunft: Contact

- AbfrageZustand
- ErmittleAnwesenheit

Importierte Datentypen

Herkunft: SimpleTypes

- zAnwesenheitZeitType

2.3.2.38 Klasse RadiatorController

2.3.2.38.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
roomTemperatureControl-lerPtr	RoomTemperatureController

2.3.2.38.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
waterTemperatureSensorPtr	WaterTemperatureSensor
radiatorValvePtr	RadiatorValve

2.3.2.38.3 Methodenschnittstellen

Signatur: **RadiatorController(InitialSectionNr:int, InitialRoomNr:int, RoomTemperatureControllerPtr:RoomTemperatureController*)**

Beschreibung: regelungsVentilEinstellung wird um den gegebenen Wert verändert.

Signatur: **void ventilFuerFrostschutzEinstellen()**

Beschreibung: RadiatorValve wird geöffnet, und auf einen Wert eingestellt, der sicherstellt, dass nichts einfriert.

Signatur: **void ventilFuerNormalbetriebEinstellen()**

Beschreibung: Es wird dafür gesorgt, daß der in regelungsVentilEinstellung gespeicherte Wert an RadiatorValve geschickt wird.

Signatur: **tempType getWaterTemperature()**

Beschreibung: Gibt den kleinsten Wert der 3 Instanzen von WaterTemperatureSensor zurück, falls eine Temperatur kleiner als 4 Grad ist, sonst wird der arithmetische Mittelwert der drei Wassertemperaturen berechnet.

2.3.2.38.4 Imports

Importierte Methoden

Herkunft: WaterTemperatureSensor

- GetTemperature

Herkunft: RadiatorValve

- SetPosition

Herkunft: RoomTemperatureController

- abfrageMinimalZimmertemperatur

Importierte Datentypen

Herkunft: SimpleTypes

- tempType, percentageType

2.3.2.39 Klasse RadiatorValve

2.3.2.39.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.39.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.39.3 Methodenschnittstellen

Signatur: **RadiatorValve(SectionNr:int, RoomNr:int, ActuatorName:char*)**

2.3.2.39.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- percentageType

2.3.2.40 Klasse Room

2.3.2.40.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.40.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
radiatorValvePtr	TempOpModusErmittlung
ControlPanelPtr	ControlPanel
pRLightControllerPtr	PRLightController
PRTempControllerPtr	PRTempController
PRTempControllerPtr	Tagesintervalle
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	DimmableLightController
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	Temperaturdaten
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	Wochenintervalle
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	Urlaubszeiten
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ZeitspannenVerlassen
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassDT2

Pointername	Klassenname des aggregierten Objektes
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassZeitspanneEintrittNormal
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassZeitspanneEintrittStandbyNormal
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassZeitspanneAufheizenNormal
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassZeitspanneAufheizenStandbyNormal
dimmableLightControllerPtr 1, dimmableLightControllerPtr 2	ClassDT2Standby
presenceControllerRoomPtr	PresenceControllerRoom

2.3.2.40.3 Methodenschnittstellen

Signatur: **Room(InitialSectionNr:int, InitialRoomNr:int, Flaeche:double, ZeitPtr:Timer*)**

2.3.2.40.4 Imports

Importierte Methoden

Herkunft: Tagesintervalle

- DeleteStandbyTime
- DeleteComfortTime
- SetStandbyTime
- SetComfortTime

Herkunft: Temperaturdaten

- SetWert
- SetWertl
- SetAkzeptanzintervall
- SetDefaultwert

Herkunft: Wochenintervalle

- DeleteWorkingDay
- SetWorkingDay

Herkunft: Urlaubszeiten

- SetIntervall

Herkunft: ZeitspannenVerlassen

- SetWert

Herkunft: ClassZeitspanneEintrittNormal

- SetWert

2.3.2.41 Klasse RoomLightController

2.3.2.41.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
oLSPtrs[6]	OutdoorLightSensor
mtRoomPtr	Room

2.3.2.41.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.41.3 Methodenschnittstellen

Signatur: **RoomLightController(RoomPtr:Room*)**

2.3.2.41.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intensityType
- sensorType

2.3.2.42 Klasse RoomLightControllerWithBlinds

2.3.2.42.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
blindControllerPtr	BlindController

2.3.2.42.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.42.3 Methodenschnittstellen

2.3.2.42.4 Imports

Importierte Methoden

Herkunft: BlindController

- GetBlindAngle
- GetBlindPosition
- calculateOptimalBlindPos
- calculateOptimalBlindAngle

Herkunft: OutdoorLightSensor

- GetIntensity

Importierte Datentypen

Herkunft: SimpleTypes

- percentageType
- intensityType
- degreeType

2.3.2.43 Klasse RoomLightControllerWithoutBlinds

2.3.2.43.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.43.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.43.3 Methodenschnittstellen

Signatur: **RoomLightControllerWithoutBlinds()**

2.3.2.43.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intensityType
- sensorType

2.3.2.44 Klasse RoomLightRegulator

2.3.2.44.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
rLControllerPtr	RoomLightController
dimmableLightControllerPtr	DimmableLightController

2.3.2.44.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.44.3 Methodenschnittstellen

Signatur: **RoomLightRegulator(DimmableLightControllerPtr1:DimmableLightController*, DimmableLightControllerPtr2:DimmableLightController*, RoomLightControllerPtr:RoomLightController*)**

2.3.2.44.4 Imports

Importierte Methoden

Herkunft: DimmableLightController

- dim
- switchLight

Importierte Datentypen

Herkunft: SimpleTypes

- intensityType
- percentageType
- degreeType

2.3.2.45 Klasse RoomLightRegulatorWithBlinds

2.3.2.45.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
blindControllerPtr	BlindController

2.3.2.45.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.45.3 Methodenschnittstellen

Signatur: **RoomLightRegulatorWithBlinds(DimmableLightControllerPtr1:DimmableLightController*, DimmableLightControllerPtr2:DimmableLightController*, RoomLightControllerPtr:RoomLightController*, BlindControllerPtr:BlindController*)**

2.3.2.45.4 Imports

Importierte Methoden

Herkunft: BlindController

- BlindEinstellungLichtEnabled
- BlindEinstellungLicht

Herkunft: DimmableLightController

- dim

Herkunft: RoomLightControllerWithBlinds

- calculateOptimalBlindAngle
- BlindEinstellungLicht
- calculateOptimalBlindPos
- GetBlindPosition

Importierte Datentypen

Herkunft: SimpleTypes

- percentageType
- angleType

2.3.2.46 Klasse RoomLightRegulatorWithoutBlinds

2.3.2.46.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.46.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.46.3 Methodenschnittstellen

2.3.2.46.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- percentageType
- angleType

2.3.2.47 Klasse RoomTemperatureController

2.3.2.47.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.47.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
roomTemperatureSensorPtr	RoomTemperatureSensor

2.3.2.47.3 Methodenschnittstellen

Signatur: **RoomTemperatureController(InitialSectionNr:int, InitialRoomNr:int)**

2.3.2.47.4 Imports

Importierte Methoden

Herkunft: RoomTemperatureSensor

- GetTemperature
- GetTEmperature

Importierte Datentypen

Herkunft: SimpleTypes

- tempType

2.3.2.48 Klasse RoomTemperatureSensor

2.3.2.48.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.48.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.48.3 Methodenschnittstellen

Signatur: **RoomTemperatureSensor(SectionNr:int, RoomNr:int, SensorName:char*)**

2.3.2.48.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- tempType

2.3.2.49 Klasse RoomWithBlinds

2.3.2.49.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.49.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
blindControllerPtr	BlindController
PRBlindControllerPtr	PRBlindController
TemperatureregulatorWith-BlindsPtr	TemperatureregulatorWithBlinds
valueInterfaceWithBlindsPtr	ValueInterfaceWithBlinds
rLRegulatorWithBlindsPtr	RoomLightRegulatorWithBlinds
rLControllerWithBlindsPtr	RoomLightControllerWithBlinds

2.3.2.49.3 Methodenschnittstellen

Signatur: **RoomWithBlinds(sectionNr:int, roomNr:int)**

2.3.2.49.4 Imports

Keine.

2.3.2.50 Klasse RoomWithoutBlinds

2.3.2.50.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.50.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
valueInterfaceWithout-BlindsPtr	ValueInterfaceWithoutBlinds
temperatureregulatorWit-houtBlindsPtr	TemperatureregulatorWithoutBlinds
rLRegulatorWithout-BlindsPtr	RoomLightRegulatorWithoutBlinds
rLControllerWithout-BlindsPtr	RoomLightControllerWithoutBlinds

2.3.2.50.3 Methodenschnittstellen

Signatur: **RoomWithoutBlinds()**

2.3.2.50.4 Imports

Keine.

2.3.2.51 Klasse ScheduledFct

2.3.2.51.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.51.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.51.3 Methodenschnittstellen

2.3.2.51.4 Imports

Keine.

2.3.2.52 Klasse Scheduler

2.3.2.52.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.52.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.52.3 Methodenschnittstellen

2.3.2.52.4 Imports

Keine.

2.3.2.53 Klasse Section

2.3.2.53.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.53.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
Pointername: RoomPtr	Room
hallwaySectionPtr	HallwaySection

2.3.2.53.3 Methodenschnittstellen

Signatur: **Section(InitialSectionNr:int, ZimmerFlaechenListe:SList<double>, ZeitPtr:Timer*)**

2.3.2.53.4 Imports

Importierte Methoden

Herkunft: Room

- SetAnwesenheitsTempIntervall
- SetAnwesenheitsTempDefault
- SetZeitspanneAufheizenIntervall

2.3.2.54 Klasse Sensor

2.3.2.54.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.54.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.54.3 Methodenschnittstellen

Signatur: **Sensor()**

Signatur: **int GetDefaultValue()**

Beschreibung: Die Methode GetDefaultValue liefert den aktuellen DefaultValue des Sensors zurueck.

Signatur: **int SendEmail()**

Beschreibung: Diese Methode wird nur intern zum Versenden von Emails an den Facility Manager verwendet.

Signatur: **int CallBitInputPort(void)**

Beschreibung: Diese Methode wird den abgeleiteten Klassen als Schnittstelle zu den entsprechenden Methoden Portbausteins zur Verfügung gestellt. Dadurch wird erreicht, daß die Attribute SectionNr, RoomNr und SensorName nur innerhalb der Basisklasse sichtbar sein muessen.

Signatur: **int CallIntInputPort(void)**

Beschreibung: Diese Methode wird den abgeleiteten Klassen als Schnittstelle zu den entsprechenden Methoden des Portbausteins zur Verfügung gestellt. Dadurch wird erreicht, daß die Attribute SectionNr, RoomNr und SensorName nur innerhalb der Basisklasse sichtbar sein muessen.

2.3.2.54.4 Imports

Keine.

2.3.2.55 Klasse StatusLine

2.3.2.55.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.55.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.55.3 Methodenschnittstellen

Signatur: **StatusLine(SectionNr:int, RoomNr:int, SensorName:char*)**

2.3.2.55.4 Imports

Keine.

2.3.2.56 Klasse Tagesintervalle

2.3.2.56.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.56.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.56.3 Methodenschnittstellen

Signatur: **Tagesintervalle()**

2.3.2.56.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType
- minType

2.3.2.57 Klasse TempOpModusErmittlung

2.3.2.57.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
HeizperiodenPtr	Heizperioden
ValueInterfacePtr	ValueInterface
TagesintervallePtr	Tagesintervalle
TemperaturdatenPtr	Temperaturdaten
WochenintervallePtr	Wochenintervalle
UrlaubszeitenPtr	Urlaubszeiten
TimerPtr	Timer
ClassDT2Ptr	ClassDT2
ClassZeitspanneEintrittNormal	ClassZeitspanneEintrittNormal
ClassZeitspanneEintrittStandbyNormalPtr	ClassZeitspanneEintrittStandbyNormal

2.3.2.57.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.57.3 Methodenschnittstellen

Signatur: **TempOpModusErmittlung(WochenintervallePtr:Wochenintervalle*, UrlaubszeitenPtr:Urlaubszeiten*, HeizperiodenPtr:Heizperioden*, TagesintervallePtr:Tagesintervalle*, InitialAktuellerModus:modeType)**

2.3.2.57.4 Imports

Importierte Methoden

Herkunft: ValueInterface

- SetValues

Importierte Datentypen

Herkunft: SimpleTypes

- modeType

2.3.2.58 Klasse Temperaturdaten

2.3.2.58.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.58.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.58.3 Methodenschnittstellen

Signatur: **Temperaturdaten(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Überprüft ob Wert im Akzeptanzintervall liegt.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für eine Temperatur im neu eingegebenen Akzeptanzintervall für diese Temperatur enthalten ist.

2.3.2.58.4 Imports

Importierte Methoden

Herkunft: TempOpModusErmittlung

- UpdateWerte

2.3.2.59 Klasse Temperatureregulator

2.3.2.59.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.59.2 Aggregationen von Objekten anderer Klassen

Pointername	Klassenname des aggregierten Objektes
radiatorControllerPtr	RadiatorController
roomTemperatureControllerPtr	RoomTemperatureController

2.3.2.59.3 Methodenschnittstellen

Signatur: **Temperatureregulator(InitialSectionNr:int, InitialRoomNr:int)**

Signatur: **void setzeZieltemperatur()**

Beschreibung: Bei Änderung der Anwesenheit oder des Operationsmodus setzt diese Methode die Werte minTemp, maxTemp und ZielTemp neu.

2.3.2.59.4 Imports

Importierte Methoden

Herkunft: RadiatorController

- ventilOeffnen
- ventilSchliessen

Herkunft: RoomTemperatureController

- abfrageMittwelwertZimmertemperatur
- abfrageMittelwertZimertemperatur
- abfrageMittelwertZimmertemperatur

Importierte Datentypen

Herkunft: SimpleTypes

- tempType, modeType, anwesenheitType

2.3.2.60 Klasse TemperatureregulatorWithBlinds

2.3.2.60.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
blindControllerPtr	BlindController
roomLightControllerWith-BlindsPtr	RoomLightControllerWithBlinds

2.3.2.60.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.60.3 Methodenschnittstellen

Signatur: **TemperatureregulatorWithBlinds(InitialSectioNr:int, InitialRoomNr:int, BlindControllerPtr:BlindController*, RoomLightControllerWithBlindsPtr:RoomLightControllerWithBlinds*)**

2.3.2.60.4 Imports

Importierte Methoden

Herkunft: BlindController

- blindEinstellungTemp

Herkunft: RoomLightController

- GetIntensityLevelAct
- GetIntensityLevelBlindsUp
- GetIntensityLevelBlindsDown

Herkunft: RoomTemperatureController

- abfrageMittelwertZimmertemperatur

Herkunft: RoomLightControllerWithBlinds

- GetIntensityLevelBlindsDown
- GetIntensityLevelBlindsUp
- GetRoomLightIntensity
- GetTemperature

2.3.2.61 Klasse TemperatureregulatorWithoutBlinds

2.3.2.61.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.61.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.61.3 Methodenschnittstellen

2.3.2.61.4 Imports

Keine.

2.3.2.62 Klasse Time

2.3.2.62.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.62.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.62.3 Methodenschnittstellen

2.3.2.62.4 Imports

Keine.

2.3.2.63 Klasse Timer

2.3.2.63.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.63.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.63.3 Methodenschnittstellen

2.3.2.63.4 Imports

Keine.

2.3.2.64 Klasse Urlaubszeiten

2.3.2.64.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.64.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.64.3 Methodenschnittstellen

Signatur: **Urlaubszeiten()**

2.3.2.64.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType
- yeardayType

2.3.2.65 Klasse ValueInterface

2.3.2.65.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
temperatureregulatorPtr	Temperatureregulator
	PRTempController

2.3.2.65.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.65.3 Methodenschnittstellen

Signatur: **ValueInterface(temperatureregulatorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*)**

2.3.2.65.4 Imports

Keine.

2.3.2.66 Klasse ValueInterfaceWithBlinds

2.3.2.66.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
	PRBlindController

2.3.2.66.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.66.3 Methodenschnittstellen

Signatur: **ValueInterfaceWithBlinds(temperatureregulatorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*, pRBlindControllerPtr:PRBlindController*)**

2.3.2.66.4 Imports

Keine.

2.3.2.67 Klasse ValueInterfaceWithoutBlinds

2.3.2.67.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.67.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.67.3 Methodenschnittstellen

Signatur: **ValueInterfaceWithoutBlinds(temperatureregualtorPtr:Temperatureregulator*, pRTempControllerPtr:PRTempController*)**

2.3.2.67.4 Imports

Keine.

2.3.2.68 Klasse WaterTemperatureSensor

2.3.2.68.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.68.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.68.3 Methodenschnittstellen

Signatur: **WaterTemperatureSensor(SectionNr:int, RoomNr:int, SensorName:char*)**

2.3.2.68.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- tempType

2.3.2.69 Klasse Wochenintervalle

2.3.2.69.1 Assoziationen zu Objekten anderer Klassen

Keine

2.3.2.69.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.69.3 Methodenschnittstellen

Signatur: **Wochenintervalle()**

2.3.2.69.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType
- weekdayType

2.3.2.70 Klasse ZeitspannenVerlassen

2.3.2.70.1 Assoziationen zu Objekten anderer Klassen

Pointername	Klassenname des referenzierten Objektes
TempOpModusErmittlungPtr	TempOpModusErmittlung

2.3.2.70.2 Aggregationen von Objekten anderer Klassen

Keine

2.3.2.70.3 Methodenschnittstellen

Signatur: **ZeitspannenVerlassen(TempOpModusErmittlungPtr:TempOpModusErmittlung*, InitialDefaultwert:int, InitialWert:int, InitialAkzeptanzintervall:intervallType)**

Signatur: **void Check(Eingabe:int)**

Beschreibung: Prüft ob Wert im Akzeptanzintervall liegt.

Signatur: **void CheckIntervall(Eingabeintervall:intervallType)**

Beschreibung: Diese Methode wird von der Methode SetAkzeptanzintervall() aufgerufen und prüft, ob der derzeitig eingestellte Benutzerwert für ZeitspanneVerlassenNormal bzw. ZeitspanneVerlassenStandbyNormal im neu eingegebenen Akzeptanzintervall für diese Zeitspanne enthalten ist.

2.3.2.70.4 Imports

Importierte Datentypen

Herkunft: SimpleTypes

- intervallType

2.4 Verifikation

In diesem Abschnitt wird der Systementwurf verifiziert. In Kapitel 2.4.1 wird angegeben, welche Aspekte bei der Verifikation zu beachten sind. Die Ergebnisse der Verifikation werden ab Kapitel 1.4.4 festgehalten.

2.4.1 Verifikations-Checklisten

In diesem Abschnitt sind die Aspekte aufgelistet, die bei der Verifikation der einzelnen Dokumente zu beachten sind.

Überprüfe jeden Use Case im Use Case Diagramm auf:

Einhaltung der Dokumentationsrichtlinien:

1. Wurde die UML-Syntax eingehalten?
2. Existiert ein Akteur für den Use Case?
3. Existiert zu dem Use Case eine informelle Beschreibung?

Interne Konsistenz der Modelle:

1. *Wenn ein Use Case zu einem anderen in Beziehung steht:* Findet sich die Beziehung auch in den Szenarien, die zu dem Use Case gehören?
2. Sind die Szenarien, die zum Use Case gehören, in der informellen Beschreibung dokumentiert?

Konsistenz mit den informellen B-Anforderungen/Referenzdokumenten

1. *Wenn im Use Case ein Aktuator benutzt wird:* Ist der Aktuator in der Gebäudearchitektur beschrieben?
2. Ist der Akteur des Use Cases in der Gebäudearchitektur / den informellen B-Anforderungen beschrieben?
3. Gibt der Use Case das in den B-Anforderungen beschriebene Verhalten wider?

Überprüfe jede Beschreibung zum Use Case Diagramm:

Einhaltung der Dokumentationsrichtlinien:

1. Existiert ein Use Case zur Beschreibung?
2. Ist der Akteur des Use Cases beschrieben?
3. Ist die Beschreibung verständlich?

Überprüfe jedes Szenario auf:

Einhaltung der Dokumentationsrichtlinien:

1. Sind alle für das Verhalten des Systems wichtige Ausgangsvariablen im Szenario beschrieben?
2. Enthält das Szenario eine informelle Beschreibung?
3. Endet das Szenario mit dem Verhalten/Nichtverhalten des Systems (Verhalten könnte z.B. das Setzen von Werten oder die Ansteuerung eines Aktuators sein)?
4. Sind alle Schritte im Szenario notwendig?
5. Sind alle wichtigen Schritte im Szenario beschrieben?
6. *Wenn das Szenario auf einen Use Case verweist:* Setzen die Szenarien des Use Case das betrachtete Szenario fort?

Interne Konsistenz der Modelle:

1. Ist angegeben, auf welches Use Case sich das Szenario bezieht?
2. Existiert ein Sequenzdiagramm für das Szenario?
3. Lassen sich alle Schritte des Szenarios auf die Kommunikation zwischen Objekten in einem Sequenzdiagramm abbilden?

4. *Wenn das Szenario auf einen weiteren Use Case verweist:* Existiert im zugehörigen Sequenzdiagramm eine Notiz mit Verweis auf das Use Case?

Konsistenz mit den informellen B-Anforderungen/Referenzdokumenten:

1. Werden im Szenario die zugehörigen B-Anforderungen referenziert?
2. Gibt das Szenario die B-Anforderungen korrekt wieder?

Überprüfe jedes Teilsystem im Teilsystemdiagramm auf:

Einhaltung der Dokumentationsrichtlinien:

1. Wurde die UML-Syntax eingehalten?
2. Existiert zum Teilsystem eine Beschreibung?
3. Ist die Beschreibung des Teilsystems verständlich?

Interne Konsistenz der Modelle:

1. Existiert für jedes Teilsystem ein Klassendiagramm?
2. *Wenn eine Beziehung von einem Teilsystem zu einem anderen Teilsystem besteht:* Existiert eine Beziehung zwischen Klassen der verschiedenen Teilsysteme?

Überprüfe jede Klasse im Klassendiagramm auf:

Einhaltung der Dokumentationsrichtlinien:

1. Wurde die UML-Syntax eingehalten?
2. Hat die Klasse Attribute und Methoden?
3. Hat die Klasse Beziehungen zu anderen Klassen (im selben oder in anderen Teilsystemen)?
4. Werden die beschriebenen Attribute und Methoden für die Funktionalität des Teilsystems benötigt?
5. Ist die Klasse informell beschrieben?
6. Sind alle Aggregationen der Klasse informell beschrieben?
7. Sind alle Attribute und Methoden der Klasse informell beschrieben?
8. Sind die angegebenen Beschreibungen verständlich?

Interne Konsistenz der Modelle:

1. *Wenn die Klasse dynamisches Verhalten besitzt:* Existiert ein Statediagramm für die Klasse?
2. Werden die Assoziationen der Klasse zu anderen Klassen im Statediagramm verwendet?
3. Werden die Methoden der Klasse im Statediagramm verwendet?
4. Werden die Assoziationen der Klasse zu anderen Klassen in den Sequenzdiagrammen verwendet?
5. Werden die Methoden der Klasse in den Sequenzdiagrammen verwendet?

Konsistenz zu den informellen B-Anforderungen/Referenzdokumenten:

1. *Wenn die Klasse eine Strukturklasse ist:* Gibt es die Struktur in der Gebäudearchitekturbeschreibung?
2. *Wenn die Klasse eine Sensor/Aktuator-Klasse ist:* Gibt es den Sensor/Aktuator in der Gebäudearchitekturbeschreibung?
3. *Wenn die Klasse eine Regelungsklasse ist:* Ist die Klasse notwendig, um die in den informellen B-Anforderungen beschriebene Funktionalität zu erbringen?

Überprüfe jedes Sequenzdiagramm auf:

Einhaltung der Dokumentationsrichtlinien:

1. Wurde die UML-Syntax eingehalten?
2. Gibt es einen Akteur (<<actor>>)?
3. *Wenn mehrere Objekte einer Klasse (auch in anderen Sequenzdiagrammen) existieren:* Werden die Objekte über einen eigenen Namen referenziert?
4. *Wenn Objekte über einen eigenen Namen referenziert werden:* Existiert eine informelle

Beschreibung für jedes Objekt?
5. *Wenn eine informelle Beschreibung für ein Objekt existiert:* Ist die Beschreibung verständlich?

Interne Konsistenz der Modelle:

1. *Wenn ein Objekt einer Klasse mit einem anderen Objekt kommuniziert:* Gibt es eine Beziehung zwischen den Klassen im Klassendiagramm?
2. Existiert für das Sequenzdiagramm ein Szenario?
3. Ist dokumentiert, auf welches Szenario sich das Sequenzdiagramm bezieht?
4. Ist die Information aus dem Szenario korrekt wiedergegeben?
5. *Wenn es im Sequenzdiagramm eine Notiz mit einem Verweis auf ein Use Case gibt:* Wird im Szenario ebenfalls auf das Use Case verwiesen?
6. *Wenn ein Objekt einer Klasse Nachrichten (Methodenaufruf) geschickt bekommt:* Sind die Methoden im Statediagramm der Klasse als Events modelliert?
7. *Wenn ein Objekt einer Klasse Nachrichten (Methodenaufruf) geschickt bekommt:* Sind die Methoden im Klassendiagramm als Methoden der Klasse definiert?
8. *Wenn ein Objekt einer Klasse Nachrichten (Methodenaufruf) verschickt:* Sind die Methoden in den Statediagrammen als Aktionen modelliert?

Überprüfe jedes Kollaborationsdiagramm auf:

Einhaltung der Dokumentationsrichtlinien:

- nicht notwendig, da generiert

Interne Konsistenz der Modelle:

- Existiert für jedes Sequenzdiagramm ein Kollaborationsdiagramm?

Überprüfe jedes Statediagramm auf:

Einhaltung der Dokumentationsrichtlinien:

1. Wurde die UML-Syntax eingehalten?
2. Besitzt das Statediagramm einen Anfangszustand?
3. Wurden Verfeinerungen von Zuständen in einem eigenen Statediagramm umgesetzt?
4. Wurden Entry- und Exit-Aktionen und Do-Aktivitäten mit Hilfe der “State Table” beschrieben, d.h. sind sie in der Dokumentation beschrieben?
5. Gibt es keine Events (Methodenaufrufe) am Zustandsübergang, die mit einem UND-Operator verknüpft sind?
6. Kann in einem Zustand längere Zeit verbracht werden?
7. Kann jeder Zustand des Statediagramms erreicht werden?
8. Kann jeder Zustandsübergang durchlaufen werden?

Interne Konsistenz der Modelle

1. *Wenn eine Methode einer anderen Klasse aufgerufen wird:* Existiert eine Beziehung zwischen den Klassen im Klassendiagramm?
2. *Wenn eine Methode einer anderen Klasse aufgerufen wird:* Existiert die Methode der Klasse im Klassendiagramm?
3. Sind alle Events im Statediagramm in den Sequenzdiagrammen als Methoden modelliert, die die Klasse geschickt bekommt?
4. Sind alle Aktionen im Statediagramm in den Sequenzdiagrammen als Methoden modelliert, die die Klasse verschickt?
5. Sind alle global verwendeten Variablen im Statediagramm im Klassendiagramm als Attribut vorhanden?
6. *Wenn ein Event mit einer Bedingung verknüpft ist:* Ist die Bedingung in den Sequenzdiagrammen dokumentiert?

Überprüfe jede informelle B-Anforderung auf:

1. Ist die B-Anforderung mindestens in einem Szenario referenziert?

2. Ist die B-Anforderung wirklich überdeckt durch die zugehörigen Szenarien d.h., ist jede Bedingung und jeder Teilaspekt der B-Anforderung überdeckt?
3. Werden alle Aktuatoren (beschrieben in der Gebäudearchitektur) im Use Case Diagramm verwendet?
4. Gibt das Klassendiagramm die Gebäudearchitektur wieder?

Überprüfe die Verfolgbarkeitsmatrix auf:

1. Sind alle B-Anforderungen und alle Klassen des Klassendiagramms berücksichtigt worden?
2. Gibt es B-Anforderungen, die durch keine Klasse umgesetzt werden?
3. Gibt es Klassen, die für die Umsetzung der B-Anforderungen nicht benötigt werden?
4. Sind die in der Tabelle dargestellten Verknüpfungen korrekt, d.h.:
   - fehlt evtl. eine Verknüpfung?

ist eine angegebene Verknüpfung falsch?

2.4.2 Klassifikation von Fehlern

Die während der Verifikation aufgedeckten Fehler werden den folgenden Fehlerklassen zugeordnet. Um die Fehler zu klassifizieren, bitte die Fehlerklassen in der angegebenen Reihenfolge durchgehen:

Unvollständigkeit (U):

Es fehlen Informationen zu einem Objekt. Beispielsweise ist eine Klasse nicht im Data-Dictionary beschrieben, obwohl sie im Klassendiagramm vorhanden ist.

Falsche Information (F):

Eine gegebene Information zu einem Objekt ist offensichtlich falsch. Beispielsweise kann ein Zustand im Statediagramm der Klasse Fenster nie erreicht werden.

Überspezifikation und irrelevante Information (R):

Eine gegebene Information ist überflüssig, da sie nicht benötigt wird. Beispielsweise existiert eine Klasse Kesseltemperatur im Klassendiagramm. Diese wird aber für die erwartete Systemfunktionalität nicht benötigt.

Inkonsistenzen oder Widersprüche (I):

Eine gegebene Information zu einem Objekt ist inkonsistent oder widersprüchlich in bezug auf eine andere Information zu diesem Objekt. Beispielsweise ist im Klassendiagramm eine Methode einer Klasse definiert, die im Statediagramm mit ähnlichem (aber nicht gleichem) Bezeichner auftaucht.

Vorwärtsreferenzen und sonstige Fehler (S):

Alle Fehler, die sich nicht den anderen Fehlerklassen zuordnen lassen.

2.4.3 Klassifikation von Änderungsaktivitäten beim Rework

Die während der Verifikation aufgedeckten Fehler werden beim Rework behoben. Dabei sollt Ihr beschreiben, welche Arten von Änderungsaktivitäten Ihr durchgeführt habt. Mehrfachnennungen sind erlaubt.

Löschen (L):

Zur Korrektur des Fehlers mußtet Ihr ein existierendes Element (Klasse, Methode, Zustand, etc.) entfernen.

Ändern (A):

Zur Korrektur des Fehlers mußtet Ihr ein existierendes Element (Klasse, Methode, Zustand, etc.) verändern.

Hinzufügen (H):

Zur Korrektur des Fehlers mußtet Ihr ein neues Element (Klasse, Methode, Zustand, etc.) einfügen.