Programmierrichtlinien für C++

Die im folgenden beschriebenen Programmierrichtlinien sind für das SE-I-Praktikum verbindlich und müssen eingehalten werden.

1 Dateien

Für jede zu kodierende Klasse werden 2 Dateien angelegt, die Header(.hh)-Datei und die Code(.cc)-Datei. Die Dateien tragen den Namen der Klasse erweitert um die jeweilige Endung (.hh oder .cc). Die Zeilen in beiden Dateien sind maximal 80 Zeichen lang. Als Beispiel für die Gestaltung von Header- und Code-Datei dienen Any-Class.hh und AnyClass.cc.

Aufbau der Header-Datei

- Die Header-Datei beginnt immer mit den beiden Statements:

  #ifndef _KLASSE_H

  #define _KLASSE_H

  und endet mit der Zeile:
  #endif

  Dabei ist KLASSE durch den jeweiligen Klassennamen zu ersetzen.
- Im Kopf der Datei steht (als Kommentar) der Autor, das Erzeugungsdatum und die Dateihistorie. Die Einträge werden von RCS automatisch generiert.
- Danach kommen alle include-Statements, die benötigt werden.
- Unter Umständen folgt die Deklaration der Parametertypen.
- Darauf folgt die Klassendeklaration.
- Zu jeder Methode wird die Spezifikation aus dem Systementwurf und den Komponenten-Anforderungen als Kommentar angegeben.
- Inline-Code ist nur in Treibern und Stubs erlaubt.

Aufbau der Code-Datei

- Im Kopf der Datei steht wie bei der Header-Datei (als Kommentar) der Autor, das Erzeugungsdatum und die Dateihistorie.
- Danach wird die Datei <Klasse.hh> inkludiert:

  #include <Klasse.hh>

  Dabei ist Klasse durch den jeweiligen Klassennamen zu ersetzen.
- Daran schließt sich der Quellcode für die einzelnen Methoden an.
- Die einzelnen Methoden werden durch eine Kommentarzeile getrennt.

2 Code

- Es müssen alle Methoden aus dem Systementwurf und den Komponenten-Anforderungen und der Default-Konstruktor kodiert werden.
- In Klassen mit Bezügen auf andere Klassen muß ein expliziter Destruktor definiert sein oder ein Kommentar in der Klassendefinition stehen, warum der Default-Destruktor an dieser Stelle korrekt ist.

Praktikum Software Engineering I SS 99

- Alle Methoden aus dem Systementwurf, alle Konstruktoren und Destruktoren sind *public*.
- Alle anderen Methoden und alle Attribute sind *private*.
- Globale Variablen sind nicht erlaubt.
- Auf das Konstrukt “extern” ist zu verzichten.
- Variablen sollen nur am Anfang eines Blocks deklariert und initialisiert werden.
- In jeder Zeile steht nur eine Anweisung.
- Es sollen keine *type casts* verwendet werden.
- Nur für Zähler sollen *for*-Schleifen verwendet werden. Für alles andere gibt es *while*.
- Die Verwendung des Konstrukts “goto” ist nicht erlaubt.
- Die Verwendung von #define ist zu vermeiden (es sei denn zur Vermeidung mehrfacher Includes)
- Einrückungen:

  while (...) {

  }
- Einrückungstiefe ist mindestens 2.
- Schleifenausgänge:
  - kein *continue*
  - *break* nur bei *switch*
- Allgemeine Datentypen werden in die Datei: SimpleTypes.hh aufgenommen.

3 Kommentare *(aus W. Humphrey, „A Discipline For Software Engineering“)*

- Dokumentiere den Kode zum Verständnis eines Lesers
- Kommentare sollten Zweck und Verhalten erklären
- Variablendeklarationen sollten in einem Kommentar motiviert werden.

Beispiel:

guter Kommentar:

if (recordCount > limit) // *have all records been processed?*

schlechter Kommentar:

if (recordCount > limit) // *check if recordCount greater*

Beispiel: Header-Datei der Klasse AnyClass

```
#ifndef _ANYCLASS_H
#define _ANYCLASS_H

//--------------------------------------------------------------------------
//
// Filename:        $RCSfile$
// Author:          $Author$
// State:           $State$
// Revision:        $Revision$
// Release:         $Name$
//
//--------------------------------History----------------------------------
//
// $Log$
//
// -------------------------------------------------------------------------

#include <AnyOtherClass.hh>

class AnyClass {
  private:

    // Hier stehen alle Attribute und die Methoden, die nach aussen nicht
    // sichtbar sind. Jedes Attribut und jede Methode bekommt als
    // Kommentar eine Kurzspezifikation hinzugefuegt.

  public:

    // Hier stehen die Methoden, die nach aussen sichtbar sind. Jede
    // Methode bekommt als Kommentar ihre Spezifikation hinzugefuegt.

     AnyClass(); // Konstruktor
    ~AnyClass(); // Destruktor
};

#endif
```


Praktikum Software Engineering I SS 99

Beispiel: Code-Datei der Klasse AnyClass

```
//--------------------------------------------------------------------------
//
// Filename:        $RCSfile$
// Author:          $Author$
// State:           $State$
// Revision:        $Revision$
// Release:         $Name$
//
//--------------------------------History----------------------------------
//
// $Log$
//
// -------------------------------------------------------------------------

#include <AnyClass.hh>

AnyClass::AnyClass() {
...
} // AnyClass

//--------------------------------------------------------------------------

AnyClass::~AnyClass() {
...
} //~AnyClass
```