

Bewertungsschema

Softwareentwicklung Praktikum

Stand: 27.02.2008

I. Einleitung

Dieses Dokument ist an die StudienassistentInnen der Lehrveranstaltung Softwareentwicklung Praktikum adressiert und enthält Richtlinien, die bei der Punktevergabe herangezogen werden. Um ein für alle TeilnehmerInnen transparentes Bewertungsschema zu haben, wird dieses Dokument auch allen Studierenden zugänglich gemacht.

II. Bewertung der Gruppenübungsabgabe

| Zu erbringende Leistungen | | Mögliche Teilabzüge wegen | |
|---|------|---|--------------------|
| Angemessenes Designdokument | 1 | Dünne Ausführung | -1 |
| | | Widerspruch zu Implementierung | -1 |
| Dokumentation des Quellcodes | 1 | Dokumentation nicht oder mager vorhanden | -1 |
| | | Dokumentation passt nicht zu Code | -1 |
| Makefile (alle Targets funktionieren) | 1 | Zufällige funktionierende Targets | -1 |
| | | Eigene Abhängigkeiten nicht erfüllt | -1 |
| Codingstandard | 2 | 1-3 unterschiedliche Verstöße | -1 |
| | | 4-6 unterschiedliche Verstöße | -2 |
| | | Schlechte Variablennamen (i, j, ...) | -1 |
| OOP Konzept umgesetzt | 2 | Member Variablen als Parameter bei zur selben Klasse gehörenden Methoden | -1 |
| | | Member die logisch nicht zur Klasse gehören (z.B. Member statt Parameter) | -1 |
| | | Verwendung von Rückgabewerten anstatt Zuweisung auf Member Variable. | -1 |
| | | Unnötiges Scoping auf die eigene Klasse | -1 |
| | | Unnötige Friend Deklaration | -1 |
| Übersichtlicher Kontrollfluss | 1 | Unübersichtlich | -1 |
| | | Mehrere Klassendefinitionen in einer Datei (außer bei Exceptions) | -1 |
| | | Globale Variablen | -1 |
| Konstanten | 1 | Unnötige Hardcoded Values | -1 |
| | | #define verwendet | -1 |
| Exceptions richtig verwendet | 1 | Kontrollfluss durch Exceptions | -1 |
| | | Silent Catch | -1 |
| | | Exceptions überhaupt nicht verstanden | -1 |
| CopyConstructor und Assignoperator entweder überschrieben oder <i>private</i> | 1 | eines davon fehlt | -1 |
| Richtiges Speicherhandling | 3 | (Drohender) Segmentation Fault | bis -2 |
| | | Wachsender Speicherverbrauch | bis -2 |
| Bestandene Testfälle | 6-12 | Je nicht bestandenem Testfall | -0.5 oder -1 |

Anmerkung 1: Die linke Spalte stellt die erreichbaren Punkte dar, die rechte Spalte Beispiele für Gründe warum (und zu welchem Ausmaß) diese nicht erreicht werden können. Es gibt keine negativen Punkte für die einzelnen

Leistungen. Für den Tutor bedeutet dies bei jeder Leistung zu entscheiden („Ja, sie wurde erreicht“ oder „Nein, sie wurde nicht erreicht, **weil** ...“). Es kann auch andere Gründe geben eine Leistung als (teilweise) nicht erreicht zu bewerten, die Begründung muss aber in jedem Fall angegeben werden.

Anmerkung 2: Es werden je nach Beispiel eine unterschiedliche Anzahl an Tests durchgeführt. Die Anzahl der erreichbaren Punkte kann dem jeweiligen URD entnommen werden.

III. Allgemeines

Nach Addition aller Punkte und Abzüge können noch prozentuelle Abzüge von den insgesamt erreichten Punkten bei einem Beispiel abgezogen werden. Führen diese Abzüge zu Kommazahlen wird kaufmännisch gerundet.

- I. Verspätete Abgabe: **-50 %**¹
- II. Warnings beim Kompilieren mit -Wall: **-5% pro Warning**
- III. Spezifikation: Verstöße werden durch hohe Punkteabzüge geahndet. Um erste, unnötige Unachtsamkeiten nicht gleich durch einen rigorosen Abzug zu bestrafen, gibt es folgende Regelung.
 - i. Erster Verstoß: **-25 %**
 - ii. Zweiter Verstoß: **-50 %**
 - iii. Dritter Verstoß: **-100 %** und der Übungsteil der LV ist beendet.Diese Verstöße werden **über alle Beispiele hinweg** gezählt (d.h. hat man einen Verstoß bei Beispiel 1 so ist der nächste, egal bei welchem Beispiel, der zweite Verstoß (-50%).
- IV. Konsequentes Ignorieren des Codingstandards (> 6 unterschiedliche Verstöße): **-100 %**
- V. Mitschwimmer: Je nachdem, wie wenig diese sich einbringen werden sie mit **bis zu -100 % der Punkte**² bestraft.
- VI. Falsche Programmiersprache: **-100 %**
- VII. Code kompiliert nicht: **-50 %**

Spezifikationsfehler

Im Anschluss eine Auflistung an Spezifikationsfehlern. Jede dieser Kategorien zählt als separater Fehler³.

- I. Makefile ohne -Wall
- II. Falsche Dateinamen oder Archivstruktur
- III. Ausgabe auf falsche Pipe (cerr bzw. cout)
- IV. Zusätzliche Ausgaben (Debug Output)
- V. Geforderte Outputs des URDs falsch (erkennbar an der Schriftart `Courier` und Anführungszeichen)⁴
- VI. Vorgaben aus dem URD ignoriert

¹ Bei einer Abgabe innerhalb von 48 Stunden nach Ende der Abgabefrist.

² Diese Entscheidung obliegt dem Tutor. Bei wiederholtem Verdachtsfall werden immer 100% abgezogen.

³ Das heißt, wird bei einer Abgabe ein falscher Dateiname verwendet und zusätzlich Debug Output ausgegeben sind dies bereits zwei Spezifikationsfehler.

⁴ Dies betrifft **keine** einfachen Tippfehler (diese werden durch fehlende Punkte bei den Testfällen abgedeckt), sondern willkürliche Änderungen

IV. Negative Beurteilung

Ohne weitere Beurteilung führt folgendes Fehlverhalten zu einer negativen Note auf die Lehrveranstaltung:

- I. Plagiat: Kopieren von Code von einer anderen Gruppe oder aus einer externen Quelle.
- II. Fehlende Abgabe: Alle Aufgaben (Hausübungen und Gruppenübungen) müssen abgegeben werden.

V. Plagiate

In Bezug auf Abschreiben vertreten wir eine Null-Toleranz-Politik. Die Arbeit, die in dieser Lehrveranstaltung bewertet wird, soll Ihre eigene sein. Die Bestrafung durch Lehrveranstaltungsausschluss betrifft im Zweifelsfall alle involvierten Gruppen. Nur wenn eindeutig eine Quelle und eine Senke identifiziert werden, wird die Quelle mit **0 Punkten auf das aktuelle Beispiel** bestraft.

Es ist natürlich erlaubt, zu einem gewissen Teil auf Algorithmen oder Code-Fragmente zurückzugreifen, die man bei Recherchen in Büchern, im Internet oder durch Erklärungen von Kommilitonen erhält. In jedem Fall müssen hierbei folgende Voraussetzungen erfüllt sein:

- ✓ Die Länge des verwendeten Codes / Algorithmus übersteigt nicht einige Zeilen.
- ✓ Die Quelle wird genannt und der übernommene Code / Algorithmus ist eindeutig gekennzeichnet.
- ✓ Der verwendete Code / Algorithmus muß verstanden worden sein und kann beim Abgabegespräch detailliert erklärt werden.

Beispiele für Kennzeichnung:

```
// nach K. Schmaranz, Softwareentwicklung in C++, ISBN 3-540-41958-6
// aus: http://www.cplusplus.com/reference/cstdlib/rand.html
// erarbeitet mit Gruppe XYZ Tutor A.B.
```

Und danach

```
// begin
...
// end
```

Auch unter Einhaltung dieser Kriterien darf die Länge **aller** aus externen Quellen stammenden Teile Ihres Programms 1/3 der Gesamtlänge **nicht** übersteigen.

VI. Bonuspunkte

Bonuspunkte werden nur dann zum Ergebnis hinzugezählt, wenn bereits eine positive Note erreicht wurde.