Methodische Grundlagen des Software Engineering - Übung 10

10 Testen

Abgabe der Hausaufgaben am Anfang der jeweiligen Präsenzübung am 21.06.2011 bzw. 22.06.2011.

10.1 Grundlagen Testen

10.1.1 Was ist der Unterschied zwischen Testen und Debugging?

10.1.2 Erläutere den Unterschied zwischen statischem und dynamischem Test.

10.1.3 In welchem Zusammenhang stehen Reviews und die statische Analyse?

10.2 Grundlagen Äquivalenzklassen und Grenzwert

Gegeben ist die Methode containsCharacters mit folgender Signatur:

```
1 public class StringAnalyzer {
2         public static String containsCharacters(String input){
3                 // Der Code der Methode ist unbekannt.
4         }
5 }
```

Die Methode untersucht, welche ASCII-Buchstaben aus [m-sM-S] in dem String input vorkommen, und gibt diese in einem alphabethisch sortierten String zurück.
Sollte der gleiche Buchstabe mehrmals vorkommen, so wird nur ein Repräsentant im Rückgabestring aufgeführt.
Großbuchstaben kommen in der Sortierung jeweils vor dem entsprechenden Kleinbuchstaben. Alle anderen Eingabezeichen werden ignoriert.

10.2.1 Welche Äquivalenzklassen von Eingabewerten sind sinnvollerweise zu testen?

10.2.2 Geben Sie für jede Äquivalenzklasse wenigstens einen Eingabestring und das Soll-Ergebnis in einer Tabelle an. Die entstehende Tabelle soll aus folgenden Spalten bestehen: Testfallnummer, Eingabe, Äquivalenzklasse, Ausgabe(Soll) und Bemerkung.

10.2.3 Welche Sonder- und Randfälle sind sinnvollerweise ebenfalls zu testen? Geben Sie für die Sonder- und Randfälle Testfälle in Form der obigen Tabelle an.

10.2.4 Schreiben Sie JUnit-Tests, die Ihre Testfälle abprüfen.

10.2.5 Gegeben ist eine Funktion zur Bestimmung der Anzahl der Tage eines Monats mit den Übergaben Monat und Jahr.
ZahlTageMonat(int Monat, int Jahr)
Wie sehen die Äquivalenzklassen dazu aus?

10.3 Überdeckung

Gegeben folgendes Programm:

```
1  void abc(int a, int b, int c){
2          while((a + b + c) < 100) {
3                  if((a-b) < c) {
4                          b++;
5                  }
6                  if((b+c) == a) {
7                          c += 2;
8                  }
9                  else {
10                         if(a == b) {
11                                 a++;
12                         }
13                 }
14                 a++;
15         }
16 }
```

10.3.1 *Zeichnen Sie den Kontrollflussgraphen.*

10.3.2 *Finden Sie eine minimale Menge von Testfällen (Eingabe-Arrays mit Belegungen der Variablen a, b und c), die bei der Ausführung alle Anweisungen überdecken (C0 - Test). Notieren Sie die Reihenfolge der dabei ausgeführten Codezeilen.*

10.3.3 *Finden Sie eine minimale Menge von Testfällen, die bei Ausführung alle Zweige überdecken (C1 - Test). Notieren Sie die Reihenfolge der dabei ausgeführten Codezeilen.*

10.3.4 *Wie viele Fälle (Eingabe-Arrays) sind zur Pfadabdeckung (C2 - Test) notwendig? Bitte begründen Sie Ihre Antwort.*

10.3.5 *Wie sieht ein minimales Programm aus, für das mindestens zwei Eingabewerte notwendig sind, um eine Anweisungsüberdeckung zu erreichen.*

Hausaufgabe

10.4 Äquivalenzklassen und Grenzwerte

10.4.1 *Ein Programm zur Berechnung von Preisen soll getestet werden. Das Programm erfasst die Artikelnummer, die Stückzahl und einen Rabatt. Die Artikelnummer ist eine 5-stellige Zahl. Die Stückzahl ist auf 10.000 begrenzt und muss mindestens gleich 1 sein. Der Rabatt liegt zwischen 0 und 100%. Stellen Sie alle Äquivalenzklassen auf. Bilden Sie Testfälle für die Grenzwertanalyse.*

2 P

10.5 Überdeckung

Gegeben ist das folgende Programm. Es wandelt eine Binärzahl in eine Dezimalzahl. Die Stellen der Binärzahl werden invers eingelesen, d. h. die letzte Ziffer zuerst, danach die vorletzte usw.

```
1  #define INT_MAX 200
2  int wandleDezimalZahl(){
3          int zahl = 0;
4          int potenzwert = 0;
5          char zeichen;
6          scanf(%c,zeichen);
7          while (((zeichen == "0") || (zeichen == "1")) && (zahl < INT_MAX)){
8                  if (zeichen == "1") {
9                          zahl = zahl + pow(2,potenzwert);
10                 }
11                 potenzwert++;
12                 scanf(%c,zeichen);
13         }
14         return zahl;
15 }
```

10.5.1 *Wandeln Sie das Programm in einen Kontrollflussgraphen um.*

1 P

10.5.2 Erstellen Sie Testfälle für einen vollständigen Anweisungsüberdeckungstest (C0)

1 P

10.5.3 Erstellen Sie Testfälle für einen vollständigen Zweigüberdeckungstest (C1)

1 P