Softwaretechnik
Wintersemester 2024/25, Übungsblatt 8

Prof. Dr.-Ing. Norbert Siegmund
Stefan Jahns, M. Sc.

Diskussion am: 08.01.2025

Aufgabe 1: Grundlagen

a) Welche Fehlerklassen gibt es, geben Sie für drei der Fehlerklassen Beispiele.

b) Was verbirgt sich hinter dem Begriff Regression Testing?

c) Nennen Sie die entscheidenden Vor- und Nachteile von
   I. Testing
   II. Model Checking.

Aufgabe 2: Kontrollflussgraph & Testabdeckung

a) Zeichnen Sie den Kontrollflussgraphen für folgende Java-Methode.

```
1  public static double compute(double a, double b) {
2          double c = a + b;
3          if (c < 0) {
4              c = c + a;
5          } else if (c > 0) {
6              c = 2 * c;
7          } else {
8              java.lang.System.out.print("c is 0!");
9          }
10         if (c == 0.0 && a < -2.0) {
11             c = 10.0;
12             if (b > 10.0) {
13                 c = 20.0;
14             }
15         }
16         return c;
17     }
```

b) Wie viele Tests werden für einen C_0-Test benötigt? Geben Sie ein minimales Test-Set an.

c) Wie viele Tests werden für einen C_1-Test benötigt? Geben Sie ein minimales Test-Set an.

Aufgabe 4: Black-Box- und White-Box- Unit-Testing

Gegeben sei ein Softwareprojekt, welches Zeitberechnungen ermöglicht. Folgende Methoden sollen getestet werden:

```
public class Date {
        /**
          * @param year a valid year
          * @param month a month position from 1 to 12
          * @param day a valid day position within the month
          * @throws InvalidDateException if date is invalid
          */
        public Date(int year, int month, int day) throws InvalidDateException        { ... }


        /**
          * A method to modify dates without crossing month boundaries.
          *
          * @param numDays the number of days to add to the current date - should        work with
positive and negative values, probably maybe.
          * @return a new Date object representing the result
          * @throws InvalidDateException if resulting date is not in the same month
          */
        public Date modifyDaysWithinMonth(int numDays) throws InvalidDateException
        { ... }

        /**
          * @return true if the current year is a leap year
          */
        public boolean isLeapYear()
        { ... }
}
```

Es ist davon auszugehen, dass der Quelltext nicht fehlerfrei ist. Gegebenenfalls enthaltene Fehler sollen nicht behoben werden. Um Ihre Tests im Rahmen dieser Aufgabe ausführen zu können, stellen wir den Quelltext im Moodle-Kurs (Uebung-Quelltext-Ordner) bereit.

Für diese Aufgabe ist die Sichtung des Quelltextes im Moodle-Kurs notwendig.

a) Entwerfen Sie einen Testplan mit Black-Box-Tests und notieren Sie dabei für jeden Test die Eingabewerte und das erwartete Ergebnis. Bedenken Sie dabei, dass Sie bei Black-Box Testing lediglich die Signatur und Dokumentation einer Methode, aber nicht deren Quelltext verwenden können. Greifen Sie hierbei auf Verfahren der Vorlesung zurück:

- Äquivalenzklassen finden — Gültige und ungültige Bereiche von Parametern bestimmen
- Grenzwerte testen — also jeweils Werte innerhalb und außerhalb von Grenzen
- Erfahrung und Heuristik — eine Routenplanungsmethode sollte bspw. für die Strecke Leipzig – New York keinen Weg zu Fuß empfehlen

b) Implementieren Sie Ihre Black-Box-Tests mit JUnit. Zu JUnit 5 steht ein kurzes Beispiel im Moodle-Kurs zur Verfügung (JUnit-Beispiel-Ordner).

c) Geben Sie aus dem Quelltext des gegebenen Softwareprojekts zwei Code-Bereiche an, die mit reinem Black-Box-Unit-Testing mit hoher Wahrscheinlichkeit nicht abgedeckt werden.

d) Bestimmen Sie die Anweisungsüberdeckung Ihrer bisherigen Black-Box-Tests.
Nutzen Sie hierfür die Funktionalität Ihrer IDE. Für Eclipse bietet sich das Plugin EclEmma an. IntelliJ bietet hierfür einen Button neben dem Debug-Symbol.

e) Schreiben Sie White-Box-JUnit-Tests um eine möglichst vollständige Anweisungsüberdeckung zu erzielen.

f) Bestimmen Sie anschließend Ihre erzielte Anweisungsüberdeckung.