Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

**Objektorientierte Analyse und Design**

Wintersemester 2024
**1b. Aufgabenblatt**

**Die Lösungen zu diesem Blatt müssen im Praktikum am 15./16./10.10(!) vorliegen.**

**Aufgabe 4 (3 Punkte)**

In der Veranstaltung wird davon ausgegangen, dass Sie bereits grundlegende Kenntnisse über das automatische Testen haben. Sie haben Tests im Studium bereits kennengelernt und sich selbst damit intensiver in Ihren individuellen Projekten beschäftigt. Diese Aufgabe bietet eine ergänzende Fundierung.

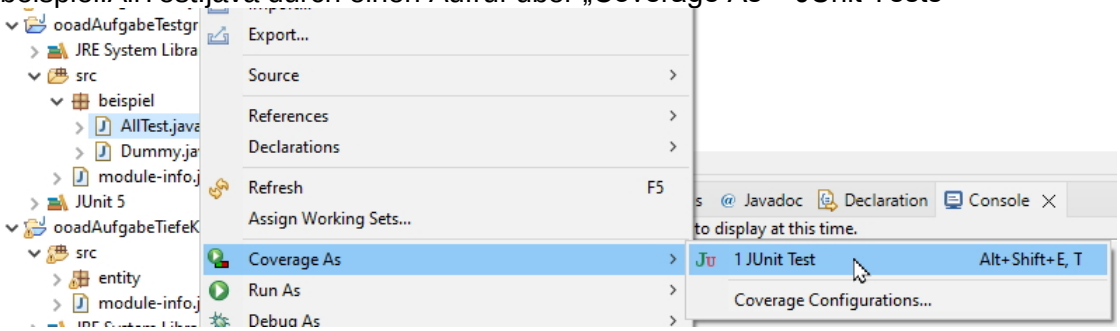
Gegeben sei das Projekt ooadAufgabeTestgrundlagen von der Veranstaltungsseite.

a) Erklären Sie schriftlich was der Unterschied in Java zwischen den Zeilen

```
boolean erg = a && b;
boolean erg = a & b;
```

Geben Sie weiterhin ein Beispiel an, bei dem sich ein Programm anders verhält, wenn an einer Stelle && durch & ersetzt wird.

b) Führen Sie die im Projekt gegebenen Tests mit einem Rechtsklick auf beispiel.AllTest.java durch einen Aufruf über „Coverage As > JUnit Tests“




aus. Beim Klick auf die Klasse Dummy.java Sie erhalten eine Ausgabe mit verschiedenen Farben und kleinen Rauten am Rand. Erklären Sie schriftlich, was diese Farben und für jede kleine Raute, was sie bedeuten.

c) Ergänzen Sie eine Klasse FehlendeTest.java, so dass bei der gemeinsamen Ausführung zusammen mit AllTest die Ausgabe nur grüne Zeilen anzeigt. Um die Tests zusammen auszuführen, kann z. B. ein Rechtsklick auf dem Projekt und dort wie oben „Coverage As > JUnit Tests“ ausgewählt werden.

d) Generell sind hohe Test-Überdeckungen elementar für die Qualität von Software. Allerdings bieten hohe Überdeckungen keine garantierte Korrektheit (einfachster Fall: Assertions werden vergessen oder schlecht formuliert). Gegeben sei die folgende Methode aus der Klasse ueberdeckung.KeinBlindesVertrauen. Schreiben Sie Tests mit sinnvollen Assertions, so dass der Code vollständig überdeckt wird, aber das Problem nicht gefunden wird. Ihre Tests sollen überprüfen, dass der absolute Betrag des Ergebnisses der Methode calc() immer 6 ist.

```
public static int calc(boolean a, boolean b){
  int x = 0;
  if (a) {
    x = 2;
  } else {
    x = 3;
  }
  if (b) {
    return(6 / (x - 3));
  }
  return(6 / (x - 2));
}
```

Prof. Dr. Stephan Kleuker
Hochschule Osnabrück
Fakultät Ing.-Wissenschaften und Informatik
- Software-Entwicklung -

**Objektorientierte Analyse und Design**

Wintersemester 2024
**1b. Aufgabenblatt**

**Aufgabe 5 (5 Punkte)**
Laden Sie von der Veranstaltungsseite das Projekt ooadAufgabeListe.

a) Realisieren Sie das enthaltene Interface util.MyList in der Klasse util.MyListImpl. Sie dürfen dabei keine Collections, also z. B. Arrays, Lists, Sets oder Maps, nutzen und sollen die Listenelemente einzeln verknüpfen. Entwickeln Sie testgetrieben und realisieren Sie zunächst add(), get() und size() und testen Sie dann Ihre Implementierung mit der Klasse util.MyListImplTest. Realisieren Sie dann schrittweise immer eine weitere Methode und lassen Sie dann die Tests wieder laufen.
b) Messen Sie die Überdeckung. Falls eine Ihrer Implementierungen nicht „grün“ sein sollte, schreiben Sie auf, warum das so ist.
c) Beim genaueren Blick auf die Interface-Beschreibung sollten Ihnen zwei wichtige Details auffallen, die nicht getestet werden; welche sind das?

**Aufgabe 6 (1 Punkt)**
Überlegen Sie sich ein Programm P, das die Klassen A1, A2, B1, B2 nutzt, wobei folgende Eigenschaften erfüllt sind:

- A2 erbt von A1
- B2 erbt von B1
- P läuft, wenn A1 und B1 genutzt werden
- P läuft, wenn A1 durch A2 ersetzt wird zusammen mit B1
- P läuft, wenn B1 durch B2 ersetzt wird zusammen mit A1
- P läuft nicht, wenn A1 durch A2 und B1 durch B2 ersetzt werden.

Schreiben Sie die geforderten Klassen und geben Sie zu jeder der vier zuletzt genannten Eigenschaften passende JUnit-Tests von P an.
P könnte z. B. einen Konstruktor `public P(A1 a, B1 b)` haben.