

# 11. Übungsblatt zu Software Qualität

Michel Meyer, Manuel Schwarz

18. Januar 2013

## Aufgabe 11.1 - Mutationen-Test

### (a) Mutanten

Siehe \*.java-Dateien Pow2M1 - Pow2M6.

### (b) Starker Mutationstest

Die Tabelle 1 zeigt die Ergebnisse des starken Mutationstests. Dabei steht eine 1 für einen erkannten Mutanten und eine 0 für einen nicht erkannten Mutanten. Der *Test B* ist der beste, da dieser 4/6

Tabelle 1: Ergebnisse des starken Mutationstests.

	M1	M2	M3	M4	M5	M6
Testfall A	0	1	1	1	0	0
Testfall B	0	1	1	1	0	1
Testfall C	0	1	0	0	0	0

Mutanten erkennt. *Test C* hingegen erkennt nur 1/6 Mutanten und ist damit der schlechteste.

### (c) Score

Für den Score gilt folgende Formel:

$$Score = \frac{\text{Anzahl erkannter Mutanten}}{\text{Anzahl aller Mutanten} - \text{Anzahl äquivalenter Mutanten}} \quad (1)$$

Dabei ist ein Mutant zum Original äquivalent, wenn keine Testfälle existieren, die den Mutanten als solchen identifizieren können. Ist der Score eines Testfalles 1, so nennt man diesen *adäquat*.

$$\text{Test A: } Score = \frac{3}{6 - 2} = 0.75$$

$$\text{Test B: } Score = \frac{4}{6 - 2} = 1.0$$

$$\text{Test C: } Score = \frac{1}{6 - 2} = 0.25$$

Damit ist *B* ein *adäquater* Testfall.

### (d) Beispiel

Als Beispiele dienen die Klassen Pow2.java und Pow2M1.java, da diese äquivalent sind. In Pow2M1.java steht

```
i = n
if (n < 1) then
    i = -n
end if
```

---

Dies ist äquivalent zu:

```
i = n
if (n < 0) then
    i = -n
end if
if (n == 0) then
    i = -n
end if
```

---

Und da  $-0 = 0$  sind Pow2.java und Pow2M1.java äquivalent.

Ein weiteres Beispiel könnte ein Programm mit der Zeile

```
if (a == b) then
    do stuff
end if
```

---

sein. Durch eine erste Mutation könnte dies dann so aussehen:

```
if (a != b) then
    do stuff
end if
```

---

Und eine zweite Mutation könnte das Folgende verursachen:

```
if !(a != b) then
    do stuff
end if
```

---

Damit wäre dieser Mutant äquivalent zum Original.

## Aufgabe 11.2 - Stilanalyse

### 5.1.4 End-Of-Line Comments

Diese Konvention trägt dazu bei, dass erfahrene Java Programmierer schneller erkennen können, um was für Kommentare es sich handelt. Dabei soll “//” suggerieren, dass es sich (höchstwahrscheinlich) um auskommentierten Code handelt und soll sich damit zu beispielsweise “/\* ... \*/” abgrenzen, da letzteres für tatsächliche, hilfreiche Kommentare genutzt wird. Ein Programmierer weiß dadurch sofort genau, wo er zu gucken hat, wenn er beispielsweise nach auskommentierten Code sucht.

### 7.3 return Statements

A return statement with a value should not use parentheses unless they make the return value more obvious in some way. Example:

```
return;  
  
return myDisk.size();  
  
return (size? size : defaultSize);
```

Die Klammern wegfällen zu lassen ist aus zwei Gründen sinnvoll:

- Das statement ist als Satz lesbar: “`return myDisk.size()`” => “gib `myDisk.size()` zurück”.
- Klammern (als Alternative) würden `return` als eine Funktion suggerieren, bei dem der Rückgabewert als Parameter angegeben wird, was einer imperativen Sprache entgegen spräche.

Daher gibt es auch Ausnahmefälle wie oben, denn “`return size? size : defaultSize;`” würde sich vielmehr als “`(return size)? size : defaultSize;`” lesen, was nicht der Semantik (und Syntax) entspricht.