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

Software-Qualität

Sommersemester 2024
5. Aufgabenblatt

Aufgabe 0.5 (0 Punkte)
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Hinweis: Etwas anders als in den Folien zeigt EclEmma (statt JaCoCo) nicht direkt die Anzahl der „Covered Branches“ an, die für C1 relevant ist. Da es aber die Diamanten am Rand von Booleschen Bedingungen gibt und die Bedingungen eingefärbt werden, ist das kein echtes Problem. Für eine 100% C1-Überdeckung muss die Bedingung eingefärbt sein. Für eine 100% C0-Überdeckung gilt weiterhin, dass es keine rote Zeile geben darf.

Aufgabe 9 (2 Punkte)

Die Angabe „Missed Branches“ von der Eclipse-Überdeckung (EclEmma) entspricht nicht genau der vorgestellten C1-Überdeckung, wenn zusammengesetzte Boolesche Ausdrücke genutzt werden.

a) Überlegen Sie sich für die Klasse auf der rechten Seite aus dem Projekt qsAufgabeEclEmmaAnalyse von der Veranstaltungsseite Tests, die zu einer C0- aber nicht zu einer C1-Überdeckung führen. Welches Ergebnis liefert Eclipse? (Machen Sie einen Screenshot)
b) Überlegen Sie sich, woher die Ergebnisse von Eclipse kommen könnten und schreiben Sie Tests, die zu 100% Coverage in Eclipse führen.

Anmerkung: Es könnte sinnvoll sein, das Projekt für die zweite Teilaufgabe zu kopieren und die Ergebnisse zu dokumentieren.

```
package tmp;

public class Mini {

  public int mach(int x, int y){
    int erg = 40;
    if(x > 0 | y > 0){
      erg++;
    }
    erg++;
    return erg;
  }
}
```

Aufgabe 10 (4 Punkte)

```
public enum Waffe {
  Uzzi, Volksmusik, Wumme, Mine, Handgranate, Gebabbel
}

public class Schadensevaluator {
	public int abzug(Waffe links, Waffe rechts){
		int ergebnis = 0;
		if(links == Waffe.Gebabbel & rechts == Waffe.Wumme)
			ergebnis += 40;
		else
			ergebnis += 10;
		if(links == Waffe.Gebabbel | rechts == Waffe.Volksmusik)
			ergebnis += 100;
		if(links == rechts)
			ergebnis *= 2;
		else
			ergebnis += 10;
		return ergebnis;
	}
}
```

Gegeben sei die Klasse Schadensevaluator aus dem Projekt qsAufgabeUeberdeckungAbzug, deren Methode **abzug** getestet werden soll. Schreiben Sie Ihre Überlegungen und Tests zunächst auf. Überprüfen Sie danach Ihre Ergebnisse mit der Eclipse-Coverage, beachten Sie, dass die C1/C2-Betrachtungen in den Eclipse-eigenen Überdeckungsbetrachtungen

zusammengefasst sind. (Methodenergebnisse dürfen vereinfachend ignoriert werden, wobei Sie in der Realität natürlich elementar wichtig sind).
Statt das Projekt mehrfach zu kopieren, können Sie auch Bildschirmfotos der erreichten Überdeckungen machen und dazu die genutzten Testfälle notieren, die Sie mit Hilfe von JUnit-Tags (@Tag, @Tags, auch @SelectPackages nutzen und bedenken dass dazu alle Testklassen auf „Test“ enden müssen) gruppieren sollen.

a) Geben Sie eine Menge von Testfällen zur C0-Überdeckung von **abzug** an, die keine C1-Überdeckung ist.
b) Geben Sie eine Menge von Testfällen zur C1-Überdeckung von **abzug** an, die keine C2-Überdeckung ist. Begründen Sie, warum keine C2-Überdeckung vorliegt.
c) Geben Sie eine Menge von Testfällen zur C2-Überdeckung von **abzug** an, die keine C1-Überdeckung ist. Begründen Sie, warum keine C1-Überdeckung vorliegt.
d) Geben Sie eine Menge von Testfällen zur Überdeckung von **abzug** an, so dass EclEmma alles grün anzeigt.

Aufgabe 11 (2 Punkte)

Anmerkung: Bei a) – c) kann der Code einfach in ein Textdokument kopiert werden.

a) Schreiben Sie ein kurzes lauffähiges Java-Programm, von dem es keine C0-Überdeckung geben kann.
b) Schreiben Sie ein kurzes lauffähiges Java-Programm, von dem es keine C1-Überdeckung geben kann, für das aber eine vollständige C0-Überdeckung existiert.
c) Schreiben Sie ein kurzes lauffähiges Java-Programm, von dem es keine C2-Überdeckung geben kann, für das aber eine vollständige C1-Überdeckung existiert.
d) Gegeben sei die folgende Methode. Informieren Sie sich über die Überdeckungsvariante MC/DC (Modified Condition/Decision Coverage) und erklären Sie sie anhand von Überdeckungen der Methode.

```
public boolean und(boolean a, boolean b, boolean c){
  if (a & (b | c)) /* keine Kurzschlussauswertung ! */
    return true;
  return false;
}
```

e) Messen Sie die Überdeckung zu Ihrer Lösung der Intervall-Aufgabe qsAufgabeIntervall vom dritten Aufgabenblatt. Falls nicht alle geforderten Methoden *des Interfaces* zu 100% überdeckt sind, prüfen Sie und begründen Sie, warum dies der Fall ist.

Anmerkung: Sie sollten bei den Lösungen zu allen Aufgaben dieses Blattes festgestellt haben, dass Sie keine Zusicherungen (Assertions) in den Code schreiben mussten. Dies soll Ihnen verdeutlichen, dass Überdeckungen sinnvoll und notwendig sind, aber noch nichts über die Qualität der Tests aussagen.