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

Software-Qualität

Sommersemester 2024
8. Aufgabenblatt

Aufgabe 0.8 (0 Punkte)
Geben Sie das Lösungswort des Quiz aus der Lernnotiz an.

Aufgabe 17 (4 Punkte)
Gegeben sei folgende Klasse `Gegner`, deren Methode `knoten` getestet werden soll. Es liegen keine weiteren Implementierungen anderer Klassen vor, so dass Sie gezwungen sind, Mocks zu schreiben. Da dies bei Enumerations nicht möglich ist, muss hier eine konkrete Implementierung angegeben werden. Da leider keine Spezifikation der erwarteten Ergebnisse vorliegt, müssen Sie sich auf die Überdeckung konzentrieren.

```
public class Gegner {

  @SuppressWarnings("unused")
  private String typ;
  private boolean hatFK;
  private int feuerweite;

  public Gegner(String typ, boolean hatFK, int feuerweite) {
    this.typ = typ;
    this.hatFK = hatFK;
    this.feuerweite = feuerweite;
  }

  public int knoten(Alarm alert, Eigenschiff e, int abstand) {
    int ergebnis = 42;
    if (alert == Alarm.ROT & abstand < this.feuerweite) {
      ergebnis = e.getMaxKnoten();
    }
    if (alert == Alarm.ORANGE & e.wichtigerAuftrag()) {
      ergebnis = e.sicherheit(e.hatAbwehr(this.hatFK), abstand);
    }
    return ergebnis;
  }
}
```

a) Geben Sie möglichst einfache, aber dennoch sinnvoll nutzbare Mock-Klassen an (ohne Nutzung eines Mock-Frameworks), so dass Sie eine 100%-Branch-Überdeckung für Ihre zu erstellenden Tests erhalten. Nutzen Sie das Projekt qsAufgabeGegnerMock.
b) Schreiben Sie Ihre Lösung aus a) um (oder neu), so dass die Mock-Objekte durch Mockito erstellt werden. Nutzen Sie das Projekt qsAufgabeGegnerMockito.

Aufgabe 18 (4 Punkte)
Ein Überwachungsprogramm für ältere Maschinen fragt zyklisch zwei Werte von nachträglich installierten Sensoren ab. Die Maschinen sind über folgendes Interface angeschlossen.

```
public interface MInterface {
    Integer getWert1() throws NetworkException, HardwareException;
    String getZustand1() throws NetworkException;
}
```

Das Überwachungsprogramm endet, wenn ein schwerer Fehler festgestellt wird, was u. a. an der Variable `alarm` zu erkennen ist. Ein schwerer Fehler tritt auf, wenn in drei aufeinanderfolgenden Zyklen jeweils mindestens einer der abgefragten Werte als Antwort eine Exception oder aufeinander folgende Messwerte mit einem Abstand größer 5 liefert. Ein

schwerer Fehler tritt weiterhin auf, wenn die Frage nach dem Status1 einmal mit „defekt“ beantwortet wird.

<table>
<tr><td rowspan="3">schwerer Fehler</td><td rowspan="2">drei beliebige einfache Fehler direkt hintereinander</td><td>einfacher Fehler: im Zyklus treten eine oder mehrere Exceptions auf</td></tr>
<tr><td>einfacher Fehler: aufeinander folgende Messwerte haben Abstand größer 5 (bei Exceptions wird dabei immer der letzte Messwert vor der Exception betrachtet)</td></tr>
<tr><td colspan="2">oder Status1 ist „defekt“</td></tr>
</table>

Zum nachfolgenden zu testenden Überwachungsprogramm `Ueberwachung` aus dem von der Veranstaltungsseite herunterladbaren Projekt qsAufgabeMockitoMaschinenueberwachung fehlen leider Maschinen, die das Interface MInterface realisieren.

a) Prüfen Sie die Funktionalität der Klasse Überwachung, genauer der Methode `sensorenLesen()`, mit Tests, nutzen Sie dazu passende, mit Mockito zu erstellende, Mocks für Maschinen. Sie sollten zumindest eine 100%-Branch-Überdeckung dieser Methode mit erreichen.
b) In a) sollten Sie mindestens zwei (miteinander „verwandte“) Fehler finden, die zu einem zu frühen und zu einem fehlenden Alarm führen. Korrigieren Sie das Programm in einer neuen Klasse und nutzen Sie Ihre Tests aus a) zur Überprüfung.