**Software Engineering - Praktikum 04 - Unit-Tests**

**Vorbereitung**

In der Vorlesung wurde Ihnen die Wichtigkeit der Qualitätssicherung erläutert. Sie haben u.a. erfahren, wie Unit-Tests in Java umgesetzt werden können. In diesem Praktikum bauen Sie Ihre Unit-Testing-Kompetenz aus. Insbesondere sollen Sie lernen, wie sie Tests an der Konsole ausführen können. Dies ist wichtiger Teil der Verständnisbildung für spätere Praktika und Lehrinhalte.

Darüber hinaus haben Sie auch eine Methode kennengelernt um die Entwicklung von Test-Code und funktionalem Code zu verzahnen: Test-Driven-Development. Sie wenden diese Methode in diesem Praktikum an. In diesem Praktikum arbeiten Sie folglich nicht mit fertig implementierten Java-Klassen, im Gegenteil, es ist Teil dieses Praktikums, dass Sie Test-Code - und nur soweit erforderlich - funktionalen Code in einem Test-Driven-Design Ansatz entwickeln.

**Vorbereitung**

Sie arbeiten weiter an dem Projekt, welches Sie im vorangegangenen Praktikum begonnen haben.

**Durchführung**

Im Folgenden werden die Arbeitsschritte beschrieben, die Sie durchführen müssen, um das Praktikum erfolgreich zu absolvieren.

**Schritt 0: JUnit Setup**

Damit Sie Unit-Tests entwickeln und an der Konsole ausführen können, müssen Sie wenige einfache Vorbereitungen treffen:

- Für die Ausführung von Unit-Tests mit JUnit 5 an der Konsole verwenden Sie das Java Module *JUnit Platform Console Standalone* in der Version 1.10.0. Sie können dieses unter folgendem Link herunterladen: https://repo1.maven.org/maven2/org/junit/platform/junit-platform-console-standalone/1.10.0/junit-platform-console-standalone-1.10.0.jar
- Das Module muss bei der Kompilierung der Testklassen und für die Ausführung der Tests im *Java classpath* auffindbar sein
- Machen Sie sich im Internet, z.B. unter https://junit.org/junit5/docs/current/user-guide/#running-tests-console-launcher mit der Nutzung des Console-Launchers vertraut.

Arbeitsschritte:

1. Schreiben Sie eine TestHelloWorld.java Testklasse mit der Sie prüfen, dass Sie Tests kompilieren und ausführen können. Die Klasse muss keine anderen Klassen

verwenden, Sie können sich einfache Tests ausdenken und in dieser Klasse einige Assertions (mind. eine !) Ihrer Wahl verwenden.

2. Dokumentieren Sie in der README.md wo im Projekt das Module *junit-platform-console-standalone-1.10.0.jar* im Projektverzeichnis abgelegt werden soll.
3. Dokumentieren Sie in der README.md wie eine *TestHelloWorld.java* an der Konsole mit dem Befehl javac kompiliert und mit dem Befehl java mittels des Console-Launchers ausgeführt wird. Alternativ können Sie auch ein bash-Skript mit dem Namen compileAndRunTests.sh erstellen, welches Ihre Testklasse kompiliert und die Tests ausführt.
4. Stellen Sie sicher, dass *junit-platform-console-standalone-1.10.0.jar nicht* in das git-Repository eingefügt wird, sondern von git ignoriert wird. Begründen Sie, warum *junit-platform-console-standalone-1.10.0.jar* nicht in das git-Repository aufgenommen werden soll.

**Schritt 1: JUnit Warm-Up und Java Packages**

Erweitern Sie TestHelloWorld.java wie folgt:

1. Ordnen Sie die Klasse dem Java-Paket de.hsd.medien.se.hsdchess.domain_` zu und seihen Sie bereit zu erläutern, was dafür zu tun ist.
2. Ergänzen Sie die Klasse um mehrere Beispiel-Tests:
    1. Ein Test, der überprüft ob eine Variable den Wahrheitswert *true* hat,
    2. einen Test, der prüft, ob durch einen Aufruf ein Fehler geworfen wird,
    3. einen Test, der prüft, ob ein Ergebnis gleich dem Wert *42* ist.
3. Experimentieren Sie mit den Lifecycle-Annotationen und setzen Sie folgende Anforderungen um:
    1. Die Klasse soll automatisch vor der Ausführung aller Tests eine Klassenvariable mit dem Namen testZaehler vom Typ int mit dem Wert 0 initialisieren und den String "TESTVORBEREITUNG" über System.out ausgeben.
    2. Vor jedem Test soll automatisch der Wert von testZaehler um 1 erhöht werden.
    3. Nach jedem Test soll über System.out der aktuelle Wert von testZaehler an System.out in der Form *n Tests wurden ausgeführt* ausgegeben werden.
    4. Nachdem alle Tests ausgeführt wurden, soll automatisch über System.out die Ausgabe der Anzahl der erfolgten Tests erfolgen und eine Ausgabe erfolgen, die kennzeichnet, dass die Ausführung der Tests in TestDemo beendet ist.

Schritt 1 ist erfolgreich erledigt, wenn

1. Die Testklasse an der richtigen Stelle im Projekt vorliegt und
2. die Tests der Testklasse entsprechend Ihrer Dokumentation ausgeführt werden können
3. die Lifecycle-Funktionen und Ausgaben zum testZaehler korrekt umgesetzt sind und funktionieren.

**Schritt 2: Testfälle für die Figur beschreiben**

In diesem Schritt sollen Sie gute Testfälle für eine Schachfigurimplementierung definieren. Sie können eine beliebige Figur - außer der im vorangegangenen Praktikum gewählten Figur - wählen.

1. Machen Sie sich mit den Spielregeln für die gewählte Spielfigur im Schach vertraut.
2. Analysieren Sie den folgenden Code-Rumpf, der die Eigenschaften und Funktionen der Figur beschreibt.

```
package de.hsd.medien.se.hsdchess.domain;


/**
 * Diese Klasse implementiert die Spielfigur X im Schach.
 * Die Spielfigur hat eine Farbe (weiß oder schwarz) und einen Status,
 * der festhält, ob sie noch auf dem Spielbrett steht oder ob sie geschlagen wurde,
 * an welcher Position auf dem Spielbrett sie steht (A1 ... H8) und
 * wie oft sie im Spielverlauf bereits bewegt wurde.
 * Die Klasse implementiert die Methode *zieheNach* für die regelkonforme Bewegung der Figur und bietet einen Konstruktor für die Erstellung eines Figurobjekts.
 */
public class FigurX {

}
```

1. Überlegen Sie gut, welche Tests (im folgenden auch Testfälle genannt) Sie zunächst für eine solche Klasse formulieren wollen. Beschreiben Sie mindestens 5 relevante Testfälle für diese Klasse in der folgenden Tabelle.
2. Die Tabelle enthält je Testfall die folgenden Information:

1. Der Name des Testfalls, exakt in der Schreibweise, die sie auch für den Namen der Testmethode in der Testfallimplementierung verwenden werden, z.B. *PferdSollteSichSoUndSoVerhalten*.
2. Der Name der Methode (kann auch der Konstruktor sein), die durch den Test ausgeführt und überprüft wird.
3. Die verwendeten Parameterwerte, die Sie der Methode übergeben.
4. Welches Verhalten Sie mit dem Testfall prüfen, z.B. *Verhalten bei regelkonformem Zug*
5. Was das erwartete Resultat (bei richtiger Implementierung) ist, z.B. *wenn Methode so und so aufgerufen wird, dann sollte das Resultat ein Fehler / die Figur an neuer Position / … sein*
6. Warum Sie genau diesen Testfall vorsehen, z.B. *Trivialtest*, *Prüfung von Randbereich*, *Funktionstest* …

| **Testfallname** | **Geprüfte(r) Methode/Konstruktor** | **Parameterwerte** | **Beschreibung des zu prüfenden Verhaltens** | **Erwartetes Verhalten oder erwarteter Rückgabewert oder erwartetes Ereignis** | **Begründung für den konkreten Testfall** |
|---|---|---|---|---|---|
| … | … | … | … | … | … |

**Schritt 3: TDD der Figur**

Implementieren Sie die Figur in einem TDD-Ansatz. Halten Sie sich an die in der Vorlesung vorgestellten Regeln des TDD.

1. Erstellen Sie für die zu testende Klasse den Code-Rumpf in Ihrem Projekt (an der richtigen Stelle) in dem Paket de.hsd.medien.se.hsdchess.domain. Implementieren Sie zunächst höchstens leere Methoden und uninitialisierte Member-Variablen für die Klasse!
2. Erstellen Sie eine Testklasse, z.B. FigurXTest in dem Paket de.hsd.medien.se.hsdchess.domain für die Überprüfung des korrekten Verhaltens der Figur-Implementierung.
3. Implementieren Sie in der Testklasse für jeden Ihrer oben erarbeiteten 5 Testfälle entsprechende Testmethoden mit JUnit wie folgt:

1. Erstellen Sie die Testmethode in der Testklasse und flechten Sie den Testfallnamen im Methodennamen ein
2. Ergänzen Sie die Implementierung der getesteten Funktion (Methode/Konstruktor) jeweils nur um so viel Code, dass die Testmethode ausführbar ist. Oft reicht als erster Schritt eine leere Methode (ggf. mit Dummy-Rückgabewert) aus.
3. Führen Sie die Testklasse aus. Neu erstellte Tests sollten zunächst fehlschlagen, da die korrekte Implementierung in der getesteten Klasse noch fehlt.
4. Implementieren Sie erst dann die getestete Klasse genau so weit, dass der jeweils neue Testfall erfolgreich ist.

Der Schritt ist erfolgreich erledigt, wenn Sie

1. mindestens 5 Testfälle implementiert haben, und
2. die Implementierung der getesteten Klasse so weit fortgeschritten ist, dass alle Testfälle erfolgreich ablaufen und
3. die Tests an der Konsole ausgeführt werden können

**Abtestat**

Führen Sie das Resultat ihrer erarbeiteten Lösung vor. Erläutern Sie Ihre Lösung, benennen Sie Ihre Entscheidungsoptionen und begründen Sie Ihre Entscheidungen. Seien Sie bereit auf Fragen zu Umsetzungsalternativen, ihrem konkreten Lösungsweg und ihrer konkreten Lösung ausführlich antworten zu können.

**Fragen & Antworten**

- Wann gilt das Praktikum als bestanden? Sie haben ein Teiltestat bestanden, wenn Sie alle Schritte erfolgreich bearbeiten und das Abtestat bestehen, d.h. die gestellten Fragen korrekt und ausführlich beantworten können.
- Sie haben Fragen bei der Bearbeitung des Praktikums, bei einer konkreten Aufgabe oder einem konkreten Bearbeitungsschritt? Dann nutzen Sie auch das Forum in Moodle, sicherlich gibt es Gruppen mit den selben Fragen und häufig auch gute Antworten und Lösungsideen in anderen Gruppen.

Viel Erfolg!