KBE-Beleg 1: TestRunner

Fertigstellungstermin: Ihre Übung am Di, 05.11. (Zug 1) bzw. Ihre Übung am Mi, 06.11. (Zug 2)

User Story: Als Entwickler_in möchte ich einen Prototyp für ein Unit-Testing-Framework erstellen (Wir nehmen mal an, so etwas gibt es noch nicht). Ihr Framework soll die Annotation `MyTest` und einen TestRunner zur Verfügung stellen. Nutzer_innen Ihres Prototyps sind SW-Entwickler_innen. Der TestRunner soll eine von den Entwickler_innen geschriebene Klasse laden und die Methoden, die mit `@MyTest` annotiert sind, ausführen. Der TestRunner kann nur Methoden mit folgenden Eigenschaften ausführen:

- mit `@MyTest` annotiert
- `public`
- ohne Argumente
- mit Rückgabewert vom Typ `boolean`

Der Rückgabewert dieser Methoden vom Typ `boolean` zeigt an, ob ein Test erfolgreich war oder nicht.
Beispiel einer Klasse, die an Ihren TestRunner übergeben wird:

```
import htwb.ai.MyTest;
public class MyClassTest {

    @MyTest public boolean testmethod1() {
        boolean result = false;
        // teste irgendwas: result bekommt einen neuen Wert
        return result;
    }
    @MyTest public boolean testmethod2() {
        boolean result = false;
        // teste irgendwas: result bekommt einen neuen Wert
        return result;
    }
    @MyTest public boolean testmethod3() {
        boolean result = false;
        // teste irgendwas: Es wird eine Exception geworfen
        return result;
    }
}
```

`testmethod1` wird ausgeführt, Rückgabewert ist `true`, dann soll Ihr TestRunner den Test als ‚passed' dokumentieren. `testmethod2` wird ausgeführt, Rückgabewert ist `false`, dann soll Ihr TestRunner den Test als ‚failed' dokumentieren. `Testmethod3` wird ausgeführt, aber ein Fehler tritt auf, dann soll Ihr TestRunner den Test mit ‚error' und Fehlergrund dokumentieren. Die Ausgabe Ihres TestRunners muss so aussehen:

```
------- TEST RESULTS FOR MyClassTest -------
Result for 'testmethod1': passed
Result for 'testmethod2': failed
Result for 'testmethod3': error due to NullPointerException
```

Ihre Aufgaben:

1. Sie definieren die Annotation namens MyTest in `htwb.ai.MyTest.java` . Diese soll zur Laufzeit ausgewertet werden und soll nur für Methoden verwendet werden.
2. Implementieren Sie den TestRunner entsprechend den Anforderungen oben. Bedenken Sie, dass Ihre Nutzer_innen sich nicht immer an Ihre Vorgaben halten werden.
3. Ihr Programm soll so aufgerufen werden (in der Kommandozeile):
   `java -jar testrunner-1.0-jar-with-dependencies.jar -c className`
   wobei: className ist der „vollständige" Name einer Klasse, die Methoden mit @MyTest enthalten kann.
   -c mit einem Wert muss angegeben werden. Falls Ihre Nutzer_innen diesen Flag und Wert nicht vollständig angeben, muss Ihr Programm eine entsprechende „Usage"-Message ausgeben (Bitte keine Default-Klassen laden).
4. Nutzen Sie zum Parsen der Kommandozeilen-Option eine externe Bibliothek.
5. Unit-Tests. Diese werden bewertet.

Abgabe/Präsentation:

- Erfolgt in der **Übung am Di, 05.11. (Zug 1) bzw. in der Übung am Mi, 06.11. (Zug 2)**
- Ihr Code befindet ausgecheckt sich auf einem Laborrechner unter Ubuntu (**nicht unter Windows**).
- Keine Code-Änderungen und kein Einchecken von Code während dieser Übung. Die Befehle `git status & git log` werden von allen Teams vor der Präsentation ausgeführt.
- Der Befehl `mvn clean package` wird von jedem Team während Präsentation ausgeführt und läuft fehlerfrei durch.
- Ich werde Ihnen am Tag der Präsentation eine Java-Klasse geben, die Sie in Ihr testrunner-Projekt integrieren werden. Ihr TestRunner soll diese Klasse laden und die MyTest-Methoden ausführen. Wir werden uns die Ausgabe Ihres Programms zusammen angucken.