ÜbunGszettel TDD

# Aufgabe 1: Theorie

Arbeite dich in die Theorie zum TDD ein. Nach dieser Aufgabe musst du folgende Punkte genauestens erklären können:

* Testdriven Development
* Red-Green-Refactor
* FIRST-Acronym
* Kent Beck
* Testpyramide
* Testarten
  + Funktionale vs. Nicht Funktionale Tests
  + Unit-Tests (Sociable, Solitary, Mocks)
  + Integrationstests
  + UI-Tests / EndToEndTests / Systemtests
  + Akzeptanztests
* JUNIT (inbs. auch die Neuerungen in Junit5)
* Mockito (Sinn und Funktionsweise von Mocking-Bibliotheken)

# Aufgabe 2: BlueJ-Buchabschnitte zu TDD

Arbeite im BlueJ-Buch die Abschnitte 9.2, 9.3 und 9.4 durch (incl. allen Übungen)

# Aufgabe 3: Ausgangsprojekt

Laden Sie sich das gegebene Maven-Ausgangsprojekt (tdd.zip) herunter (enthält ein Maven-Projekt mit allen nötigen Dependencies für JUNIT5 und Mockito, ohne Testrunner-Config, d.h. die Verwendung müsste über eine IDE verfolgen, die die Tests direkt ohne Testrunner ausführen kann, z.B. IntelliJ), laden Sie es als Maven-Projekt in ihre IDE und schauen Sie sich an, wie das Projekt aufgebaut ist:

* pom.xml (Depencendies, Java-Version etc.)
* Gegebene Domänen-Klassen (Kinosaal, Ticket etc.)
* Gegebene Start-Junit5-Tests in test /java/at.itkolleg/AppTest

Starten Sie den Test AppTest über den grünen Pfeil und versichern Sie sich, dass alles korrekt läuft. Starten Sie auch die App (main-Methode).

# Aufgabe 4: Einarbeitung in den gegebenen Code

Arbeiten Sie sich in den gegebenen Code zur Kinoverwaltung ein. Verwenden Sie die gegebenen Klassen KinoSaal, Ticket, Vorstellung, Kinoverwaltung in der App-Klasse (main-Methode), um ein Gefühl für die Funktionsweise des Programms zu bekommen. Führen Sie folgende Punkte durch:

* Kinosäle anlegen
* Vorstellungen anlegen
* Vorstellungen über die Kinoverwaltung einplanen
* Tickets für Vorstellungen ausgeben
* etc.

# Aufgabe 5: JUNIT-Tests für KinoSaal

Testen Sie alle Methoden der Klasse KinoSaal (Testklasse TestKinoSaal).

# Aufgabe 6: JUNIT-Tests für Vorstellung

Testen Sie alle Methoden der Klasse Vorstellung (Testklasse TestVorstellung).

# Aufgabe 7: JUNIT-Tests für KinoVerwaltung

Testen Sie alle Methoden der Klasse KinoVerwaltung (Testklasse TestKinoverwaltung).

# Aufgabe 8: JUNIT-Tests Advanced 1

Falls nicht schon in den vorhergehenden Aufgaben passiert, testen Sie folgende Punkte unter Verwendung der fortgeschrittenen Features von JUNIT 5:

1. Schreiben Sie einen Test, der validiert, dass das Anlegen einer Vorstellung korrekt funktioniert. Der Test sollte eine fachliche Bezeichnung haben und die Assertions sollten bei Validierungsfehler eine Hinweistext liefern.
2. Schreiben Sie einen Test, der validiert, dass das Einplanen mehrerer Vorstellungen korrekt funktioniert. Stellen Sie zudem sicher, dass beim möglichen Auftreten eines Fehlers trotzdem alle Validierungen ausgeführt werden.
3. Schreiben Sie einen Test, der sicherstellt, dass ein Fehler geworfen wird, wenn eine Veranstaltung doppelt eingeplant wird.
4. Schreiben Sie einen parametrisierten Test, der mehrere Ticketkäufe mit unterschiedlichen Parametern überprüft.
5. Schreiben Sie eine dynamische TestFactory die den Ticketkauf mit zufälligen Werten bombardiert. Der Test soll sicherstellen, dass der Ticketkauf entweder funktioniert oder nur einen der definierten Fehlermeldungen (z.B. new IllegalArgumentException("Nicht ausreichend Geld.")) ausgibt. Die Tests müssen reproduzierbar sein.

# Aufgabe 9: Mockito Einführung

Lesen Sie sich in das Mocking-Framework Mockito ein (Links siehe Moodle). Startpunkt sollte folgendes Tutorial sein: <https://www.youtube.com/watch?v=aNCPy803rkI> .

Verwenden Sie die wesentlichen Mockito-Möglichkeiten praktisch in kleinen Programmen.

# Aufgabe 10: Testen einer Spring BOOT Applikation

Im folgenden Tutorial bekommen Sie gezeigt, wie man mit Mockito und JUnit eine SpringBoot-App testet:

<https://www.youtube.com/watch?v=8Vd3MYml2Js>

Schaue Sie sich dieses Tutorial genau an.

Erstellen Sie danach für möglichst alle ihre Layers in ihrer Spring-Boot App aus der Fullstack-Schulübung vom vorhergehenden Modul mindestes 2 Tests unter Verwendung von JUNIT und Mockito.