Einstiegsbeispiel (Übungsblatt 0)

Ausgabe: 15.10.2021 Abgabe: 22.10.2021 5 Punkte

Bitte laden Sie die Übungsbeispiele rechtzeitig am Sonntag vor der Übungsstunde bis spätestens 23:55 als Zip Datei (`<Matrikelnummer>_<Name>_SE1-0.zip`) in Moodle hoch und tragen Sie Ihre Kreuze in ZEUS ein. Da es in ZEUS keine halben Punkte gibt, wurde die Anzahl der Punkte für ZEUS verdoppelt → laut ZEUS können Sie für diese Übung 10 Punkte bekommen. Für die Übung selbst werden aber die am Übungsblatt angegebenen Punkte gezählt → es können maximal 5 Punkte erreicht werden (= Anzahl der Punkte in ZEUS / 2). Sollten Sie Fragen zur Übung haben oder Unklarheiten auftreten, nutzen Sie bitte das Moodle-Forum.

Hinweis: Das Einstiegsbeispiel dient auch als Self-Assessment, um einschätzen zu können, ob Sie die notwendigen Voraussetzungen für die LV bereits mitbringen. Sie müssen daher das Einstiegsbeispiel *vollständig* lösen, um am Kurs teilnehmen zu können. Bei Nicht-Abgabe oder unvollständigen Lösungen, scheiden Sie aus dem Kurs aus!

Abgabedateien

Die Zip Datei `<Matrikelnummer>_<Name>_SE1-0.zip` muss folgende Dateien beinhalten (bitte VOR der Abgabe prüfen):

- `.git`
- `uebung-0-<Matrikelnummer>/pom.xml`
- `uebung-0-<Matrikelnummer>/src/main/java/Einstiegsbeispiel.java`
- `uebung-0-<Matrikelnummer>/src/test/java/EinstiegsbeispielTest.java`

Jede abgegebene Datei muss als Kommentar ganz oben in der Datei Ihren vollständigen Namen und Ihre Matrikelnummer beinhalten!

0.1 Setup

Links zur Installation und zu hilfreichen Tutorials finden Sie am Ende des Angabezettels.

1. Installieren Sie Git.
2. Erstellen Sie in IntelliJ ein Maven-Projekt mit der *groupId* `se1.aau` und der *artifactId* `uebung-0-<Matrikelnummer>`. Navigieren Sie in der Konsole zum soeben erstellten Projekt-Ordner `uebung-0-<Matrikelnummer>` und erstellen Sie in diesem Ordner ein lokales Git-Repository mit dem Befehl `git init`. Erstellen Sie im Ordner `src/main/java` eine Java-Klasse mit dem Namen `Einstiegsbeispiel.java` und im Ordner `src/test/java` die Java-Klasse `EinstiegsbeispielTest.java`.
3. Danach fügen Sie den Projektordner, die Klassen und ihre Ordner mit `git add <filename>` bzw. mit `git add *` dem Git-Repository hinzu. Überprüfen Sie, ob alle Dateien hinzugefügt worden sind mit dem Befehl `git status`.
4. Committen Sie die neu hinzugefügten Dateien mit dem Befehl `git commit -m '<message>'`. Geben Sie dem Commit mit der Message eine sinnvolle Beschreibung.

Im Folgenden werden Sie eine einfache Methode implementieren und testen. Committen Sie Ihre Änderungen nach jeder Aufgabe mit `git commit -am '<message>'`.

0.2 Implementierung

Implementieren Sie **eine** der folgenden Methoden und zwar diejenige, die Ihrer Matrikelnummer zugeordnet ist. Wenn Ihre Matrikelnummer 12345678 lautet und somit auf *78* endet, implementieren Sie die Berechnung des Skalarproduktes.

MatNr	Aufgabe
00-25	Hamming Distanz berechnen
26-50	Umrechnen von Binärzahlen in Dezimalzahlen
51-75	Inneres Produkt (Skalarprodukt) zweier Vektoren berechnen
76-99	Erkennen sich wiederholender Sequenzen

Lesen Sie die Beschreibung genau durch und implementieren Sie die Methode, die Ihrer Matrikelnummer zugeordnet ist. Denken Sie auch an etwaige Fehlerfälle, auf die Rücksicht genommen werden muss. Committen Sie Ihre Änderungen im Code sinnvoll nach jedem Schritt!

Erkennen sich wiederholender Sequenzen

- Methodenname: getRepetitions
- erwarteter Input: ein Array mit sich wiederholenden Sequenzen z.B. [1,1,1,2,2,2,2,3,4,4,4]
- erwarteter Output: eine Map mit dem jeweiligen Wert als Key und der Anzahl der Wiederholungen als Value.
  z.B. {1→3, 2→4, 3→1, 4→3}

Hamming Distanz berechnen

- Methodenname: getHammingDistance
- erwarteter Input: zwei mit Booleans befüllte Arrays gleicher Länge
  z.B. [true true true] und [false false true]
- erwarteter Output: Integer, der der Summe der Unterschiede in den Arrays entspricht
  z.B. 2

Umrechnen von Binärzahlen in Dezimalzahlen

- Methodenname: binary2decimal
- erwarteter Input: Variable vom Typ Integer, die eine Binärzahl darstellt z.B. 1110_2
- erwarteter Output: Integer, der der Dezimalzahl des Inputs entspricht z.B. 14_{10}

Inneres Produkt (Skalarprodukt) zweier Vektoren berechnen

- Methodenname: getInnerProduct
- erwarteter Input: zwei mit Integer-Werten befüllte Arrays gleicher Länge
- erwarteter Output: ein Integer, der dem inneren Produkt beider Vektoren entspricht

0.3 Testen mit JUnit

Binden Sie die Dependencies `junit-jupiter-api` und `junit-jupiter-engine` für JUnit 5.1.0 mit einem entsprechenden Eintrag im `pom.xml`-File ein, siehe Video-Tutorial. Lesen Sie die Tutorials über Testen mit JUnit durch.

Testen von Normalfällen

Schreiben Sie zwei Tests in die Datei `EinstiegsbeispielTest.java`, die die Funktionalität Ihrer zuvor implementierten Methode bei der Eingabe korrekter Werte überprüfen. Die Testmethode wird mit `@Test` annotiert. Verwenden Sie Assertions, wie `assertTrue` bzw. `assertEquals`, um das Verhalten Ihrer Methode zu überprüfen. Committen Sie Ihre Änderungen sinnvoll!

- *Korrekte Eingabewerte* für die Methode `addPositiveInteger(int a, int b)`, die die Summe zweier positiver ganzer Zahlen `a` und `b` berechnen soll, sind beispielsweise $a = 5$ und $b = 20$. Gilt das für beliebig große Zahlen?

Testen von Fehlerfällen

Schreiben Sie zwei weitere Tests, die das korrekte Verhalten Ihrer Methode bei *nicht* korrekten Eingaben überprüfen. Funktioniert Ihre Methode richtig? Passen Sie gegebenenfalls die Implementierung an. Geben Sie im Fehlerfall mittels Exceptions, die Sie eventuell werfen, sinnvolle Nachrichten an den Aufrufer zurueck. Am Ende sollten alle ihre Tests grün durchlaufen und die implementierte Methode auf ihr Verhalten bei korrekten sowie nicht korrekten Eingabewerten überprüfen. Committen Sie Ihre Änderungen wieder sinnvoll!

- *Nicht korrekte Eingabewerte* für die Methode `addPositiveInteger(int a, int b)` sind beispielsweise negative Werte wie $a = -5$ oder $b = -20$. Bei der Eingabe von $a = -5$ und $b = 20$ wird z.B.die Nachricht `Die Eingabe ist nicht korrekt! Beide Eingabewerte müssen größer als 0 sein!` mit einer eventuellen Exception retourniert.

Hilfreiche Links

- Git Tutorials, vor allem sind Learn Git, Beginner, sowie Getting Started interessant
- Working with Maven in IntelliJ IDEA
- Setting the `-source` and `-target` of the Java Compiler
- Erste Testfälle mit JUnit
- JUnit User Guide, vor allem das Kapitel 3