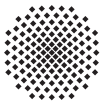

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 14

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: — — Besprechung: —

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studenten**.
- Dieses Übungsblatt beinhaltet nur einen Teil (A). Teil A ist in der Präsenzübung zu lösen. Es gibt keinen Heimarbeitsteil.

Lernziel: Dieses Übungsblatt dient dazu, dass Sie die in der Vorlesung vorgestellten Konzepte zum Programmablauf und den Kontrollstrukturen vertiefen und anwenden. Zudem kennen Sie bereits ein umfangreiches Set von Stilrichtlinien, welche Sie anwenden, üben und vertiefen sollen.

Vorbereitung: Stellen Sie sicher, dass Sie sich mit den Vorlesungsunterlagen zu Weakest Preconditions, Streams und Lamdas vertraut gemacht haben und die Übungsblätter 12 und 13 absolviert haben.

Punkte: Diese Übungsblatt enthält einen Teil. Teil A mit 2 Aufgaben.

Style: Bitte halten Sie die in der Vorlesung vorgestellten Style-Regeln ein. Dazu gehören auch JavaDoc sowie Vor- und Nachbedingungen. Der Style Ihrer Implementierung wird mit bis zu 50% bewertet.

Unterlagen: Alle relevanten Unterlagen, für die Bearbeitung des Übungsblatts finden Sie wie gehabt in unserem git-Repository: `https://github.com/RSS-PSE-WS1920/`

Viel Erfolg!

1 Teil A - Präsenzaufgaben

Aufgabe 1 Hamster starten

Lernziel: Ziel dieser Aufgabe ist es, alles Gelernte zusammenzubringen und anzuwenden.

In vielen Übungen haben wir den RSS Hamstersimulator genutzt. Bisher waren immer Teile dieser Nutzung vorgegeben. In dieser Aufgabe sollen Sie nun all Ihr erworbenes Wissen kombinieren und den Hamstersimulator selbst von Grund auf ansteuern und Paule eine einfache Aufgabe lösen lassen. Dazu benötigen Sie, wie immer, die API des Hamstersimulators.

(a) Beginnen Sie damit, eine Main-Operation anzulegen, die Sie nutzen, um den Simulator zu instantiieren, zu konfigurieren und laufen zu lassen. Als erstes benötigen Sie noch ein HamsterGame Objekt. Dieses bekommen Sie nur, wenn Sie vorher die JAR-Bibliothek des Hamstergames in Ihren Build-Path aufnehmen. Die JAR des Hamstersimulators finden Sie in unserem git-Repo für diese Übung (`https://bit.ly/2FX8Lrn`). Eine Anleitung zum Hinzufügen von Jars zum Build Path findet sich u.a. unter `https://www.youtube.com/watch?v=E1HTwMJhWVA`.

(b) Holen Sie sich vom HamsterGame Objekt den TerritoryBuilder. Damit können Sie das Territory per API definieren. Bauen Sie ein Territory der Größe 5 Spalten auf 3 Zeilen, umranden Sie es mit Wänden und platzieren Sie Paule auf die Location (1,1). Außerdem legen Sie noch ein Korn auf die Location bei Zeile 1, Spalte 3. Am Ende müssen Sie das `HamsterGame` initialisieren, indem Sie dem Game bei der `initialize`-Operation das Builder-Objekt übergeben.

(c) Initialisieren Sie JavaFX über die Klassenoperation `start` der JavaFXUI Klasse. Starten Sie die Game UI, indem Sie die passenden Operationen in der Hamster Game API suchen. Konfigurieren Sie dabei auch des Input Interface (was für Paules Read und Write Operationen zuständig ist). Dies können Sie auch via Klassenoperation von der JavaFXUI abfragen.

(d) Starten Sie das Hamsterprogramm über die `startGame`-Operation, ohne dass das Spiel pausiert ist.

(e) Schreiben Sie ein Hamsterprogramm in eine Operation `runHamster`. Paule soll dabei einfach das eine Korn aufsammeln. Starten Sie dann das Programm über die `runGame`-Operation. Gestalten Sie Ihr runHamster so, dass es über einen Lambda Ausdruck übergeben werden kann.

(f) Beenden Sie das HamsterGame durch die passende stop-Operation.

Aufgabe 2 Streams

In dieser Aufgabe sollen Sie die Konzepte der Collection-Streams in Java vertiefen und vor allem anwenden. Dazu liegt im git eine Testumgebung bereit.

(a) Importieren Sie das Projekt aus dem git-Repository in Ihr Eclipse.

(b) In der Klasse `StreamsAndFilterExercise` finden Sie die `main`-Operation und die Operationen, die Sie im Folgenden füllen sollen. Machen Sie sich zunächst mit der Implementierung vertraut. Schauen Sie sich dabei die Klassen `StudentRecord, Student` und `Exam` an.

(c) Nutzen Sie Streams, um sich die Namen aller Studenten auf der Konsole ausgeben zu lassen.

(d) Nutzen Sie Streams, um sich die Anzahl der Studenten mit einem Alter über X auszugeben.

(e) Nutzen Sie Streams, um sich die Namen der Studenten ausgeben zu lassen, die über X Jahre alt sind.

(f) Nutzen Sie Streams, um sich die Namen aller Studenten ausgeben zu lassen, die über X Jahre alt sind und die Klausur Y nicht bestanden haben.

(g) *Optional:* Implementieren Sie die Aufgaben C-F ohne die Nutzung von Streams.

Aufgabe 3 Optional: Weakest Preconditions

Lernziel: zu gegebenem Code die schwächste Vorbedingung bestimmen können

Betrachten Sie dazu den folgenden Code:

```
int faculty(int n) {
    int result = 1;
    int i = 0;
    // invariant \result = i!; variant n - i;
    while (i < n) {
        i = i + 1;
        result = result * i;
    }
    return result;
}
```

(a) Bestimmen Sie wp(Code,\result = n!). Annotieren Sie die Beweisschritte mit den angewendeten Regeln und schreiben Sie auch alle Zwischenergebnisse als Hoare-Tripel auf, wie wir es in der Vorlesung gelernt haben. Sie dürfen in Ihrer Herleitung bei Anwendung der Assignment-Regel die beschränkte Darstellung von Integern ignorieren.

(b) **Herausforderung** Ändern Sie jetzt den Beweis so ab, dass Sie die begrenzte Darstellung von Integer berücksichtigen. Welche geänderte WP ergibt sich dadurch?

Hinweise zu den Übungen:

- Die Abgabe erfolgt im ILIAS.
- Durch die Teilnahme am Übungsbetrieb können Sie sich für die Teilnahme an der Klausur qualifizieren.
  - Bestehen von min. 80% aller Übungsblätter.
  - Ein Übungsblatt gilt als bestanden, wenn 50% der Punkte erreicht wurden.
  - Teilnahme an min. 80% der Übungen.
  - Bestehen der Scheinklausur.

Viel Erfolg!