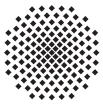

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 5

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 22.11.2019, 23:59 Uhr — Besprechung: KW 49

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studenten**.
- Dieses Übungsblatt besteht aus zwei regulären Teilen (A+B) und zwei Vorbereitungsteilen. Teil A ist in der Präsenzübung zu lösen. Teil B ist in Heimarbeit (Gruppe von 2 Studenten) zu lösen und rechtzeitig abzugeben.
- Für alle Aufgaben aus Teil B erstellen Sie bitte **ein** PDF-Dokument. Dieses Dokument sollte ein Titelblatt enthalten, aus dem alle Metadaten entnommen werden können (siehe nächster Punkt).
- Geben Sie zu Beginn der Datei Ihre Namen (Vor- und Nachname), die Matrikelnummern und die E-Mail-Adressen an.
- Benennen Sie die Datei nach dem folgenden Schema:
  PSE[**ÜBNr**]-[**NachnamenTeammitglieder**]-[**NachnameTutors**]**.pdf**.

Lernziel: Dieses Übungsblatt dient dazu, Sie mit den grundlegenden Prinzipien von Objekterzeugung so wie logischen Ausdrücken (Präsenzaufgaben) und Verträgen (Heimaufgaben) vertraut zu machen.

Vorbereitung: Stellen Sie sicher, dass Sie die Übungsblätter 1 bis 4 absolviert haben, alle Software installiert und funktionsfähig ist (Java 1.11), und Ihr W-Lan richtig konfiguriert ist.

Punkte: Dieses Übungsblatt enthält vier Teile. Zwei Vorbereitungsteile zur Einführung von Eclipse. Teil A mit 5 Aufgaben und Teil B mit 2 Aufgaben. Im Teil B können Sie bis zu 40 Punkte erzielen.

1 Teil 0 - Vorbereitung

Aufgabe 1 Vorbereiten der IDE

Für den kommenden Übungsverlauf werden wir die IDE Eclipse benutzen. Um einen reibungslosen Ablauf der Übung zu gewähren, führen Sie folgende Schritte bereits vor der Übung aus:

(a) Laden Sie sich die aktuelle Version von Eclipse herunter: `https://www.eclipse.org/downloads/`. Achten Sie darauf, die richtige Eclipse-Version herunterzuladen oder im Installations-Wizard auszuwählen (Eclipse IDE for Java Developers).

(b) Importieren Sie die bereitgestellten Projekte `Exercise5-presence` und `Exercise5-homework` in Ihre Eclipse IDE. Befolgen Sie dazu die referenzierte Anleitung (Video: `https://youtu.be/38JFCqi_X3c`, Text: "Teil 1 - Importieren des git Projekts in Eclipse"). Prüfen Sie am Ende, dass Sie für die Präsenzaufgaben fünf Klassen vorfinden. Prüfen Sie auch, dass Ihr Eclipse JDK 11 zum Kompilieren benutzt.

(c) Beachten Sie, dass für einen reibungslosen Ablauf Eclipse mit Java 11 gestartet und konfiguriert sein muss. Sollten Sie mehrere Java Versionen installiert haben, kann es zu Problemen kommen, wenn nicht die richte Version in Eclipse eingestellt ist.

Zudem kann es vorkommen, dass die vm Variable nicht gesetzt ist. In dem Fall beachten Sie bitte die Hinweise aus dem Eclipse Wiki Seite `https://wiki.eclipse.org/Eclipse.ini`.

2 Teil 1 - Importieren des git Projekts in Eclipse

Um das Eclipseprojekt aus dem git in Eclipse zu importieren, führen Sie bitte folgende Schritte aus:

1. In Eclipse wählen Sie: File → Import...
2. Öffnen Sie im Dialog den Ordner “Git” und wählen Sie “Projects from Git” aus. Drücken Sie dann auf Next >.
3. Wählen Sie “Clone URI” aus und drücken Sie wieder auf Next >.
4. Kopieren Sie die URL `https://github.com/RSS-PSE-WS1920/exercise-sheet-5-presence-partA` und fügen Sie diese in Eclipse unter URI ein. Drücken Sie dann auf Next >.
5. Im nächsten Fenster sollte “master” bereits ausgewählt sein. Falls dies nicht der Fall sein sollte, setzen sie die einen Hacken bei “master” und drücken Sie dann auf Next >.
6. Nun können Sie im Feld “Directory” angeben, wo das Projekt gespeichert werden soll. Drücken Sie dannach wieder auf Next >.
7. Wählen Sie “Import existing Eclipse projects” und klicken Sie auf Finish.
8. Eventuell erscheint ein Dialog, der Sie fragt, ob sie eine module-info erstllen wollen. Diesen können Sie mit Don’t Create schließen.
9. Stellen Sie sicher, dass “JavaSE-11” als execution environment ausgewählt ist.

Wechseln Sie im Anschluss in die Java-Perspektive: Window → Perspective → Open Perspective → (evtl Other... →) Java (default).
Hinweis: Sollte dieser Schritt bei Ihnen den Fehler „Connecting Git team provider failed“ anzeigen, überprüfen Sie ob das Projekt an den gewünschten Ort heruntergeladen wurde. Sollte dies der Fall sein, können Sie über File → Import... → General → Existing Project into Workspace → Next > das Projekt importieren. Andernfalls wählen Sie statt Projekt aus Git „Projekt aus Git (with smart Import)“ und folgen Sie dem Importprozess.

3 Teil A - Präsenzaufgaben

In dieser Übung werden Sie die Inhalte zur Vorlesung Logik sowie Objekterzeugung vertiefen. Die Dokumentation der Hamster-API finden Sie wie immer hier:
`http://caloundra.informatik.uni-stuttgart.de` (nur aus dem Uni-Netz erreichbar).

Aufgabe 1 Logik

In dieser Aufgabe geht es um die Grundlagen der Logik – unter anderem in Form von Wahrheitstabellen.

(a) Gegeben sind folgende logische Beschreibungen:

- Die Anzahl der Körner auf dem Territorium ist größer als 5.
- Paule schaut nach Norden
- Paule schaut nicht nach Norden und hat Körner im Mund.
- Paule hat keine Körner im Mund und schaut nicht nach Norden und die Anzahl der Körner auf seinem Feld ist nicht 5 oder weniger.

Schreiben Sie zu jeder Aussage den passenden JML Ausdruck auf. Nehmen Sie dabei an, dass es einen Hamster `paule` und ein Territorium `territory` gibt.

(b) Nehmen Sie an, Sie sollen eine Operation implementieren, die zurückgibt, ob Paule glücklich ist oder nicht. Ob Paule glücklich ist oder nicht, kann mit folgendem Ausdruck beschrieben werden:

```
1 !paule.mouthEmpty() &&
2 (paule.getLocation() == paula.getLocation() ||
3 territory.getNumberOfGrainsAt(paule.getLocation()) >= 1)
4 <==> paule.isHappy()
```

Übersetzen Sie den Ausdruck in natürliche Sprache und diskutieren Sie mit Ihrem Team, wann Paule glücklich ist.

Hinweis: Nehmen Sie an dass paule.getLocation() == paula.getLocation() wahr ist, wenn beide auf der selben Kachel stehen.

(c) Erstellen Sie eine Wahrheitstabelle für den obigen Ausdruck. Zerlegen Sie dazu den Ausdruck zunächst in die einzelnen boolschen Ausdrücke. Jeder Ausdruck soll dann in einer Spalte der Tabelle erscheinen und mit jeder Kombination von `true` und `false` ausgewertet werden.

Aufgabe 2 Objekterstellung

(a) Öffnen Sie das git-Repo für Übungsblatt 5 (exercise-sheet-5-presence-partA). URL: `https://github.com/RSS-PSE-WS1920/`.
Eine Anleitung hierzu finden Sie in Teil 1 (siehe oben).

(b) Erstellen Sie einen neuen Hamster in der (diagonal) gegenüberliegenden Ecke des Territoriums. Nennen Sie die Objektvariable Ihres neuen Hamsters `paula`. Bewegen Sie Paule durch das Territorium zu Paula. Lassen Sie ihn dabei alle Körner aufsammeln. Sobald Paule bei Paula angekommen ist, soll er alle Körner auf dem Feld mit Paula ablegen. Paula soll dann alle Körner aufsammeln.

Bearbeiten Sie dazu die Klasse `PaulesNewFriendGame`. Überlegen Sie sich zuerst, wie Sie, analog zur Vorlesung, an die nötigen vier Argumente für den Hamsterkonstrukor kommen. Legen Sie lokale Variablen an und speichern die Argumente vorher in diesen Variablen – ebenfalls analog zur Vorlesung. Überlegen Sie sich durch Suchen in der API, wie Sie an die jeweiligen Werte bzw. Objektreferenzen kommen.

Um ihr Programm in Eclipse auszuführen, können Sie z.B. auf Ihre Java Klasse im Package Explorer Rechts klicken und dann "Run As.." → "Java Application" auswählen. Speichern Sie vorher Ihre Datei. Eclipse kompiliert die Klasse im Hintergrund automatisch, Sie müssen nicht wie bei BlueJ "Übersetzen" auswählen.

(c) Erweitern Sie ihren Code so, dass, sobald sich Paule und Paula treffen, zwei neue Hamster erstellt werden (Paulchen und Paulina), die jeweils "neben" Paula auf das Spielfeld gesetzt werden. Gehen Sie beim Programmieren dazu wieder analog zum Vorgehen im vorherigen Aufgabenteil vor.

Aufgabe 3 Packages und Imports

(a) Überlegen Sie sich für jede Klasse in Ihrem Eclipse Projekt einen passenden Paketnamen und fügen Sie diesen Ihrer Implementierung hinzu.

(b) Importieren Sie die Standardklasse `StringBuilder` in Ihre Klasse `PaulesNewFriendGame`. Suchen Sie die Informationen zur Klasse `StringBuilder` hier: `https://docs.oracle.com/en/java/javase/11/docs/api/index.html`.

(c) Machen Sie sich mit der Dokumentation vertraut und überlegen Sie sich, wie Sie die Klasse `StringBuilder` einsetzen können, um die Aufgabe 2 a) von Übungsblatt 3 - Teil A zu lösen. Sie sollen dabei keinen "+"-Operator nutzen.

Aufgabe 4 Herausforderung I: Heart Game

Öffnen Sie die Klasse `PaulesHeartGame` und lassen Sie Paule mithilfe eines Algorithmus' ein Herz aus Körnern auf das Territorium legen. Beachten Sie, dass Paule in diesem Spiel bereits 10 Körner im Maul hat.

Aufgabe 5 Herausforderung II: Pathfinder

Ziel dieser Aufgabe soll es sein, einen Algorithmus in der Klasse `PathFinderGame` zu schreiben, der Paule automatisch durch das Teritorrium zu Paula laufen lässt.

(a) Überlegen Sie sich also zunächst einen geeigneten Algorithmus.

(b) Überlegen Sie sich, wie ihr Algorithmus mithilfe einer for-Schleife umgesetzt werden kann. Das Abbruchkriterum der Schleife soll sein, dass Paule bei Paula angekommen ist.

(c) Schreiben Sie eine Operation, die ihren Algorithmus implementiert.

(d) Testen Sie Ihren Algorithmus mit verschiedenen Territorien.

Hinweis: Sie können sich merken, auf welchen Feldern Sie bereits waren, indem Sie dort ein Korn platzieren.

4 Teil B - Heimarbeit

Aufgabe 1 Klassen, Objekte und Schnittstellen

Ihr Software-Unternehmen bekommt den Auftrag, eine Home-Banking-Software zur Verwaltung von Familienkonten zu entwickeln. Für jedes Familienmitglied kann ein eigener Benutzerzugang eingerichtet werden. Für Eltern gibt es genau ein gemeinsames Konto, auf das beide Eltern Zugriff haben. Kinder haben ein eigenes Konto. Das Elternkonto kann so konfiguriert werden, dass es entweder gar nicht oder um max. 100 Euro ins Minus gehen kann. Konten von Kindern dürfen niemals einen negativen Betrag aufweisen. Die Software erlaubt es, Nachrichten zwischen Benutzern zu verschicken.

Mittels der Software soll zum Beispiel folgendes Szenario umzusetzen sein: Hannelore und Ernst Stäbler haben die Kinder Susi und Tom. Alle Familienmitglieder haben einen eigenen Zugang zur Software. Vater Ernst konfiguriert sein Konto so, dass er 100 Euro ins Minus gehen kann. Dann überweist er Susi und Tom jeweils 30 Euro für ihre Klassenfahrt. Tom schuldet seiner Schwester noch 10 Euro und überweist ihr diesen Betrag. Tom nutzt die Nachrichtenfunktion, um seine Schwester über die Überweisung zu informieren. Susi löscht die Nachricht und loggt sich aus.

In dieser Aufgabe sollen Teile der Software — in Form von Klassen und Schnittstellen — skizziert werden, um schließlich das genannte Szenario umzusetzen.

(a) *(5 Punkte)* Nennen Sie die Namen von mindestens fünf Klassen, die in der Software zu finden sein könnten. Geben Sie außerdem eine kurze Dokumentation dessen an, welchen Zweck die jeweilige Klasse erfüllt.

(b) *(7 Punkte)* Geben Sie für die zur Umsetzung des obigen Szenarios benötigten Klassen Kommandos und Abfragen (inkl. Dokumentation) an. Es müssen mindestens zwei Klassen und insgesamt vier Kommandos sowie drei Abfragen definiert werden. Jede dieser Klassen muss über mindestens eine Operation verfügen. Sie müssen in dieser Teilaufgabe (noch) keine Vor- und Nachbedingungen beschreiben.

(c) *(4 Punkte)* Überlegen Sie sich für zwei Kommandos Vor- und Nachbedingungen (kein JML nötig). Sie können Ihre oben definierten Abfragen nutzen.

(d) *(4 Punkte)* Geben Sie Ausdrücke in Java-Syntax an, die die Schritte des obigen Szenarios mittels Ihrer Klassen und Schnittstellen bzw. Operationen und Objekte umsetzen. Es ist kein durchgängiger Programmablauf nötig. Geben Sie an mindestens zwei Stellen an, was die dann gültigen Rückgabewerte von Abfragen wären.

Aufgabe 2 Verträge

Ziel dieser Aufgabe ist es, die Definition von Verträgen (ohne und mit JML) zu üben. Hierzu betrachten wir — wie in der Aufgabe 2 des vorigen Arbeitsblatts — in dieser Aufgabe die Klasse `Hamster.java`, die Ihnen diesmal unter folgendem Link zur Verfügung steht: `https://github.com/RSS-PSE-WS1920/exercise-sheet-5-homework-partB`

(a) *(4 Punkte)* Definieren Sie für die folgende Abfrage Vor- und Nachbedingungen in natürlicher Sprache oder in JML-Syntax:

- `grainAvailable`

(b) *(6 Punkte)* Definieren Sie analog für die folgenden Kommandos Vor- und Nachbedingungen:

- `putGrain`
- `turnRight` (hier nehmen wir an, dass das Kommando existiert)

(c) *(4 Punkte)* In der Vorlesung wurden in JML definierte Vor- und Nachbedingungen für das Kommando `move` vorgestellt, die ebenfalls in der Datei `Hamster.java` enthalten sind. Erstellen Sie analog die Vor- und Nachbedingungen für ein Kommando `moveBackwards` in JML-Syntax. Das Kommando soll den Hamster eine Position nach hinten setzen. Nach Ausführung des Kommandos schaut der Hamster in die gleiche Richtung wie zuvor.

Sofern nötig, dürfen Sie selbst erfundene Abfragen verwenden, wenn Sie kurz beschreiben, was diese zurückgeben würden.

Die JavaDoc-Beschreibung des Kommandos sei wie folgt:

```
1  /**
2   * Move the hamster one step against its looking direction.
3   */
```

Listing 4: Pre- und postcondition for `moveBackwards`

(d) *(6 Punkte)* Nennen Sie drei Klasseninvarianten für die Klasse `Territory` (kein JML nötig).

5 Teil L - Lehramtsteil

Dieser Teil enthält eine Präsenzaufgabe die speziell für Lehramtsstudierende konzipiert ist. Das bedeutet allerdings nicht, dass "Nicht-Lehramtsstudierende" diese nicht auch bearbeiten dürfen. Sie bezieht sich auf den Vorlesungsstoff und die Übungsaufgaben, betrachtet diese aber aus einer didaktischen Perspektive.

Aufgabe 1 BlueJ im Unterricht

BlueJ wurde speziell für den Einsatz im Unterricht entwickelt. Dafür unterscheidet es sich in den Funktionen von einer Entwicklungsumgebung wie eclipse. Diese Aufgabe soll speziell auf die Funktionen von BlueJ, nicht des Hamsters, eingehen.

(a) Welche Funktionen in BlueJ könnten den Schülerinnen und Schülern den Einstieg ins Programmieren – auch im Hinblick auf Objekte – erleichtern?

(b) Welche Funktionen könnten den Schülerinnen und Schülern Schwierigkeiten bereiten?

(c) Diskutieren Sie mit Ihrem Team, ob Sie einen Einsatz von BlueJ als sinnvoll erachten (unter der Voraussetzung, dass Java als Unterrichtssprache genutzt wird) und

- Wenn JA: Warum und in welchem Bereichen?
- Wenn NEIN: Warum nicht?

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!