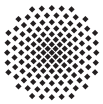

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 6

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 29.11.2019, 23:59 Uhr — Besprechung: KW 50

- Bitte lösen Sie die Übungsaufgabe in **Gruppen von 2 Studenten**.
- Dieses Übungsblatt besteht aus zwei Teilen (A+B). Teil A ist in der Präsenzübung zu lösen. Teil B ist in Heimarbeit (Gruppe von 2 Studenten) zu lösen und rechtzeitig abzugeben.
- Für Ihre Abgabe erstellen Sie bitte zwei Dateien. Erstens, eine PDF-Datei mit Titelblatt, das die unten aufgeführten Informationen enthält, sowie die Lösungen aus Teil B Aufgabe 1. Zweites, eine Java-Datei mit dem Namen `CustomHamsterGame` und der Lösung aus Teil B Aufgabe 2.
- Geben Sie zu Beginn der Dateien Ihre Namen (Vor- und Nachname), die Matrikelnummern und die E-Mail-Adressen an.
- Benennen Sie die Dateien nach dem folgenden Schema:
  PSE[**ÜB-Nr**]-[**Aufgaben-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**].[**java/pdf**].

Lernziel: Dieses Übungsblatt dient dazu, Sie mit dem grundlegenden Prinzip von Kontrollflüssen vertraut zu machen. Sie sollen lernen, wie Sie Schleifen benutzen und mit Objektreferenzen umgehen.

Vorbereitung: Stellen Sie sicher, dass Sie alle Übungsblätter 1-5 absolviert haben, alle Software installiert und funktionsfähig ist (Eclipse und Java 11), und Ihr W-Lan richtig konfiguriert ist und Sie sich die Vorlesungsunterlagen zu Klassen und Kontrollfluss sowie Objekterzeugung angeschaut haben.

Punkte: Dieses Übungsblatt enthält zwei Teile. Teil A mit 3 Aufgaben und Teil B mit 2 Aufgaben. Im Teil B können Sie bis zu 32 Punkte erzielen.

1 Teil A - Präsenzaufgaben

In dieser Übung werden Sie die Inhalte der beiden Vorlesungen „Referenzen und Objekterzeugung“ und „Kontrollflussstrukturen“ vertiefen. Insbesondere werden Sie den Umgang mit Konstruktoren, Referenzen und Schleifen üben, sowie erste eigene Algorithmen für den Hamster Paule implementieren.

Aufgabe 1 Algorithmen entwickeln und verstehen

In dieser Aufgabe geht es darum, Ihr Verständnis von Algorithmen zu vertiefen.

(a) Bitte wiederholen Sie dazu zunächst die Definition von „Algorithmus“ und diskutieren Sie mit Ihrem Partner 3 alltägliche Vorgänge, die mehrere elementare Einzelschritte beinhalten (bspw. Ihr Weg zur Universität oder das Bearbeiten eines Übungsblattes bis hin zur Abgabe).

(b) Schreiben Sie einen dieser Vorgänge natürlichsprachlich (oder in Pseudocode) im Stile eines Algorithmus auf. Bringen Sie dabei alle drei fundamentalen Kontrollstukturen des „Structured Programmings“ zur Anwendung.

(c) Tauschen Sie Ihre Algorithmusbeschreibung mit einer anderen Gruppe und versuchen Sie, deren Algorithmus zu verstehen.

Aufgabe 2 Konstruktoren, Bedingungen und APIs

In Übungsblatt 5 haben Sie bereits die Erzeugung eines zweiten Hamsters Paula implementiert. In dieser Aufgabe geht es nun darum, den Umgang mit Konstruktoren und Bedingungen zu vertiefen. Öffnen Sie analog zu dem Vorgehen aus Übung 5 das git-Repo für Übungsblatt 6. URL: `https://github.com/RSS-PSE-WS1920/exercise-sheet-6`.

(a) Öffnen Sie die Klasse `WithOrWithoutPaulaHamsterGame`. Erweitern Sie den Konstruktor der Klasse so, dass über ein Argument gesteuert werden kann, ob Paula hinzugefügt wird oder nicht. Paula soll im positiven Falle an der Position (3,5) platziert werden und in westliche Richtung schauen. Zudem soll sie bereits 10 Körner im Mund haben. Im negativen Falle soll nichts geschehen und Paula nicht erzeugt werden.

(b) Erweitern Sie Ihr Programm aus der vorherigen Aufgabe so, dass sich Paule, falls Paula nicht erzeugt wird, einmal im Kreis dreht (360°) und sich umschaut, ob Paula nicht doch irgendwo im Territorium zu finden ist.

(c) **Herausforderung I:** Erweitern Sie ihr Programm so, dass der Nutzer im Falle der Erzeugung von Paula mittels eines separaten Fensters gefragt wird, wieviele Körner Paula im Mund haben soll.

Aufgabe 3 Schleifen und Bedingungen

In den letzten Übungsblättern hatten Sie oft die mühsame Aufgabe, Paule explizit mittels Operationsaufrufen durch das Territorium zu navigieren. In dieser Übungsaufgabe sollen Schleifen verwendet werden, um Ihren Programm-Code flexibler und generischer zu gestalten.

(a) Öffnen Sie die Klasse `LoopingPaule`. Die Operation `multiMove(int numberOfSteps)` soll es ermöglichen, dass Paule mehrere Schritte auf einmal in die aktuelle Richtung ausführt. Implementieren Sie diese Operation mittels einer For-Schleife, sodass der übergebene Integer-Parameter `numberOfSteps` bestimmt, wieviele Schritte Paule ausführt.

(b) Die Operation `spreadGrainsInCurrentDirection` soll Paule anweisen, alle Körner, die er aktuell im Mund hat, auf die vor ihm liegenden Felder zu verteilen (ausgehend von seiner aktuellen Richtung). Auf jedes Feld soll genau ein Korn gelegt werden. Implementieren Sie diese Operation mittels einer geeigneten Schleife. Diskutieren Sie vorweg mit Ihrem Partner, welcher Schleifentyp dafür geeignet ist **und** geben Sie die Schleifeninvariante sowie die Schleifenvariante an (z.B. natürlichsprachlich).

(c) Die Operation `moveToNextWall` soll es ermöglichen, dass Paule solange in die aktuelle Richtung läuft, bis er auf eine Wand stößt oder das Territorium endet. Implementieren Sie diese Operation mittels einer geeigneten Schleife. Diskutieren Sie vorweg mit Ihrem Partner, welcher Schleifentyp dafür geeignet ist.

(d) Erstellen Sie für alle oben genannten Operationen entsprechende Javadoc-Kommentare inklusive Vor- und Nachbedingungen in natürlicher Sprache.

(e) **Herausforderung II:** Geben Sie die Schleifeninvariante sowie die Schleifenvariante an (z.B. natürlichsprachlich) für Aufgabe (c).

(f) **Herausforderung III:** Erweitern Sie die Teilaufgabe (c) so, dass der Nutzer im Falle einer im Weg stehenden Wand gefragt wird, in welche Richtung Paule weiterlaufen soll. Anschließend soll Paule in die vom Nutzer eingegebene Richtung weiterlaufen, bis er wieder gegen eine Wand stößt. Dann soll das Prozedere von vorne beginnen.

(g) **Herausforderung IV:** Erweitern Sie die Teilaufgabe (d) so, dass sich Paule merken kann, auf welchem Feld er nach jedem Schritt gestanden ist. Schreiben Sie außerdem eine Abfrage, welche die Schrittnummer als Parameter akzeptiert und dafür die damalige Position zurückgibt.

2 Teil B - Heimarbeit

Aufgabe 1 Objekte und Referenzen

In der folgenden Aufgabe geht es um Ihr Wissen im Bezug auf Objekte und Referenzen.

(a) *(2 Punkte)* Erklären Sie den Unterschied bzw. die Beziehung zwischen Klassen und Objekten.

(b) *(2 Punkte)* Erklären Sie, was eine Referenz in Java ist.

(c) *(2 Punkte)* Erklären Sie die Bedeutung und Nutzen der Keywords: `this` und `null` in Java.

(d) *(2 Punkte)* Erklären Sie, was `NullPointerExceptions` sind und klären Sie die Frage, ob diese durch statische Codeanalysen identifiziert werden können. Begründen Sie Ihre Antwort kurz.

(e) *(2 Punkte)* Gegeben ist folgender Code:

```
Point maxPoint = new Point(10, 10);
Point minPoint = new Point(2, 2);
minPoint = maxPoint;
maxPoint.setLocation(13, 13);
```

Geben Sie den Wert von `minPoint.getX()` an und begründen Sie Ihre Antwort. Die Dokumentation der Klasse Point finden Sie in der Dokumentation der Java Base Class Library.

(f) *(2 Punkte) Bonus*: Gegeben ist folgender Code:

```
String blue = "blue";
String red = "red";
red = blue;
blue = "light blue";
```

Überlegen Sie sich den Wert von `red`. Recherchieren Sie, ob sich der Code genauso verhält, wie in (e) und wenn nein, begründen Sie kurz.

Aufgabe 2 Objekterzeugung und Konstruktoren

In dieser Aufgabe geht es darum, dass Sie einen eigenen Konstruktor schreiben, in dem Sie mehrere Hamster anlegen.

Beachten Sie bei allen Aufgaben (und auch zukünftigen Aufgaben) die Richtlinien des Java-Style-Guides und überlegen Sie sich Vor- und Nachbedingungen.

Hinweis: Bei Nicht-Einhaltung des Java-Style-Guides können bis zu 50% der Punkte abgezogen werden.

(a) *(2 Punkte)* Betrachten Sie die Klasse `CustomHamsterGame`. Erweitern Sie die Klasse zunächst um drei weitere Objektvariablen: `lessy`, `tiffany` und `ronny` vom Typ `Hamster`.

(b) *(2 Punkte)* Schreiben Sie nun einen Konstruktor für die Klasse. In dem Konstruktor soll zunächst der Konstruktor der Oberklasse aufgerufen werden. Nutzen Sie dazu folgenden Ausdruck: `super(TERRITORY_EMPTY)`.

Im Anschluss soll die Operation `initLessyAndTiffany()` aufgerufen werden. Fügen Sie hierzu die passende Operation, noch ohne Funktionalität, hinzu.

(c) *(4 Punkte)* Implementieren Sie nun die Funktionalität der Operation `initLessyAndTiffany()`, welche die zwei Hamster Lessy und Tiffany initialisiert, sodass am Ende in drei von vier Ecken des Territoriums ein Hamster steht und ins Spielfeld schaut (also entweder nach Westen oder Osten).

Um die volle Punktzahl zu erhalten, passen Sie Ihren Code so an, dass die Platzierung der Hamster dynamisch erfolgt. D.h. für alle leeren Territorien ohne Wände funktioniert.

(d) *(4 Punkte)* Füllen Sie als nächstes die Operation `run`: Als erstes soll der letzte verbleibende Hamster Ronny initialisiert werden. Die Position soll vom Nutzer abgefragt werden. Suchen Sie in der API nach einem passenden Befehl. Blickrichtung und Körner können Sie selbst bestimmen.

(e) *(2 Punkte)* Im Anschluss sollen Sie Paule und Tiffany auf ein gemeinsames Feld laufen lassen. Das gemeinsame Feld darf weder das Startfeld von Paule, noch das Startfeld von Tiffany sein.

(f) *(6 Punkte)* Passen Sie Ihre Operation `run` so an, dass, wenn Ronny auf ein Feld mit Lessy oder Tiffany gesetzt wird, ein weiterer Hamster daneben erzeugt wird.

Wenn Ronny auf das Feld mit Paule gesetzt wird, soll Paule schreiben: "Hier ist kein Platz für uns beide" und wenn Ronny auf ein leeres Feld gesetzt wird, soll er selbst sagen: "Wuhu, hier ist Platz".

Hinweis: Denken Sie daran, Ihren Programmcode zu dokumentieren und definieren Sie Vor- und Nachbedingungen für den Konstruktor sowie die Operationen. Geben Sie auch die Klasseninvarianten an.

3 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 Informatikunterricht an sich

Dieses Mal geht es weniger um Paule sondern mehr um den gesamten Informatikunterricht an sich. Informatik – egal ob in der Mittel- oder Oberstufe - unterscheidet sich in einigen Punkten von anderen Fächern wie Mathematik oder Deutsch. Diese Aufgabe soll Sie über genau diese Unterschiede nachdenken lassen.

(a) Diskutieren Sie im Team, welche Unterschiede es zwischen dem Informatikunterricht und anderen Fächern (Mathematik, Deutsch, Englisch, Geschichte, ...) gibt.

(b) Welche dieser Unterschiede können als Vorteile angesehen werden? Diskutieren und begründen Sie.

(c) Welche dieser Unterschiede können als Nachteile angesehen werden? Diskutieren und begründen Sie.

(d) Was bedeuten diese Unterschiede im Hinblick auf die Vor- und Nachbereitung des Informatikunterrichts?

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!