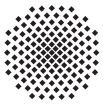

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 7

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 06.12.2019, 23:59 Uhr — Besprechung: KW 51

- 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.
- 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:

1. **PSE[ÜB-Nr]-[Nachnamen der Teammitglieder]-[Nachname des Tutors].pdf**
2. **MountPauleHamsterGame.java**
3. **AmazingPauleHamsterGame.java**

Lernziel: Dieses Übungsblatt dient dazu, dass Sie die in der Vorlesung vorgestellten Konzepte zum Programmablauf und zu 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 zum Kontrollfluss vertraut gemacht haben und die Übungsblätter 5 und 6 absolviert 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 53 Punkte erzielen.

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

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

Die Dokumentation der Hamster-API finden Sie wie gewohnt hier: `http://caloundra.informatik.uni-stuttgart.de`

Viel Erfolg!

1 Teil A - Präsenzaufgaben

Aufgabe 1 Schleifen, Schleifeninvariante und Schleifenvariante

Bei dieser Frage geht es darum, die vorgestellten Konzepte von Schleifeninvarianten und Schleifenvarianten zu verstehen. Beantworten Sie bitte die folgenden Fragen für den angegebenen Code-Ausschnitt:

```
protected void run() {
    game.displayInNewGameWindow();
    game.initialize();

    for (int i = 0; i <= 3; i++) {
        paule.pickGrain();
        paule.move();
    }
}
```

(a) Was wird Paule durch den angegebenen Code angewiesen zu tun?

(b) Für die Operation fehlen die Vor- und Nachbedingungen. Nachdem Sie verstanden haben, was der Code tut, ergänzen Sie bitte die fehlende Dokumentation. Vor- und Nachbedingungen können in JML oder natürlicher Sprache definiert werden.

(c) Definieren Sie für die angegebene Schleife Invariante sowie Variante. Bitte definieren Sie diese oberhalb der Schleife, wie in der Vorlesung vorgestellt.

(d) Zeigen Sie, dass die definierte Schleifeninvariante in jedem Bereich der gegebenen Schleife gilt: Vor dem Betreten der Schleife, vor und nach dem Ausführen des Rumpfes und nach dem Verlassen der Schleife.

(e) Beweisen Sie die Schleifenvariante in der gleichen Art und Weise wie es in der Vorlesung gezeigt wurde.

(f) In der Vorlesung wurde vorgestellt, dass jede for-Schleife in eine while-Schleife oder do-while-Schleife konvertiert werden kann. Bitte wandeln Sie die angegebene for-Schleife in eine while-Schleife und eine do-while-Schleife um.

(g) Wirkt sich dies auf die Schleifeninvariante und die Schleifenvariante aus?

Aufgabe 2 Ausnahmebehandlung

Oftmals sind die Ressourcen, die das Programm benötigt, nicht vorhanden (z.B. eine Datei). In solchen Fällen möchten wir, dass sich unser Programm korrekt verhält. In dieser Übung werden Sie die `run`-Operation von `ExceptionalHamsterGame` ändern, um solche Fälle zu behandeln.

Importieren Sie dazu zunächst das Projekt `exercise-sheet-7` aus unserem git-Repository `https://github.com/RSS-PSE-WS1920`.

(a) Betrachten Sie zunächst die Klasse `ExceptionalHamsterGame`. Öffnen Sie im Anschluss die Klasse `Main`. Die Klasse hat eine Operation: `main`. Füllen Sie die Operation `main`, sodass eine neue Instanz der Klasse `ExceptionalHamsterGame` instanziiert wird. Im Anschluss können Sie auf der Instanz die Operation `doRun()` aufrufen. Schauen Sie, was passiert.

(b) Ändern Sie die `run`-Operation in der Klasse `ExceptionalHamsterGame`, indem Sie einen fehlerhaften Pfad angeben (z.B. my-territory.ter hinzufügen). Was beobachten Sie?

(c) Lesen Sie die Dokumentation des Befehls, der für die Initialisierung des Territoriums verantwortlich ist. Was bemerken Sie und wie ist die Ausnahme aktuell behandelt?

(d) Ändern Sie die `run`-Operation des `ExceptionalHamsterGame` so ab, dass sie die Ausnahme besser behandelt. Eine Möglichkeit wäre, das Standardgebiet zu laden, wenn die Gebietsdatei fehlt. Suchen Sie in der API die passende Operation und ändern Sie die Ausnahmebehandlung entsprechend. Im Fall der Ausnahmebehandlung soll Paule den Nutzer über das Laden des Standardgebiets informieren. Implementieren und testen Sie dieses Verhalten im `ExceptionalHamsterGame`.

Aufgabe 3 Hungry Paule

In der folgenden Übung sind Sie dafür verantwortlich, die Klasse `HungryPauleGame` umzustrukturieren.

Ihre Aufgabe ist es dabei, Operationen zu identifizieren, welche extrahiert werden können. Als Faustregel ist (allgemein!) davon auszugehen, dass keine Operation länger als 10 Zeilen sein sollte.

Hinweis: In dieser Aufgabe wird Hamster Paule nicht aufhören, sich zu bewegen, bis er alle Körner auf dem Feld gegessen hat.

(a) Öffnen Sie die Klasse `HungryPauleGame`. Lesen Sie den Quellcode und führen Sie den Code aus, um zu verstehen, was er tut. *Hinweis:* Um den Quellcode auszuführen, müssen Sie anlog zu Aufgabe 1 die Klasse `Main` bearbeiten und eine neue Instanz der Klasse `HungryPauleGame` instanziieren. Im Anschluss müssen Sie erneut die Operation `doRun()` aufrufen.

(b) Welche Operationen haben Sie bereits implementiert, die Ihnen helfen könnten, dasselbe auf eine viel "sauberere" (lesbarere) Weise zu erreichen? Zählen Sie diese auf.

(c) Extrahieren Sie sich wiederholenden Code in Operationen, die wiederverwendet werden können. Verbessern Sie den Code, indem Sie die in den vorherigen Aufgaben erlernten Konzepte (z.B. Schleifen) verwenden. Dokumentieren Sie die neuen Operationen.

2 Teil B - Heimarbeit

Aufgabe 1 MountPaule - Schleifen und Algorithmen

In dieser Aufgabe geht es darum, sich zunächst konzeptionell einen Algorithmus zu überlegen, der anschließend technisch implementiert werden soll. Bitte geben Sie dazu für Aufgabenteil a) eine PDF ab, für die Aufgabenteile b) - d) die vervollständigte MountPauleHamsterGame.java-Datei.

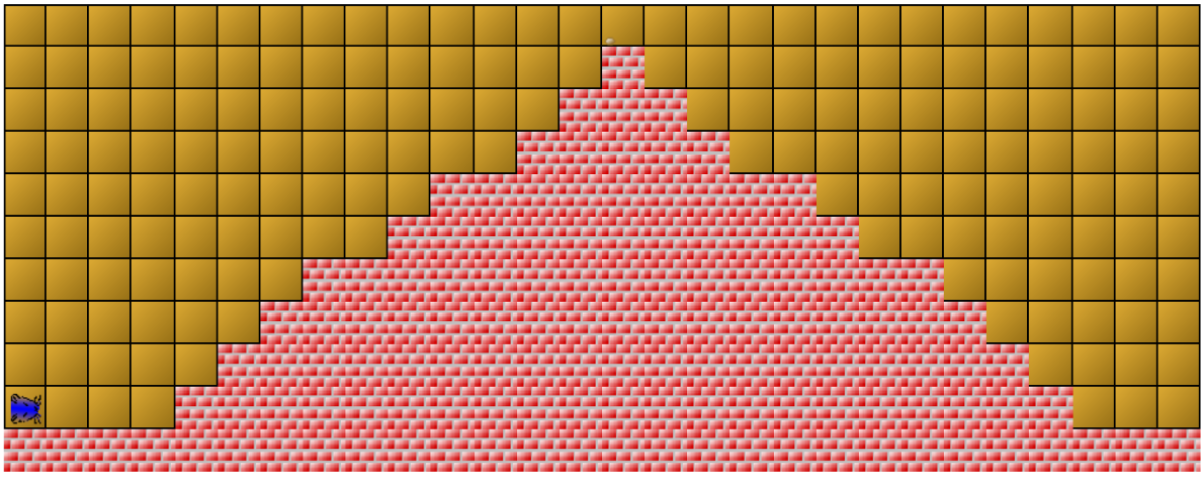

Abbildung 1: MountPaule - Ein Berg mit diskreten Stufen, den es zu erklimmen gilt

Folgendes Szenario ist gegeben: Abbildung 1 zeigt Ihnen ein Territorium aus der Seitenperspektive, d.h. Sie sehen Paule auf dem Boden stehend, der einen großen Berg vor sich hat. Diesen Berg gilt es zu erklimmen, da es auf der Spitze des Berges ein Korn zu ergattern gibt und Paule gegenwärtig keinerlei Körner im Mund hat.

(a) *(4 Punkte)* Entwicklen Sie konzeptionell einen Algorithmus in natürlicher Sprache, welcher das Erklimmen eines **beliebigen** Berges bis hin zum Einsammeln des Korns beschreibt. Der Algorithmus soll dabei nicht nur spezifisch für einen bestimmten Berg funktionieren, sondern auch für anders geartetete Berge, beispielsweise kleinere Berge, größere Berge, nicht so steile Berge, etc. Paule kann jedoch in jedem Schritt maximal die Höhe einer Stufe überwinden, d.h. zu steile Berge müssen nicht berücksichtigt werden. Gehen Sie davon aus, dass sich auf der Spitze des Berges immer ein Korn befindet, das es einzusammeln gilt.

(b) *(10 Punkte)* Implementieren Sie Ihren Algorithmus in der Operation `climbTheMountain` in der Klasse `MountPauleHamsterGame`. Verwenden und implementieren Sie dazu unter anderem die Operation `climbUb`, welche Paule eine Stufe des Berges erklimmen lassen soll.

Hinweis I: Vergrößern Sie das startende HamsterGame-Fenster, welches den Berg zeigt, soweit, dass das gesamte Territorium zu sehen ist. Andernfalls kann es sein, dass Fehler geworfen werden.

Hinweis II: Sie können Ihr Programm testen, indem sie die Operation `main` aufrufen. Der einfachste Weg hierfür ist, einen Rechtklick auf die Klasse im Package Explorer zu machen, „Run As“ und anschließend „1 Java Application“ auszuwählen.

(c) *(2 Punkte)* Testen Sie, ob Ihr Algorithmus Paule auch auf andere Berge klettern lassen kann. Dazu müssen Sie im Konstruktor der Klasse `MountPauleHamsterGame` ein anderes Territorium angeben, welches einen anderen Berg beinhaltet. Der Name des neuen Territoriums ist `territories/territory-mountpaule-2-ub7.ter`. Verwenden Sie dieses Territorium in Ihrer abzugebenden MountPauleHamsterGame-Datei, sodass ersichtlich ist, dass Sie diese Aufgabe umgesetzt haben.

(d) *(12 Punkte)* Erstellen Sie JavaDoc-Kommentare für die Klasse `MountPauleHamsterGame` sowie deren Operationen. Dokumentieren Sie zudem für jede Operation (`climbTheMountain`, `climbUb`) mindestens eine Vor- und eine Nachbedingung in JML **und** eine Vor- sowie eine Nachbedingung in natürlicher Sprache.

Aufgabe 2 A'Mazing Paule - Schleifen und Algorithmen

Wie in der vorherigen Aufgabe geht es darum, sich zunächst konzeptionell einen Algorithmus zu überlegen, der anschließend implementiert werden soll. Bitte geben Sie dazu für Aufgabenteil a) eine PDF ab, für die Aufgabenteile b) - d) die vervollständigte AmazingPauleHamsterGame.java-Datei.

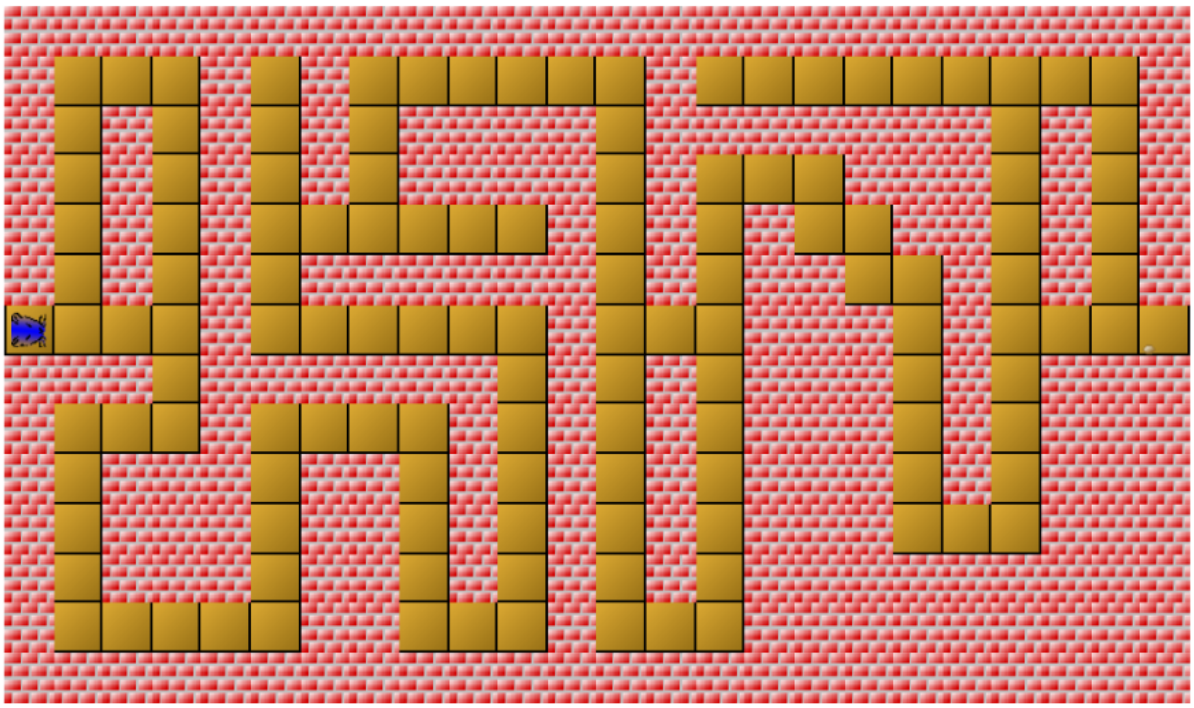

Abbildung 2: A'MazingPaule - Ein Labyrinth gilt es zu passieren

Folgendes Szenario ist gegeben: Abbildung 2 zeigt Ihnen ein Territorium wie gewohnt aus der Vogelperspektive, d.h. Sie sehen Paule von oben, welcher vor einem Labyrinth positioniert ist. Paule ist wiederum sehr hungrig und hat keine Körner im Mund. Am Ende des Labyrinths ist jedoch ein Korn zu finden, das es einzusammeln gilt.

(a) *(6 Punkte)* Entwickeln Sie konzeptionell einen Algorithmus in natürlicher Sprache, welcher das Passieren des Labyrinths beschreibt. Der Algorithmus soll dabei nicht nur spezifisch für ein bestimmtes Labyrinth funktionieren, sondern auch für anders geartetete Labyrinthe. Gehen Sie davon aus, dass Paule immer am Eingang des Labyrinths positioniert ist und sich am Ausgang des Labyrinths ein Korn befindet, das es einzusammeln gilt.

Hinweis I: Recherchieren Sie die „Linke-Hand-Regel" für Labyrinthe, falls Sie eine kleine Hilfestellung benötigen.

Hinweis II: Sie können Ihr Programm testen, indem sie die Operation `main` aufrufen. Der einfachste Weg hierfür ist, einen Rechtklick auf die Klasse im Package Explorer zu machen, „Run As" und anschließend „1 Java Application" auszuwählen.

(b) *(10 Punkte)* Implementieren Sie Ihren Algorithmus in der Operation `passTheMaze` in der Klasse `AmazingPauleHamsterGame`.

Hinweis: Vergrößern Sie das startende HamsterGame-Fenster, welches das Labyrinth zeigt, soweit, dass das gesamte Territorium zu sehen ist. Andernfalls kann es sein, dass Fehler geworfen werden.

(c) *(2 Punkte)* Testen Sie, ob Ihr Algorithmus Paule auch durch andere Labyrinthe führen kann. Dazu müssen Sie im Konstruktor der Klasse `AmazingPauleHamsterGame` ein anderes Territorium angeben, welches ein anderes Labyrinth beinhaltet. Der Name des neuen Territoriums ist `territories/territory-amazingpaule-2-ub7.ter`. Verwenden Sie dieses Territorium in Ihrer abzugebenden AmazingPauleHamsterGame-Datei, sodass ersichtlich ist, dass Sie diese Aufgabe umgesetzt haben.

(d) *(7 Punkte)* Erstellen Sie JavaDoc-Kommentare für die Klasse `AmazingPauleHamsterGame` sowie deren Operationen. Dokumentieren Sie zudem mindestens eine Vor- und eine Nachbedingung in JML und eine Vor- und eine Nachbedingung in natürlicher Sprache für die Operation `passTheMaze`.

Aufgabe 3 Anmeldung zur Scheinklausur

Die Scheinklausur wird am 13.01.2020 um 15:45 Uhr stattfinden. Um an der Klausur teilzunehmen ist eine Anmeldung bis zum 06.12.2019 notwending.

Die Anmeldung erfolgt über die Einschreibung in die folgende ILIAS Gruppe: `https://ilias3.uni-stuttgart.de/goto_Uni_Stuttgart_grp_1816937.html`

Zusätzlich kann die Gruppe im ILIAS Kurs der PSE unter '00 Organisatorischens und Zusatzmaterialien' gefunden werden.

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 Aus vorher folgt nachher

In den letzten Wochen haben Sie Verträge und damit auch Vor- und Nachbedingungen kennen gelernt. Diese sind ein wichtiger Bestandteil des Programmierens vor allem für große Projekte. Sie sollen sich in dieser Aufgabe Gedanken darüber machen in wie weit Vor- und Nachbedingungen in der Schule sinnvoll sind.

(a) Diskutieren Sie mit Ihrem Team, in wie weit man Vor- und Nachbedingungen in der Schule einführen könnte.

(b) Sammeln Sie gemeinsam mit Ihrem Team, welche Schwierigkeiten Schülerinnen und Schüler bei Vor- und Nachbedingungen haben könnten.

Hinweis: Denken Sie an Schwierigkeiten, die Sie vielleicht selbst damit hatten.

(c) Überlegen Sie gemeinsam mit Ihrem Team, welche Lerninhalte bei Vor- und Nachbedingungen in der Schule wichtig sein könnten und welche man ggf. weglassen könnte, ohne den zu vermittelnden Kern zu verändern.

Hinweis: Dies ist etwas, was Sie bei Themen später in der Schule immer wieder tun müssen. Es nennt sich "didaktische Reduktion" und kann den Stoff sowohl in der Breite ("horizontal") als auch in der Tiefe ("vertikal") reduzieren.

Hinweise zu den Übungen:

- Die Abgabe erfolg 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!