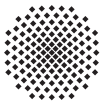


Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 8

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 13.12.2019, 23:59 Uhr — Besprechung: KW 2

- 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. Nutzen Sie bei Java-Dateien die korrekte JavaDoc-Syntax.
- Benennen Sie die Dateien nach dem folgenden Schema:
  1. **CollectorPauleGame.java**
  2. **PainterPauleGame.java**
  3. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-[**Nachname des Tutors**].**pdf**

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 zum Kontrollfluss vertraut gemacht haben und die Übungsblätter 6 und 7 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 40 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 relevanten Unterlagen, für die Bearbeitung des Übungsblatts 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` (nur aus dem Uni-Netz erreichbar).

Viel Erfolg!

1 Teil A - Präsenzaufgaben

Aufgabe 1 Mutable und Immutable

Ziel dieser Aufgabe ist es, Ihnen die Konzepte von unveränderlichen Klassen näher zu bringen. Dazu sollen Sie eine unveränderliche Klasse für Studierende implementieren. Attribute der Objekte dieser Klasse sollen nur im Konstruktor setzbar sein.

Importieren Sie dazu zunächst das Projekt `exercise-sheet-8` aus unserem git-Repository `https://github.com/RSS-PSE-WS1920`.

(a) Bearbeiten Sie im Folgenden die Klasse `MyImmutableStudent`. Fügen Sie zwei Objektvariablen hinzu: eine Matrikelnummer und einen Namen (Name des Studenten). Erstellen Sie im Anschluss alle benötigten Getter- und Setter-Operationen. Implementieren Sie die Klasse so, dass Ihre Objekte immutable sind.

Hinweis: Denken Sie gründlich über die Zugriffsrechte und weitere Modifikationen für die Deklarationen innerhalb Ihrer Klasse nach.

(b) Erstellen Sie eine weitere Klasse `Main`, die eine Klassenoperation `main` hat (vgl. Foliensatz 11, Folie 24). Erstellen Sie in dieser Operation ein Objekt der Klasse `MyImmutableStudent`. Rufen Sie dann die Operationen `getMatrikelnummer` und `getName` auf dem Objekt auf. Kompilieren Sie Ihre Klassen, stellen Sie sicher, dass es keine Fehler gibt und führen Sie die Main-Operation aus. Versuchen Sie, die Werte im Anschluss neu zu setzen. Ist dies möglich und wenn nein warum nicht?

Aufgabe 2 Mutable-Objekt innerhalb von Immutable-Objekt

In dieser Aufgabe vertiefen wir die Konzepte von Mutable und Immutable. Sie haben bereits eine Klasse `MyImmutableStudent` implementiert.

In dieser Übung erweitern Sie Ihre Klasse `MyImmutableStudent` um eine weitere Objektvariable namens `address`. Die Adresse soll vom Typ `Address` sein. Die Klasse wurde bereits von einem Ihrer Kommilitonen implementiert. Sie finden die Implementierung der Klasse `Address` in unserem Repository.

(a) Erläutern und begründen Sie, warum die Klasse nicht unveränderlich ist und geben Sie ein Beispiel an.

(b) Fügen Sie nun ein neues Attribut zu Ihrer Klasse `MyImmutableStudent` vom Typ `Address` hinzu. Das Attribut soll den Geburtsort des/der Studierenden repräsentieren.

(c) Diskutieren Sie mit Ihrem Gruppenpartner, ob Ihre Klasse noch unveränderlich ist oder nicht – jeweils mit Begründung natürlich.

(d) Falls Ihre Klasse nicht mehr unveränderlich ist, passen Sie Ihren Code so an, dass die Klasse wieder unveränderlich wird.

(e) Tauschen Sie Ihre Implementierung mit einer anderen Gruppe und diskutieren Sie mit der Gruppe, ob Ihre Klasse unveränderlich ist oder nicht.

Aufgabe 3 Vererbung

Gegeben ist folgendes Szenario:

Sie arbeiten für das Kraftfahrt-Bundesamt (KBA) und sind damit für die gesamte Erfassung von motorisierten Flug-/Fahrzeugen verantwortlich. Um einen besseren Überblick zu bekommen und die Prozesse zu optimieren, werden Sie von Ihrem Chef gebeten, eine Verwaltungssoftware für Kraftfahrzeuge zu entwerfen.

Da Sie in Ihrem Studium von objektorientierter Programmierung gehört haben, wollen Sie als erstes Klassen und deren Beziehungen identifizieren.

Da Sie hoch motiviert sind, machen Sie sich gleich an die Aufgabe und befragen Ihre Kollegen in den unterschiedlichen Abteilungen. Dabei ist jede Abteilung für einen anderen Fahrzeugtyp verantwortlich. Im KBA gibt es drei Abteilungen: Luftfahrt, Schifffahrt, sowie Ihre eigene Abteilung Landfahrzeuge.

Als erstes interviewen Sie Ihre Kollegen aus der Luftfahrt. Diese erzählen Ihnen, dass sie alle Arten von Fluggeräten betrachten. Generell unterscheiden Sie in flügellose (Fesselballons und Heißluftballon) sowie Flügler (Flugzeuge und Raketen). Fesselballons haben keinen zusätzlichen Brenner,

dafür sind sie mit einem bestimmten Gas gefüllt, während Heißluftballon einen Brenner haben. Bei Flugzeugen wird zusätzlich zwischen militärischen und zivilen Flugzeugen unterschieden. Während militärische Flugzeuge eine militärische Zertifizierung nach MIL-STD haben, werden zivile Flugzeuge nach dem ICAO-Standard zertifiziert und haben auch immer eine bestimmte Anzahl an Notausgängen.

(a) Identifizieren Sie für den gegeben Sachverhalt eine Vererbungshierarchie. Geben Sie für jede identifizierte Klasse zwei weitere Attribute an.

(b) Erweitern Sie Ihre Vererbungshierarchie für den Zweig Landfahrzeuge. Überlegen Sie sich mindestens 4 weitere Klassen in mindestens 2 Vererbungshierachien. Geben Sie für jede identifizierte Klasse zwei weitere Attribute und ein Alleinstellungsmerkmal an.

(c) Diskutieren Sie Ihre Lösung mit einer anderen Gruppe.

(d) Implementieren Sie mind. 5 Klassen Ihrer Lösung für eine Vererbungshierarchie (z. B. Landfahrzeuge oder Luftfahrt) in Eclipse in einem neuen Projekt. Implementieren Sie auch Attribute mit entsprechenden Sichtbarkeiten sowie Getter- und Setter-Operationen, wo es Sinn ergibt.

(e) Denken Sie auch an Vor- und Nachbedingungen sowie JavaDoc.

2 Teil B - Heimarbeit

Aufgabe 1 Schleifen

Wir hätten gerne eine Operation, die es dem Hamster Paule ermöglicht, mehr als ein Korn von dem aktuellen Feld zu sammeln.

Dazu ist die Klasse `CollectorPauleGame` gegeben, welche eine Implementierung der Operation `multiMove(int numberOfSteps)` aus der letzten Übung enthält.

Zudem ist ein Territorium gegeben, das zwei Felder mit Körnern besitzt. Auf dem einen Feld sind es 10 Körner. Auf dem anderen sind es mehr als 10 Körner.

(a) *(4 Punkte)* Implementieren Sie die Operation `collectTen()` mit einer For-Schleife, um genau 10 Körner zu sammeln.

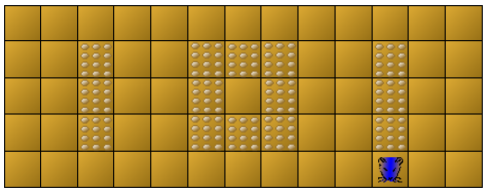
(b) *(3 Punkte)* Testen Sie das implementierte Verhalten der Opertation `collectTen()`. Vervollständigen Sie hierzu die Operation `run()`. Wie ist die Reihenfolge der Befehle für den Test, um einen erfolgreichen Durchlauf zu erhalten, einen fehlerhaften Durchlauf zu erhalten und einen Durchlauf, der zwar erfolgreich ist, aber **nicht alle** Körner auf dem Feld einsammelt? Erstellen Sie die einzelnen Tests jeweils in Hilfsoperationen (zB.: testAllSuccess, testFail und testNotAllGrainsSuccess).

(c) *(4 Punkte)* Implementieren Sie eine weitere Operation `collectAll` so, dass von nun an eine beliebige Anzahl von Körnern aufgesammelt werden kann. Nutzen Sie dazu eine While-Schleife und testen Sie Ihre neue Implementierung.

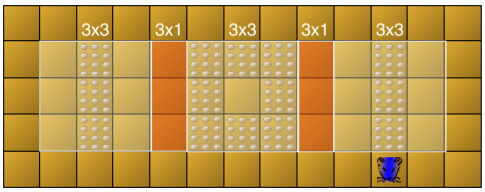
Aufgabe 2 Malen von Binärzahlen

In dieser Aufgabe programmieren Sie `PainterPauleGame`. Laden Sie die im Repository angegebene `PainterPauleGame`-Klasse herunter. PainterPauleGame sollte in der Lage sein, Zahlen von 0 bis 7 in binärer Form zu malen (z. B. 111 für 7 und 000 für 0). Nach erfolgreicher Implementierung der fehlenden Operationen von `PainterPauleGame`, soll der Programmablauf wie folgt sein. (A) Der Benutzer wird nach einer Nummer gefragt. (B) Die Nutzereingabe wird verifiziert. (C1) Ist die Zahl korrekt, beginnt Paule mit dem Malen der Zahl. (C2) Falls nicht, wird der Nutzer erneut nach einer Zahl gefragt.

Hinweis: Der Einfachheit halber können Sie davon ausgehen, dass Paule immer auf der Position (0,0) steht und in Richtung Osten schaut mit 324 Körnern im Mund. Zusätzlich ist es Möglich die Spielgeschwindigkeit programatisch nach oben zu setzen. Die entsprechende Operation finden Sie in der API Dokumentation.



(a) Auf diese Weise malt Paule die Zahl 5.



(b) Gebietsaufteilung für die Ziffern

Bitte implementieren Sie die folgenden Operationen:

(a) *(2 Punkte)* `fillTile()` — Die Operation `fillTile()` soll Paule 12 Körner in das aktuelle Feld legen lassen.

(b) *(4 Punkte)* `paintOne()` — Die `paintOne()`-Operation lässt Paule die Ziffer "1" auf das Feld *malen* (durch das Ablegen von Körnern). Dabei soll jedes Feld mit Körnern gefüllt werden. Ein Feld gilt als gefüllt, wenn sich 12 Körner auf ihm befinden. Um eine "1" zu malen müssen drei Felder vertikal untereinander gefüllt sein.

Denken Sie an Vor- und Nachbedingungen.

(c) *(4 Punkte)* `paintZero()` — Die Operation `paintZero()` lässt Paule die Ziffer "0" malen. Die Ziffer "0" wird, wie in der Abbildung 2 und 3 gezeigt, dargestellt.

Denken Sie an Vor- und Nachbedingungen.

(d) *(2 Punkte)* `makeSpace()` — Die Operation `makeSpace()` soll ein einspaltiges "Leerzeichen" erzeugen und Paule an die Position bringen, um die nächste Ziffer zu malen.

Denken Sie an Vor- und Nachbedingungen.

(e) *(2 Punkte)* `testPaintingOne()` — Testen Sie die Korrektheit der oben implementierten Operationen. Die Operation `testPaintingOne()` soll die oben genannten Operationen nutzen, um die Dezimalzahl Eins (in binärer Form "001") zu malen.

(f) *(4 Punkte)* `integerToBinary(int number)` — Die Operation `integerToBinary` sollte eine gegebene ganze Zahl in ihre binäre Zeichenkettendarstellung transformieren. Die Operation soll einen String zurückgeben (z.B. "001").

Hinweis: Bitte recherchieren Sie, welche Java-Bibliothek Sie für die jeweilige Aufgabe wiederverwenden können.

(g) *(5 Punkte)* `paintDigit()` — Die Operation `paintDigit()` fordert den Benutzer auf, eine Eingabe zwischen 0 und 7 vorzunehmen. Ist die Zahl korrekt, beginnt Paule mit dem Malen der Zahl. Falls nicht wird der Nutzer erneut nach einer Zahl gefragt. Nutzen sie nun die Operation `integerToBinary`, um die Zahl in ein Zeichenkettenobjekt (String) umzuwandeln, das ihre binäre Representation (z. B. in Form von "111" für 7) enthält. Ihre Aufgabe ist es, den erhaltenen String zu verarbeiten und Paule anzuweisen, Einsen und Nullen basierend auf dem Wert malen zu lassen.

Hinweis: Verwenden Sie die Operation `charAt(int i)`, die für ein Zeichenkettenobjekt verfügbar ist, um das Zeichen (character) an Position i zu erhalten.

(h) *(2 Punkte)* Welche Dezimalzahl ist für Paule in Bezug auf die verwendeten Körner am teuersten?

(i) *(4 Punkte) Bonus*: Was ist der Algorithmus, um eine Dezimalzahl in eine Binärzahl umzuwandeln? Implementieren Sie den Algorithmus in Java, der eine ganze Zahl annimmt und sie in eine Binärzahl umwandelt.

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 Gamification

In der Informatik gibt es die Möglichkeit, Inhalte spielerisch aufzubereiten. Dies nennt man "Gamification" (dt. "Spielifikation"). Im Laufe dieser Aufgabe sollen Sie sich Gedanken über dieses Konzept machen, dass durchaus auch im Unterricht genutzt werden kann.

(a) Recherchieren Sie im Internet, was man unter dem Begriff "Gamification" versteht.

(b) Überlegen Sie gemeinsam im Team, ob Ihnen Software (bspw. Smartphone-Apps) einfallen, die genau dieses Prinzip umsetzen und wie sie es umsetzen

(c) Überlegen Sie sich mit Ihrem Team Ansätze, wie man die Themen, die Sie bisher gelernt haben, mit der Gamification-Idee umsetzen könnte.

(d) Diskutieren Sie mit Ihrem Team, wie man den Hamster "gamifizieren" könnte.

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!