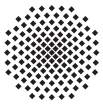

Universität Stuttgart

RSS
Reliable Software Systems

Übungsblatt 11

Programmierung und Softwareentwicklung (WiSe 2019/2020)

Abgabe: Fr. 17.01.2020, 23:59 Uhr — Besprechung: KW 5

- 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.
- **Neu:** Geben sie .java-Dateien nur im UTF-8 Encoding ab. Ändern Sie das Textdateiencoding von Eclipse auf UTF-8 ab, bevor Sie die Unterlagen herunterladen. Abhängig von Ihrem Betriebssytem müssen Sie möglicherweise auch nichts tun. (Tutorial: `https://youtu.be/07Rj8jw8cE8`)
- 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. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-**Aufgabe**[**Aufgabennummer**]**.zip**. Zum Erstellen dieser Datei exportieren Sie bitte Ihr Eclipse-Projekt über den Dialog: Datei, Exportieren, General, Archivdatei. Die Datei sollte mindestens folgende Dateien enthalten:
     - ***.java**: Alle erstellten Java-Dateien für Aufgabe 1 und Aufgabe 2
  2. **PSE**[**ÜB-Nr**]-[**Nachnamen der Teammitglieder**]-**Aufgabe2.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 zu den Themen Kontrollfluss, Vererbung und Clean Code vertraut gemacht haben und die Übungsblätter 9 und 10 absolviert haben.

Punkte: Dieses Übungsblatt enthält zwei Teile. Teil A mit 2 Aufgaben und Teil B mit 2 Aufgaben. Im Teil B können Sie bis zu 62 Punkte erzielen.

Style: Bitte halten Sie die in der Vorlesung vorgestellten Style-Regeln ein. Dazu gehören unter anderem JavaDoc, Vor- und Nachbedingungen, Prüfung derer mittels offensiver (mittels Asserts) und defensiver Checks (mittels Exceptions), Schleifeninvarianten und -varianten (mittels natürlicher Sprache oder JML im Kommentarblock über der Schleife), Sichtbarkeiten oder Read-Only Restriktionen. 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

Eine Dokumentation der Hamster-API finden Sie hier: `http://caloundra.informatik.uni-stuttgart.de` (nur aus dem Uni-Netz erreichbar).

Aufgabe 1 Berechnen von Fibonacci-Zahlen

In dieser Aufgabe sollen Sie einen Algorithmus schreiben, der die Fibonacci-Zahl n-ter Ordnung berechnet (`https://de.wikipedia.org/wiki/Fibonacci-Folge`).

(a) Erstellen Sie zunächst ein neues Projekt und eine neue Klasse mit einer main-Methode.

(b) Lesen Sie nach, was Fibonacci-Zahlen sind.

(c) Überlegen Sie sich einen rekursiven Algorithmus zur Berechnung der Fibonacci-Zahl n-ter Ordnung.

(d) Überlegen Sie sich Abbruchbedingungen.

(e) Implementieren Sie den Algorithmus. Denken Sie dabei an die Java-Dokumentation.

(f) Was passiert, wenn Sie die Abbruchbedingung aus Ihrem Code entfernen?

(g) Implementieren Sie eine weitere Operation, die eine Fibonacci-Zahl n-ter Ordnung mithilfe einer Schleife bestimmt.

(h) Messen Sie für beide Implementierungen die Zeit (Sie können die Timer-Klasse aus Übungsblatt 10 oder `System.nanoTime()` verwenden). Was stellen Sie fest? Diskutieren Sie das Ergebnis mit Ihrem Teampartner.

(i) Geben Sie die Invariante und Variante für beide Implementierungen an (iterativ und rekursiv).

Aufgabe 2 Überladen und Überschreiben

Diese Aufgabe befasst sich mit den Grundlagen der Vererbung, Polymorphie, Typen, Überladung und Überschreiben. Dazu ist folgender Code gegeben, der eine Klassenhierachie realisiert.

```
1  enum PowerSource {
2      HUMAN, FUEL, ELECTRICITY
3  }
4
5  interface Vehicle {
6      PowerSource getPowerSource();
7      void printVehicleInfo();
8  }
9
10 interface RegisteredVehicle extends Vehicle {
11     String getNumberPlate();
12 }
13
14 abstract class VehicleBase implements Vehicle {
15     private final PowerSource powerSource;
16
17     public VehicleBase(final PowerSource powerSource) {
18         this.powerSource = powerSource;
19     }
20
21     @Override
22     public void printVehicleInfo() {
23         System.out.format("Vehicle Type: %s \nPowered By: %s\n",
               getClass().getName(),this.powerSource);
24     }
25
26     @Override
27     public PowerSource getPowerSource() {
28         return this.powerSource;
29     }
```

```
30 }
31
32 abstract class RegisteredVehicleBase extends VehicleBase implements
       RegisteredVehicle {
33     protected final String numberPlate;
34
35     public RegisteredVehicleBase(final PowerSource powerSource, final
           String numberPlate) {
36         super(powerSource);
37         this.numberPlate = numberPlate;
38     }
39
40     @Override
41     public String getNumberPlate() {
42         return this.numberPlate;
43     }
44
45     @Override
46     public void printVehicleInfo() {
47         super.printVehicleInfo();
48         System.out.format("Number plate: %s\n",this.numberPlate);
49     }
50 }
51
52 class Car extends RegisteredVehicleBase implements RegisteredVehicle {
53     public Car(final String numberPlate) {
54         super(PowerSource.FUEL, numberPlate);
55     }
56 }
57
58 class Riksha extends RegisteredVehicleBase implements RegisteredVehicle
       {
59     public Riksha(final String numberPlate) {
60         super(PowerSource.HUMAN, numberPlate);
61     }
62 }
63
64 class EBike extends VehicleBase implements Vehicle {
65     public EBike() {
66         super(PowerSource.ELECTRICITY);
67     }
68 }
```

(a) Bestimmen Sie für die Programmzeilen 8-15 im folgenden Code, ob sie fehlerfrei übersetzt werden können oder nicht. Geben Sie eine Begründung für alle Fehler an.

```
1  Car dudu = new Car("Du-Du 1234");
2  Riksha hunk = new Riksha("HO-NK 123");
3  EBike bike = new EBike();
4
5  Vehicle v;
6  RegisteredVehicle rv;
7
8  v = dudu;
9  dudu = v;
10 dudu = hunk;
11 rv = v;
12 rv = new RegisteredVehicle();
```

```
13 v = rv;
14 dudu.printVehicleInfo();
15 bike.getNumberPlate();
```

(b) Bestimmen Sie für den unten gegebenen Code den statischen Typ der gegebenen Variablen in den relevanten Zeilen (8, 9, 11, 13, 15). Bestimmen Sie außerdem für diesen Code die Menge der möglichen dynamischen Typen der Variablen. Gehen Sie davon aus, dass *wantRiksha* eine durch Benutzerabfrage initialisierte boolesche Variable ist.

```
1  Car dudu = new Car("Du-Du 1234");
2  Riksha hunk = new Riksha("HO-NK 123");
3  EBike bike = new EBike();
4
5  Vehicle v1;
6  Vehicle v2;
7  RegisteredVehicle rv;
8  v1 = hunk;
9  v2 = bike;
10 if (wantRiksha) {
11     rv = hunk;
12 } else {
13     rv = dudu;
14 }
15 rv.getNumberPlate();
```

(c) Was wird in den Zeilen 3 und 4 des folgenden Programms ausgegeben? Begründen Sie Ihre Antwort unter Nutzung der vorgestellten Fachwörter.

```
1 Vehicle v1 = new EBike();
2 RegisteredVehicle v2 = new Car("Du-Du 1234");
3 v1.printVehicleInfo();
4 v2.printVehicleInfo();
```

2 Teil B - Heimarbeit

Aufgabe 1 OOP

Ziel: Ziel der Aufgabe ist es, einen Teststand zu implementieren, auf dem verschiedene Autos auf ihre Abgaswerte hin untersucht werden sollen.

Aufgabenstellung: Nehmen Sie der Einfachheit an, es existieren in einem geschlossenen System zwei verschiedene Autotypen: Tennis und M6. Tennis wird vom Hersteller Informatiker Wagen (IW) produziert und der M6 von den Sächsischen Motoren Werken (SMW). Jedes Fahrzeug besitzt neben einem Hersteller auch eine Farbe und einen Motor. Motoren können sowohl Diesel als auch Benziner sein und besitzen einen Effizienzkoeffizienten (zwischen 0 und 0,99) und eine aktuelle Drehzahl.

Jedes Fahrzeug kann seinen Benzin- bzw. Dieselverbrauch und den CO2-Ausstoß messen, indem es diese Werte über den Motor ausliest.

Jedes Fahrzeug kann einem Teststand zugewiesen werden. Ein Teststand besitzt dabei ein Testverfahren, das die Inputdaten für den Fahrzeugtest festlegt. Die Klasse Test besitzt die Operation „performTest“, die wiederum dem übergebenen Fahrzeug eine Reihe von Drehzahlen übergibt und den jeweiligen CO2-Ausstoß erhält.

Hinweise: Implementieren Sie alle Klassen unter Beachtung der Vererbungsstruktur und nutzen Sie polymorphe Aufrufe und dynamisches Binden. Achten Sie auch darauf, ihre Datenstrukturen situationsabhängig passend zu wählen. Weiterhin gibt es wie immer auf Code Style bis zu 50% der Punkte. Als Abgabe dienen die Klassen am Ende der Teilaufgaben.

(a) *(20 Punkte)* Implementieren Sie die Klassen Producer, Engine, Vehicle, Tennis und M6. Sie können auch weitere Klassen implementieren, falls Sie das für Ihre Lösung für sinnvoll erachten. Überlegen Sie sich dabei einen geeigneten Algorithmus, mithilfe dessen der Motor auf Basis des Effizienzkoeffizienten und der Drehzahl seinen CO2-Ausstoß berechnet.

Hinweis: Ihr Algorithmus sollte sinnvoll sein, muss jedoch nicht zwangsläufig realistisch sein.

(b) *(10 Punkte)* Implementieren Sie die Klassen Teststand und Test. Simulieren Sie einen Testlauf in einer main-Methode. Geben Sie die Ergebnisse auf der Konsole aus.

(c) *(6 Punkte)* Fügen Sie ein weiteres Fahrzeug Tennis-Smart hinzu, das im Test die CO2-Ausstoßwerte um 60% verringert. Erweitern Sie Ihren Test.

Aufgabe 2 Clean Code

Nutzen Sie Ihr Eclipse, um den Code aus dem Repository unter `https://github.com/RSS-PSE-WS1920/exercise-sheet-11.git` auszuchecken.

(a) *(5 Punkte)* Lesen Sie das Programm und beschreiben Sie, was es wohl machen soll. Geben Sie für 2 Beispieleingaben an, welche Ausgaben Sie erwarten würden.

(b) *(3 Punkte)* Geben Sie eine Beispieleingabe an, die das Programm mit einem unerwarteten Ergebnis quittiert. Begründen Sie, wieso dieses Ergebnis vom Programm fehlerhafterweise ausgegeben wird.

(c) *(8 Punkte)* Nennen Sie mindestens 4 Verletzungen von Style-Regeln und begründen Sie jeweils mit einem Beispiel aus dem Code.

(d) *(10 Punkte)* Implementieren Sie ein Programm, welches die erhoffte Funktionalität realisiert. Dabei müssen Sie nicht unbedingt das existierende Programm retten, sondern können auch neu beginnen. Ebenso müssen nicht exakt die gleichen Ausgaben bei gleichen Eingaben rauskommen. Es reichen inhaltlich sinnvolle Ausgaben. Achten Sie dabei selbst auf sauberen Code und unsere Style-Regeln.

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 Übungsaufgaben im Unterricht

Aufgaben in der Schule und der Universität können sich in ihrer Art deutlich unterscheiden. In dieser Aufgabe sollen Sie sich anhand der zweiten Präsenzaufgabe dieses Blattes genau darüber Gedanken machen.

Dafür gilt folgendes Unterrichtsszenario: Ihre Schülerinnen und Schüler sollen die zweite Präsenzaufgabe dieses Übungsblattes bearbeiten. Sie dürfen davon ausgehen, dass alle erforderlichen Inhalte bereits in den vorangegangenen Stunden unterrichtet wurden und die Schülerinnen und Schüler die benötigten Fachwörter und Theorien kennen.

*Hinweis: Sie sollten die **zweite Präsenzaufgabe** dieses Übungsblattes für diese Aufgabe bereits bearbeitet haben.*

(a) Diskutieren Sie mit Ihrem Team, welche Unterschiede Sie in Bezug auf Aufgaben `allgemein` zwischen der Schule und der Universität feststellen.

(b) Diskutieren Sie mit Ihrem Team, was den Schülerinnen und Schülern bei der Aufgabe Schwierigkeiten bereiten könnte und was nicht.

(c) Überlegen Sie sich mit Ihrem Team, welche sogenannte "Sozialform" für diese Aufgabe geeignet sein könnte und warum.
Hinweis: Unter "Sozialformen" versteht man - zusammengefasst - die Art und Weise, wie die Schülerinnen und Schüler zusammen arbeiten. Man unterscheidet zwischen vier Sozialformen:
- Klassenunterricht oder auch "Frontalunterricht" / "Unterrichtsgespräch" genannt.
- Gruppenarbeit
- Partnerarbeit
- Einzelarbeit

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!