HOCHSCHULE RUHR WEST
UNIVERSITY OF APPLIED SCIENCES

16.09.2021

Klausur

Programmierung 2

Prüfer: Prof. Dr. Malte Weiß

Bewertung

1	2	3	4	**Gesamt**	**Note**
46	16	18	20	100	

Zum Ablauf der Prüfung

- **Zeit für die Lösung:** 4 Stunden
- **Beginn:** 10:00 Uhr, **Ende:** 14:00 Uhr am 16.09.2021
- **Lösung *wo* hochladen:** in E-Learning im Kursraum der Vorlesung
- **Lösung *wie* hochladen (Form):** Zip-Datei im Format *prog2-exam-VORNAME-NACHNAME-MATRIKELNUMMER.zip*.
  Beispiel: Student *Mathias Bauer* mit Matrikelnummer *10001234* gibt eine Zip-Datei prog2-exam-Mathias-Bauer-10001234.zip ab.
- **Ordnerstruktur:** Jede Aufgabe liegt in einem entsprechenden Unterordner in der Zip-Datei. Siehe aufgabenspezifische Konventionen.
- **Vorlage:** Nutzen Sie die Vorlage, die dieser Klausur beigefügt ist.
- **Nicht erlaubte Hilfsmittel/Hilfe:** Kommunikation digital oder persönlich, direkt oder indirekt mit anderen Menschen (**keine** SMS, WWW-Kommunikation, Telefon, E-Mail, Chat, etc.)
- **Kompilierbarkeit:** Lösungen *müssen* kompilieren, ansonsten werden 50 % der Aufgabenpunkte abgezogen.
- **Fragen:** Für Fragen während der Klausurzeit nutzen Sie bitte das Forum im E-Learning Bereich der Veranstaltung. Preisgabe von Lösungen oder Ansätzen führt zu Nichtbestehen der Veranstaltung inkl. Fehlversuch.
- **Eidesstattliche Versicherung**: Sie müssen eine eidesstattliche Versicherung abgegeben. Die Klausur gilt als nicht angetreten, wenn Sie die eidesstattliche Versicherung nicht unterschrieben einreichen.

Hinweise zur Online-Prüfung

Gemäß § 63 Abs. 5 Satz 1 Hochschulgesetz NRW macht die Hochschule Ruhr West von dem Recht Gebrauch, von den Prüflingen eine Versicherung an Eides Statt (Anlage) zu verlangen und abzunehmen, dass die Prüfungsleistung von ihnen selbstständig und ohne unzulässige fremde Hilfe erbracht worden ist. Von diesem Recht macht die HRW im Rahmen von Online-Prüfungen Gebrauch. Die Prüflinge werden hiermit darüber belehrt, dass die Abgabe einer falschen Versicherung an Eides Statt gemäß § 156 Strafgesetzbuch zu einer Freiheitsstrafe bis zu drei Jahren oder Geldstrafe führen kann. Eine fahrlässige Falschabgabe einer Versicherung an Eides Statt kann gemäß § 161 Strafgesetzbuch zu einer Freiheitsstrafe bis zu einem Jahr oder Geldstrafe führen. Es wird darauf hingewiesen, dass eine Täuschung im prüfungsrechtlichen Sinne vorliegt, sofern nicht der Prüfling selbst, sondern eine andere Person, in ihrem:seinem Namen die Prüfung ablegt. Zudem liegt eine Täuschung im prüfungsrechtlichen Sinne vor, wenn der Prüfling andere als die von dem Prüfer zugelassenen Hilfsmittel nutzt. Das Vorliegen einer Täuschung oder eines Täuschungsversuchs führt dazu, dass die Prüfung mit der Note 5,0 bewertet wird. Zudem kann das Vorliegen einer Täuschung oder eines Täuschungsversuchs gemäß § 63 Abs. 5 Satz 6 Hochschulgesetz NRW sowie § 12 Abs. 3 e) der Einschreibungsordnung HRW zur Exmatrikulation führen

Sie haben zwei Möglichkeiten, die Eidesstattliche Versicherung abzugeben:

1. Drucken Sie den Text auf der folgenden Seite aus, unterschreiben Sie ihn und scannen/fotografieren Sie ihn ab. Speichern Sie das Foto als Bilddatei im Ordner „Eidesstattliche Versicherung“ in Ihrer Abgabe-ZIP-Datei ab.
2. Kopieren Sie den Text der folgenden Seite in eine Textdatei, setzen Sie ihren Namen und ihre Matrikelnummer darunter und speichern Sie ihn im Ordner „Eidesstattliche Versicherung“ in Ihrer Abgabe-ZIP-Datei ab.

Eidesstattliche Versicherung

Hiermit versichere ich Folgendes an Eides statt:

Ich versichere, dass ich die hier genannte Prüfung selbstständig abgelegt habe.

Zudem versichere ich, dass ich während der hier genannten Prüfung lediglich die von dem Prüfer zugelassenen Hilfsmittel verwendet habe. Ich bestätige, dass ich keinerlei andere Hilfsmittel genutzt habe.

Mir ist bekannt, dass eine eidesstattliche Versicherung eine nach den §§ 156, 161 Strafgesetzbuch (StGB) strafbewehrte Bestätigung der Richtigkeit meiner Erklärung ist. Mir sind die strafrechtlichen Folgen einer unrichtigen Erklärung bekannt.

Nach § 156 StGB wird mit Freiheitsstrafe bis zu drei Jahren oder mit Geldstrafe bestraft, wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung vorsätzlich falsch abgibt. Nach § 161 StGB wird mit Freiheitsstrafe bis zu einem Jahr oder mit Geldstrafe bestraft, wer vor einer zur Abnahme einer Versicherung an Eides Statt zuständigen Behörde eine solche Versicherung fahrlässig falsch abgibt.

Mir ist weiterhin bekannt, dass eine Täuschung im prüfungsrechtlichen Sinne vorliegt, sofern nicht ich selbst, sondern eine andere Person in meinem Namen die Prüfung ablegt. Zudem liegt eine Täuschung im prüfungsrechtlichen Sinne vor, wenn ich andere als die von dem Prüfer zugelassenen Hilfsmittel nutze. Das Vorliegen einer Täuschung oder eines Täuschungsversuchs führt dazu, dass die Prüfung mit der Note 5,0 bewertet wird. Zudem kann das Vorliegen einer Täuschung oder eines Täuschungsversuchs gemäß § 63 Abs. 5 Satz 6 Hochschulgesetz NRW sowie § 12 Abs. 3 e) der Einschreibungsordnung HRW zur Exmatrikulation führen.

Ort, Datum	*Unterschrift der:des Studierenden*

Aufgabe 1: Fahrräder, Wettrennen und Statistiken (46 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A1 gespeichert werden.

In dieser Aufgabe modellieren Sie **Fahrräder**, setzen **Wettrennen** an und erheben **Statistiken**.

Betrachten Sie zunächst diese Klassenhierarchie von Fahrrädern:

- Die abstrakte Klasse **AbstractMobilityDevice** ist ein abstraktes Beförderungsgerät und bildet die Oberklasse aller weiteren Klassen. Ein abstraktes Beförderungsgerät besitzt einen Namen (name) und eine maximale Geschwindigkeit (maxSpeed).
- Fahrräder sind für diverse Arten von Radsport geeignet oder ungeeignet. **RaceStyle** ist eine Aufzählung mit den Elementen MARATHON, MOUNTAINBIKING und SHOWOFF. Sie stellen den Radmarathon, das Mountainbiking und Show-Wettkämpfe dar.
- Die Markierungsschnittstelle **CanPerformTrick** wird von jedem Fahrrad implementiert, mit dem Tricks durchgeführt werden können.
- Ein Fahrrad wird durch die namensgebende Klasse **Bicycle** dargestellt und erbt von AbstractMobilityDevice. Ein Fahrrad besitzt typischerweise eine Anzahl an Schaltgängen (numberOfGears). Jedes Fahrrad passt zu einer oder mehreren bestimmten Radsportarten (suitableRaceStyles). suitableRaceStyles nutzt für die Ablage der Radsportarten ein Array.
- Ein Cruiser-Fahrrad wird in der Klasse **CruiserBike** repräsentiert und erbt von Bicycel. Ein Cruiser-Fahrrad eignet sich nur für Radmarathons.
- Ein Klapprad befindet sich in einer Klasse **FoldingBike**, erbt von Bicycle und eignet sich für keinen Radsport.
- Ein Mountainbike befindet sich in der namensgebenden Klasse **MountainBike**. Mountainbikes haben einen Federweg (springTravel), der bis auf zwei Nachkommastellen genau angegeben werden kann. Hat das Mountainbike keine Federung, so wird springTravel auf 0.0 gesetzt.
- Ein BMX-Fahrrad wird in der Klasse **BMXBike** dargestellt und erbt von Mountainbike. Typischerweise hat ein BMX-Fahrrad niemals eine Federung und eignet sich nur für Show-Wettkämpfe. Außerdem können auf einem BMX-Fahrrad Tricks ausgeführt werden.
- Ein Downhill-Fahrrad ist ein Mountainbike, welches sich in einer Klasse **DownhillBike** befindet. Es eignet sich prinzipiell für alle drei hier angegebenen Radsportarten und besitzt auch eine Federung.
- Ein Dirtbike ist ein Mountainbike, welches sich in einer Klasse **DirtBike** befindet. Es eignet sich vor allem für Show-Wettkämpfe, einige Fahrer:innen nutzen es aber auch für Mountainbiking. Auf einem Dirtbike können Tricks durchgeführt werden und harte Landungen durch die Federung gedämpft werden.

Ihre Aufgaben sind nun die folgenden:

a) Implementieren Sie die oben dargestellte Klassenhierarchie für Fahrräder.

Jede Klasse implementiert angemessene Konstruktoren, Getter & Setter, die toString-Methode und alle Schnittstellen entsprechend der oben beschriebenen Klassenhierarchie. Die toString-Methode gibt wie üblich alle Attribute der jeweiligen Klasse menschenlesbar auf dem Bildschirm aus. Achten Sie darauf, dass auch Attribute der Oberklassen ausgegeben werden sollten. (18 Punkte)

b) Sie erstellen nun eine einfache Simulation einer Wettkampf-Strecke.

Fügen Sie eine Klasse **RacingTrack** hinzu, welche eine Wettkampf-Strecke für den Radsport darstellt. Eine Wettkampf-Strecke wird einer Art von Radsport zugeordnet (typeOfRace). Des Weiteren können Fahrräder verwaltet werden. Für letzteres legt der Standard-Konstruktor eine leere Liste oder ein leeres Array an (listOfBicycles). (3 Punkte)

Die Klasse erhält nun weitere Methoden:

- Eine Methode **addBike** um Fahrräder hinzuzufügen. Wenn ein Fahrrad sich nicht für die Art von Radsport der Wettkampf-Strecke eignet, wird eine zu erstellende Ausnahme **InappropriateBicycleTypeException** geworfen. Die Ausnahme bekommt ein Fahrrad (Bicycle) übergeben und enthält eine sinnvolle Fehlermeldung. (4 Punkte)
- Eine Methode **removeBike** um Fahrräder zu entfernen. (1 Punkte)
- Eine sinnvolle **toString**-Methode, welche die Anzahl aller Fahrräder sowie jedes Fahrrad einzeln menschenlesbar als String zurückgibt. (1 Punkt)

c) Die zuständige Wettkampfbehörde benötigt zur Erfassung von Statistiken im Radsport Ihre Hilfe.

Fügen Sie hierfür eine weitere Klasse **RacingAuthorityUtils** hinzu. Diese implementiert nun folgende Methoden:

- Eine statische Methode **printBikesPerformingTricks**. Diese erhält eine Liste vom Typ AbstractMobilityDevice und gibt auf der Konsole alle Fahrräder aus, mit welchen ein Trick ausgeführt werden kann. (4 Punkte)
- Eine statische Methode **getAverageSpringTravel**. Diese erhält eine Liste vom Typ AbstractMobilityDevice und gibt den durchschnittlichen Federweg aller Mountainbikes zurück. (5 Punkte)
- Eine statische Methode **computeNameToSpeedMap**. Diese erhält eine Liste von Fahrrädern (AbstractMobilityDevice). Sie erzeugt auf Basis der übergebenen Liste eine Statistik, die den Namen der Fahrräder auf Ihre Maximalgeschwindigkeit abbildet. (6 Punkte)

Hinweis: Achten Sie auf eventuell notwendige Prüfungen, ob ein Objekt eine bestimmte notwendige Klasse/Schnittstelle implementiert.

d) Schreiben Sie ein Hauptprogramm in einer Klasse **RacingMain**. Das Hautprogramm demonstriert die Klassenhierarchie, die Wettkampf-Strecke sowie alle Methoden aus der Hilfsklasse RacingAuthorityUtils. (4 Punkte)

Aufgabe 2: Generics (16 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A2 gespeichert werden.

Sämtliche Java-Klassen sollen sich im Paket **de.hrw.p2.exam.generics.pricelist** befinden.

Eine Preisliste ist eine Zuordnung von Gegenständen zu Preisen. Beispiel:

Gegenstand	**Preis**
Kaffee	1,00 €
Curry-Wurst	3,50 €
Jägerschnitzel	5,50 €
Pommes	2,50 €

Schreiben Sie eine generische Klasse **PriceList**, die eine Preisliste modellieren soll. Sie soll zwei Typplatzhalter bieten:

- einen Platzhalter für den Datentyp des Gegenstands (im obigen Beispiel z.B. String)
- einen Platzhalter für den Preis (im obigen Beispiel z.B. Double). Der Preis-Platzhalter bietet die Möglichkeit, den Zahlentyp für den Preis zu bestimmen (Long, Integer, usw.), sollte aber zwingend **von Number erben**.

Die Klasse soll folgende Funktionen bieten:

- **add** fügt einen Gegenstand zur Preisliste hinzu. Existiert der Eintrag schon, wird die Java-Exception **IllegalArgumentException** geworfen.
- **getPrice** liefert den Preis für einen Gegenstand zurück. Existiert dieser nicht, wird die Java-Exception **NoSuchElementException** geworfen.
- **remove** entfernt einen Gegenstand aus der Preisliste. Existiert dieser nicht, wird die Java-Exception **NoSuchElementException** geworfen.
- **getAveragePrice** liefert den Durchschnittspreis aller Gegenstände als **double** zurück. Ist die Preisliste leer, soll -1 zurückgegeben werden.
- **toString** liefert eine String-Darstellung der Preisliste zurück.

a) Implementieren Sie die generische Klasse PriceList gemäß den obigen Anforderungen.
(12 Punkte)

b) Schreiben Sie ein Hauptprogramm, das alle Methoden der Klasse PriceList demonstriert.
Hinweis: Hierbei bitte auf eine sinnvolle Konsolenausgabe achten.
(4 Punkte)

Aufgabe 3: Streams (18 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A3 gespeichert werden.

Sämtliche Java-Klassen sollen sich im Paket **de.hrw.p2.exam.streams** befinden.

Die beigefügte Klasse **Planet** modelliert einen Planeten. Ein Planet verfügt über die folgenden Eigenschaften:

- **Name**
- **Durchmesser** in km
- **Anzahl der Bewohner**
- **Existiert Sauerstoff auf dem Planeten**: ja/nein

Vervollständigen Sie die Klasse **PlanetHelper** unter der ausschließlichen Verwendung von Streams. Es gelten folgende Auflagen:

- Verwendung von Schleifen und if/switch sind verboten
- Es dürfen keine lokalen Variablen definiert werden. Alle Aufgaben lassen sich mit einer Zeile Code lösen.

Folgende Methoden müssen vervollständigt werden (siehe auch JavaDoc in der Klasse PlanetHelper):

- **printPlanetNameToInhabitantsList** gibt eine Liste von Planeten aus. Für jeden Planeten wird der Name, die Einwohnerzahl und die Tatsache, ob es Sauerstoff auf dem Planeten gibt, ausgegeben. (3 Punkte)
- **getAverageDiameter** erhält ein Array von Planeten und gibt den Durchschnitts-Durchmesser zurück. (3 Punkte)
- **countInhabitedPlanetsWithOxigen** erhält eine Liste von Planeten und gibt die Anzahl der enthaltenen Planeten zurück, die Sauerstoff enthalten. (3 Punkte)
- **getSortedPlanetNames** erhält eine Liste von Planeten und gibt eine sortierte Liste von Planetennamen zurück. (3 Punkte)
- **getAverageInhabitantCountOfInhabitedPlanets** erhält ein Array von Planeten und gibt die durchschnittliche Einwohnerzahl der bewohnten Planeten zurück. Sind alle übergebenen Planeten unbewohnt, wird -1 zurückgegeben.
  Hinweis: Unbewohnte Planeten werden bei der Durchschnittsberechnung nicht berücksichtigt. (3 Punkte)

Implementieren Sie ein Hauptprogramm, das alle Methoden testet. (3 Punkte)

Aufgabe 4: Dateiströme (20 Punkte)

Die Lösung der Aufgabe muss im ZIP-Archiv im Ordner A4 gespeichert werden.

Sämtliche Java-Klassen sollen sich im Paket **de.hrw.p2.exam.cityreader** befinden.

Bei dieser Aufgabe sollen Städtedaten aus einer URL gelesen und in eine Datenstruktur geschrieben werden. Die Beispiel-URL[1] für diese Aufgabe lautet:

https://codingprof.hs-rw.de/files/exam-files/GdIuP_SS2020/worldcities.csv

Die URL führt zu einer CSV-Datei, die Daten zu weltweiten Städten enthält. Ihre Aufgabe ist es, diese Datenstruktur einzulesen:

a) Schreiben Sie eine Java-Klasse **City**, die eine Stadt repräsentiert. Sie enthält:

- Attribute für den **Namen der Stadt**, den **Namen der Landes** sowie **Breiten**- und **Längengrad** (im Englischen *latitude* und *longitude*). Breiten- und Längengrad sollten als double-Werte modelliert werden
- Einen Konstruktor, der alle Attribute setzt
- Getter und Setter
- Eine toString-Methode, die eine lesbarbaren String-Repräsentation des Objekts erstellt

(3 Punkte)

b) Schreiben Sie eine statische Methode **readWorldList** in einer Klasse **CityReaderUtil**. Die Methode erhält eine URL und gibt eine Liste von **City**-Objekten zurück.

Die Methode liest die CSV-Datei im obigen Format ein, extrahiert die relevanten Attribute (siehe Aufgabenteil a) und erzeugt eine Liste von **City**-Objekten, die anschließend zurückgegeben werden.

Hinweis: Die Einträge der CSV-Datei sind durch Anführungszeichen eingeschlossen. Beachten Sie bitte auch, dass es Städtenamen gibt, die ein Komma enthalten.
(12 Punkte)

c) Stellen Sie sicher, dass die von der Methode **readWorldList** zurückgegebene Liste von Städten erst nach Land und dann – bei gleichem Landesnamen – nach Stadt sortiert wird.
(3 Punkte)

b) Schreiben Sie ein Hauptprogramm, das die Funktionalität testet.
(2 Punkte)

Hinweis: Sollte die oben genannte URL nicht erreichbar sein, können Sie die beigefügte Datei **worldcities.csv** im Ordner **Ressourcen** bei sich lokal speichern und über eine lokale URL (file:///) einlesen.

[1] Quelle: https://simplemaps.com/data/world-cities, Download am 07.09.2021