Einführung in die Programmierung

WS 2017/2018, Blatt 4

Bauhaus-Universität Weimar

Prof. Norbert Siegmund
Nathalie Dittrich

Ausgabe: 18.12.2017
Abgabetermin: Montag, 08.01.2018, 11:00 Uhr

Bitte lesen Sie die folgenden Informationen zum Übungsablauf **sorgfältig** durch.

Grundsätzlich – wenn nicht anders angegeben – sind die Lösungen zu den Übungen zu Programmierung I jeden **Dienstag bis spätestens 12:00 Uhr** an die jeweiligen Tutoren per E-Mail zu schicken.

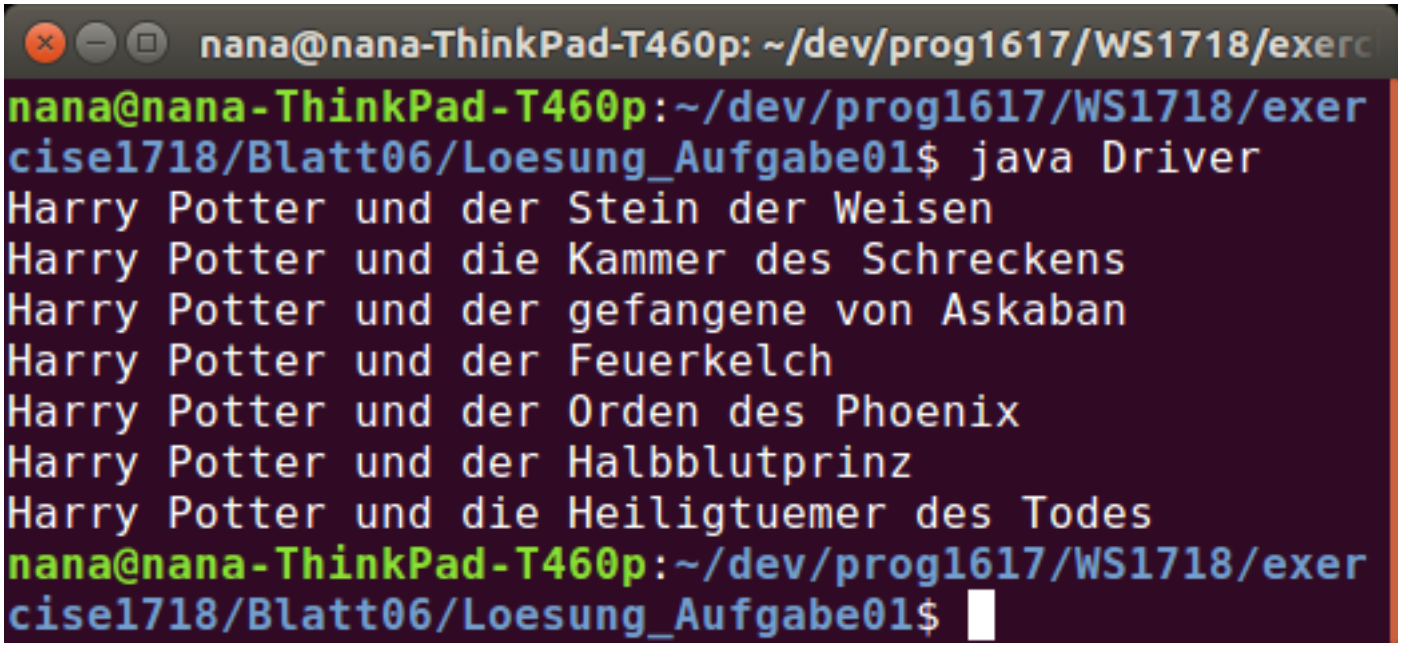
Schreiben Sie bitte im Betreff Ihrer E-Mail Ihre **Teamnummer** sowie die Nummer des Übungsblattes. In der E-Mail schreiben Sie bitte zusätzlich Ihren **Namen** und **Matrikelnummer**. Die Lösungen für Sie bitte als Java Dateien als Anlage hinzu. Es werden **keine** kompilierten Dateien, wie *.class oder *.jar angenommen.

Übungen müssen von **minimal ein** und **maximal zwei** Studierenden aus derselben Übungsgruppe in einem festen Team bearbeitet werden (Ausnahmen nur auf Anfrage beim Übungsleiter). Pro Team soll die Lösung nur einmal abgegeben werden. Aufgaben sollen **im Team gelöst** und nicht nur vom Team abgegeben werden. Sie müssen mindestens **50%** dieser Punkte für eine Zulassung zur Prüfung erreichen. Das **Abschreiben** identischer Lösungen wird jeweils mit 0 Punkten bewertet.

Bei Fragen oder Unklarheiten wenden Sie sich bitte **vor der Abgabe** des Übungsblattes an den Übungsleiter (per E-Mail oder persönlich). Es soll nie jemand sagen müssen: „Wir haben die Aufgabe nicht verstanden und konnten sie daher nicht bearbeiten."
Weitere Informationen, wie aktuelle Ankündigungen, die Angaben finden Sie online (`https://www.uni-weimar.de/de/medien/professuren/intelligente-softwaresysteme/lehre/`) unter Einführung in die Programmierung

Aufgabe 1 Getter und Setter (7 Punkte)

Anbei finden Sie die Java-Datei `Driver.java`. Die entsprechende Klasse beinhaltet eine `main`-Methode. Schreiben Sie alle benötigten Klassen und Methoden, damit der Code in `Driver` ausgeführt werden kann. Dabei darf in der `Driver.java` **nichts** verändert werden. Das Programm sollte die folgende Ausgabe erzeugen:



Bepunktung:

- Programm kompilliert: 0,5 Punkte
- `Driver.java` unverändert: 0,5 Punkte
- Alle Klassen vorhanden: 2 Punkte
- Korrekte Struktur innerhalb der Klassen: 4 Punkte

Aufgabe 2 Rekursion (4 Punkte)

Die Fibonacci-Folge beschreibt eine Folge von natürlichen Zahlen, bei denen sich jede Zahl aus der Summe der beiden vorherigen Zahlen zusammensetzt (`https://de.wikipedia.org/wiki/Fibonacci-Folge`):

$$0, 1, 1, 2, 3, 5, 8, 13, 21, ...$$

Schreiben Sie eine **rekursive** Methode `public int fibonacci(int n)`, welche einen Integer als Index entgegen nimmt und die entsprechende Zahl in der Fibonacci-Folge zurück gibt.

Bepunktung:

- Programm kompilliert: 0,5 Punkte
- Methodenkopf unverändert: 0,5 Punkte
- Rekursion verwendet: 2 Punkte
- Korrekter Output: 1 Punkt

Aufgabe 3 Switch-Case (4 Punkte)

Das folgende Codefragment berechnet einen Raumschiffflug von einer Raumstation zum Mond:

```
// travel distance in kilometers
int distance = 363000;
// accleration in m/s^2
float acceleration = 10.0f;
// maximal speed in m/s
float maxSpeed = 2000.0f;
// seconds needed to accelerate (or brake)
float accelerationTime = (maxSpeed / acceleration);
// traveled distance in m after acceleration
float accelerationDistance = (acceleration * accelerationTime * accelerationTime) / 2;
// remaining distance in m without acceleration and braking
float remainingDistance = (distance * 1000) - 2 * accelerationDistance;
// seconds needed to travel the remaining distacne
float remainingTime = remainingDistance / maxSpeed;
// travel time: acceleration + flight at maxSpeed + braking
System.out.println((remainingTime + 2 * accelerationTime) / 3600 );
```

Ändern Sie das Programm so ab, dass sie vor der Berechnung zwischen verschiedenen Raumschiffmodellen wählen können:

(a) `Orion` - Beschleunigung: $10\,m/s^2$, Maximale Reisegeschwindigkeit: $2000\,m/s$

(b) `Galactica` - Beschleunigung: $8.3\,m/s^2$, Maximale Reisegeschwindigkeit: $3200\,m/s$

(c) `Excelsior` - Beschleunigung: $230\,m/s^2$, Maximale Reisegeschwindigkeit: $130000\,m/s$

Zusätzlich sollen weitere Reiseziele möglich sein. Ermöglichen Sie die Eingabe der folgenden Reiseziele:

(a) `Mond` - Entfernung: $363.000\,km$

(b) `Mars` - Entfernung: $56.000.000\,km$

(c) `Pluto` - Entfernung: $588.000.000\,km$

Realisieren Sie die oben genannte Aufgabe **mit Hilfe von `Switch-Case` Statements**!

Bepunktung:

- Programm kompilliert: 0,5 Punkte
- Korrektheit des Programms: 0,5 Punkte
- `Switch-Case` Statements korrekt: 3 Punkte

Aufgabe 4 Referenzen (4 Punkte)

Es sollen von sechs Primzahlen die Wurzel und das Quadrat berechnet werden und jeweils in einem Array gespeichert werden. Dazu wurde folgender Code implementiert:

```
double[] primeNumbers = {3, 5, 7, 11, 13, 17};

double[] sqrtOfPrimeNumbers = primeNumbers;
for (int i = 0; i < 6; ++i)
{
  sqrtOfPrimeNumbers[i] = Math.sqrt(sqrtOfPrimeNumbers[i]);
}

double[] squaredPrimeNumbers = primeNumbers;
for (int i=0; i<6; ++i)
{
  squaredPrimeNumbers[i] = squaredPrimeNumbers[i] * squaredPrimeNumbers[i];
}
```

Lassen Sie sich die Ergebnisse des obigen Codes ausgeben um deren Korrektheit zu überprüfen. Erklären Sie warum die Ergebnisse (nicht) korrekt sind und implementieren Sie gegegebenenfalls eine korrekte Lösung.

Bepunktung:

- Fehler erkannt: 1 Punkt
- Begründung korrekt: 2 Punkte
- Korrketur richtig: 1 Punkt

Aufgabe 5 Listen für Einsteiger (4 Punkte)

Gegeben ist folgende Listenstruktur:

```
public class Cell {
  public int content;
  public Cell next;     // null for the last element
  public Cell prev;     // null for the first element
}

public class List {
  private Cell front;  // null if empty

  public void removeYAfterX(int x, int y) { ... }
}
```

Implementieren Sie die angebene Methode `removeYAfterX`. Die Methode soll die Liste so ändern, dass `direkt` nach jedem Element, das den Wert x trägt, **ein** vorhandenes Element mit Wert y gelöscht wird. Zum Beispiel wird aus der Liste $1, 9, 9, 2, 1, 9, 1, 4, 9, 5$ nach Aufruf der Methode mit $x = 1$ und $y = 9$ die Liste $1, 9, 2, 1, 1, 4, 9, 5$.

Bepunktung:

- Programm kompilliert: 0,5 Punkte
- Korrektheit des Programms: 0,5 Punkte
- Methode richtig implementiert: 3 Punkte

Aufgabe 6 Orchester (13 Punkte)

Implementieren Sie eine Klasse `Musician`.
Ein Musiker hat:

- einen Namen.
- ein Instrument (Aufzählungstyp) -> es gibt:
  - Violine
  - Cello
  - Kontrabass
  - Querfloete
  - Klarinette
  - Oboe
  - Trompete
  - Posaune
  - Triangel
- einen Konstruktor, welcher Name und Instrument setzt.
- einen Erfahrungsgrad (Aufzählungstyp) -> es gibt:
  - Anfänger (0-3 Jahre Erfahrung)
  - Fortgeschritten (4 oder mehr Jahre Erfahrung)

  Setzen Sie die benötigten Jahre an Erfahrung für den Erfahrungsgrad über den entsprechenden Konstruktor des enums.

Außerdem hat ein Musiker die folgenden Methoden:

- `getInstrument`: Gibt das entsprechende Instrument zurück.
- `setErfahrungsgrad(int jahre)`: Setzt den Erfahrungsgrad indem die Methode übergeben bekommt wie viele Jahre der entsprechende Musiker schon das Instrument spielt. Anhand der Jahre wird der Erfahrungsgrad entweder auf *Anfänger* oder *Fortgeschritten* gesetzt.
- `toString()`: Gibt den Namen, das Instrument und den Erfahrungsgrad des Musikers zurück.

Implementieren Sie zudem eine Klasse `Orchestra`.
Ein Orchester hat:

- Musiker (Array), ein Gründungsjahr, einen Namen und einen Dirigenten (selbst geschriebener komplexer Datentyp).
- einen Konstruktor, welcher den Namen, sowie das Gründungsjahr setzt.

Außerdem hat ein Orchester die folgenden Methoden:

- `setMusicians`: Setzt das Musiker Array
- `selectMusiciansForComposition(Instrument[] instrumentsNeeded)`: sucht alle Musiker raus, welche eines der genannten Instrumente spielen und gibt diese als Array zurück.
- `toString()`: Gibt den Namen, das Gründungsjahr und zeilenweise für jeden Musiker Name und Instrument zurück.

Schreiben Sie zum Schluss eine weitere Klasse mit einer `main`-Methode. In der Methode wird ein neues Orchester erstellt mit mindestens einem Musiker pro Instrument. Geben Sie daraufhin aus welche Musiker zusammen ein Stück für Violine, Oboe und Triangel spielen können.
Bepunktung:

- Alle Klassen implementiert: 4 Punkte
- Ausgabe korrekt: 1 Punkt
- Aufzählungstypen richtig implementiert: 3 Punkte
- Klassenstruktur überall korrekt: 2 Punkte
- Methoden richtig implementiert: 3 Punkte