Hochschule
Bonn-Rhein-Sieg
University of Applied Sciences
Fachbereich Informatik
Prof. Dr. Rudolf Berrendorf

Einführung in die Programmierung
WS 19/20
Ausgabe : 16.12.2019
Bearbeiten bis : 05.01.2020

Übung 12

Aufgabe 1 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Nehmen Sie folgenden Klassen und Schnittstelle als Basis:

```
public class TestKlasse2 {
    public static void main(String[] args) {
        System.out.println(gibZahl(4711));
    }
    // in dieser Methode duerfen Sie Ergaenzungen vornehmen
    public static String gibZahl(int wert) {
        ZahlKlasse z = new ZahlKlasse(wert);
        return z.toString();
    }
}

// ab hier keine Aenderung erlaubt

public interface ZahlFunktionalitaet {
    public int getZahl();
    public String toString();
}

public abstract class ZahlKlasse implements ZahlFunktionalitaet {
    private int zahl;
    public ZahlKlasse(int zahl) {
        this.zahl = zahl;
    }
    public int getZahl() {
        return zahl;
    }
    public String toString() {
        return "" + zahl;
    }
}
```

Dieser Code ist so nicht übersetzbar, weil in `main` versucht wird, ein Objekt der Klasse `ZahlKlasse` zu instanziieren, was aber aufgrund der Eigenschaft, dass diese Klasse abstrakt ist, nicht möglich ist.

An der Schnittstelle sowie an der Klasse `Zahlklasse` darf jetzt aber nichts geändert werden (zum Beispiel, weil dies zwischen Ihnen und einem zweiten Programmierer so vereinbart wurde). Wir werden auch genau diesen gegebenen Source-Code im Praktomat anwenden.

Realisieren Sie durch entsprechende Ergänzungen in der Methode `gibZahl` in `TestKlasse2`, dass ein Objekt mit dem im Parameter `wert` übergebenen Wert instanziiert werden kann (ein Aufruf `getZahl` liefert dann auch den Wert, mit dem dieses Objekt erzeugt wurde) und dass ein Aufruf der `toString`-Methode dieses Objekts als Resultat aber einen String liefert, in dem der Zahlwert des Objekts (also der in `zahl` gespeicherte Wert) verdoppelt ist.

Hinweis: anonyme Klasse.

Beispiel: Mit dem Beispielparameterwert 4711 bei der Erzeugung soll als Ergebnis auf dem Bildschirm der Wert 9422 (=2*4711) erscheinen.

Geben Sie im Praktomat nur ihren Code der Klasse `TestKlasse2` in einer Datei `TestKlasse2.java` ab. Geben Sie **keinen Code** für die Schnittstelle `ZahlFunktionalitaet` und die Klasse `Zahlklasse` ab.

Achten Sie darauf, dass die Klasse `TestKlasse2` wie angegeben `public` ist.

Aufgabe 2 :
Gegeben sind folgende Klassen:

```
public class TestKlasse {
    public static void main(String[] args) {
        ZahlKlasse z = new ZahlKlasse(1);
        for(int i=2; i<10; i++) {
            z = new ZahlKlasse(z.getZahlWert());
        }
        System.out.println(z);
    }
}

public class ZahlKlasse {
    private static int anzahl;
    private Zahl zahl;
    public ZahlKlasse(int zahl) {
        ++anzahl;
        this.zahl = new Zahl(zahl);
    }
    public static int getAnzahl() {
        return anzahl;
    }
    public int getZahlWert() {
        return zahl.getZahl();
    }
    public String toString() {
        return "" + zahl.getZahl();
    }
}

public class Zahl {
    private int zahl;
    public Zahl(int zahl) {
        this.zahl = zahl * ZahlKlasse.getAnzahl();
    }
    public int getZahl() {
        return zahl;
    }
}
```

Geben Sie eine äquivalente Lösung an, in der die Funktionalität der Klasse **`Zahl`** als innere Klasse in **`ZahlKlasse`** realisiert wird. Passen Sie die Klassen dabei so an, dass Sie die Eigenschaft einer inneren Klasse ausnutzen. Hinweis: Zugriff auf Attribute.

Aufgabe 3 :
Gegeben ist die Klasse und die Schnittstelle, die Sie beide nicht verändern dürfen:

```
class DingMitFarbe {
    protected String farbe;
    public DingMitFarbe(String farbe) { this.farbe = farbe; }
}
interface FarbeLesen {
    public String getFarbe();
}
```

Geben Sie eine abstrakte Klasse **`Fahrzeug`** an, die von **`DingMitFarbe`** erbt und die Schnittstelle **`FarbeLesen`** realisiert.

Geben Sie weiterhin eine von **`Fahrzeug`** sinnvoll abgeleitete konkrete Klasse **`PKW`** an, die nur einen Konstruktur enthält, so dass folgendes **`main`** erfolgreich übersetzbar und ausführbar ist mit der Ausgabe **`gruen`**:

```
public static void main(String[] args) {
  System.out.println(new PKW("gruen").getFarbe());
}
```

Aufgabe 4 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Ein Polynom ist eine Funktion $P(x) : \mathbb{R} \to \mathbb{R}$ mit $P(x) = \sum_{i=0}^{n} a_i x^i, n \geq 0, a_0, \ldots, a_n \in \mathbb{R}$. Die Werte $a_0, \ldots, a_n$ heißen Koeffizienten.
Oder anders ausgedrückt: durch Angabe von Koeffizientenwerten $a_0, \ldots, a_n \in \mathbb{R}$ ist ein Polynom im obigen Sinne definiert. Für ein Argument x lässt sich dieses Polynom dann auswerten mit $P(x) = \sum_{i=0}^{n} a_i x^i$.

Beispiel: Die Koeffizienten $a_2 = 1, a_1 = 2, a_0 = 3$ definieren das Polynom $P(x) = 1 \cdot x^2 + 2 \cdot x^1 + 3 \cdot x^0 = x^2 + 2x + 3$.
Der Wert dieses Polynoms an der Stelle 2 wäre dann $P(2) = 4 + 4 + 3 = 11$.

Entwickeln Sie eine Klasse **`Polynom`** in Java zur Repräsentation von Polynomen.

1. Über einen Konstruktor soll sich durch alleinige Angabe von Koeffizienten in Form eines Feldes von Fließkommawerten (**`double[]`**) ein neues Polynom anlegen lassen. Durch die Länge des übergebenen Feldes ist implizit auch der Grad des Polynoms bestimmt. Der Koeffizient mit der niedrigsten Wertigkeit steht im Feld an der Positon 0.
   Beispiel: Durch die Übergabe des Feldes **`[3,2,1]`** würde obiges Beispielpolynom definiert.
   Sorgen Sie außerdem dafür, dass nur normalisierte Polynome entstehen, in denen der höchstwertige Koeffizient ungleich 0 ist (sofern es sich nicht um das konstante Nullpolynom handelt).
   Beispiel: Werden bei der Erzeugung eines Polynoms die Koeffizienten $2, 1, 0$ (für $0 \cdot x^2 + 1 \cdot x + 2$) angegeben, so soll daraus implizit das Polynom $2, 1$ (für $1 \cdot x + 2$) entstehen.
   Hinweis: **`public Polynom(double[] koeffizienten)`**
2. Entwickeln Sie weiterhin eine Instanzmethode **`toString()`**, die das Polynom als lesbaren String zurück gibt. Beachten Sie, dass Polynome üblicherweise absteigend von der höchsten zur niedrigsten Potenz notiert werden.
   Hinweis: **`public String toString()`**

   Beispiel: **`1*x^2 + 2*x^1 + 3*x^0`**
3. Geben Sie schließlich eine Instanzmethode **`public double auswerten(double x)`** an, die für ein Argument **`x`** den Wert des Polynoms an der Stelle **`x`** zurückgibt (Beispiel siehe oben).
4. Geben Sie eine Klassenmethode **`public static int getAnzahl()`** an, die die Anzahl der bis dahin erzeugten Polynome zum Zeitpunkt des Methodenaufrufs liefert.

Schreiben Sie eine zweite Klasse **`PolynomTest`**. Diese Klasse soll eine **`main`**-Methode besitzen, in der Sie das Polynom $P(x) = 1 \cdot x^2 + 2 \cdot x + 3$ (als Objekt) erzeugen, dieses Polynomobjekt auf dem Bildschirm in lesbarer Form ausgeben und das Polynom auswerten an der Stelle 2 und diesen Wert auf dem Bildschirm ausgeben. Geben Sie anschließend auch noch die Anzahl der erzeugten Polynome auf dem Bildschirm aus. Diese drei Ausgaben sollen jeweils in einer eigenen Zeile stehen.

Die Ausgabe zu obigem Beispiel wäre dann:

```
1.0*x^2 + 2.0*x^1 + 3.0*x^0
11.0
1
```

Überlegen Sie sich, wo mögliche Fehler in ihrer Polynomklasse auftreten können. Geben Sie an den Stellen in ihrem Programm eine Meldung auf dem Bildschirm aus, die jeweils beginnt mit **`Fehler:`**

Freiwillige Erweiterung: Recherchieren Sie, was das Horner-Schema im Zusammenhang mit der Polynomauswertung bedeutet. Implementieren Sie dieses Verfahren zur Auswertung des Polynoms.

Aufgabe 5 :
Erweitern Sie ihre Polynomklasse, indem Sie folgende Methoden hinzufügen:

1. Eine Methode, die zu einem Polynom P ein zweites Polynom (-objekt) Q addiert. Das Ergebnis der Methode ist ein neues drittes Polynom $R = P + Q$. Beachten Sie, dass die beiden Polynome P und Q auch unterschiedlichen Grad haben können.
   Beispiel: $P(x) = 3x + 1$, $Q(x) = 3x^2 + 2x$. Die Addition dieser beiden Polynome ergibt ein Polynom $R(x) = 3x^2 + 5x + 1$.
2. Eine Methode, die die erste Ableitung zu einem gegebenen Polynom bestimmt. Das Ergebnis ist ein neues Polynom.
   Zur Erinnerung: Die erste Ableitung eines Polynoms $\sum_{i=0}^{n} a_i x^i = a_n x^n + \ldots + a_1 x^1 + a_0$ ist
   $\sum_{i=1}^{n} i a_i x^{i-1} = n a_n x^{n-1} + \ldots + 3a_3 x^2 + 2a_2 x + a_1$.

Beispiel: Die erste Ableitung des Polynoms $2x^3 + 1x^2 + 5x + 7$ (als Feld $[7,5,1,2]$) ist das Polynom $6x^2 + 2x + 5$ (als Feld $[5,2,6]$).

3. Erweitern Sie ihre `main`-Methode in der Klasse `PolynomTest`, indem Sie ein weiteres Polynom $Q(x) = 1 \cdot x^3 + 2 \cdot x^2 + 3 \cdot x + 4$ erzeugen und zu dem vorher erzeugten Polynom dieses neue Polynom addieren.
   Achtung:
   Das Ergebnis ist laut Wirkungsweise der Additionsfunktion ein neues Polynom R. Fahren Sie mit R wie folgt fort:
   - Geben Sie R auf dem Bildschirm aus.
   - Bilden Sie von R die erste Ableitung R' und geben Sie das Ergebnis auf dem Bildschirm aus.
   - Bilden Sie von der ersten Ableitung die erste Ableitung (also die zweite Ableitung R'' von R). Geben Sie auch diese auf dem Bildschirm aus.
   - Werten Sie R, R' und R'' an der Stelle 4711 aus.

Aufgabe 6 :
Erweitern Sie ihre Polynomklasse, indem Sie

1) die Multiplikation von Polynomen ermöglichen. Überlegen Sie sich (oder recherchieren Sie), wie die Polynommultiplikation $P(x) \cdot Q(x)$ arbeitet und implementieren Sie dazu eine Instanzmethode, die als Ergebnis ein neues Polynom liefert. Testen Sie dies in `PolynomTest`.
2) ein Polynom integrieren (Stammfunktion bestimmen). Für ein Polynom $P(x) = \sum_{i=0}^{n} a_i x^i$ hilft dabei:
   $\int P(x)dx = \sum_{i=0}^{n} \frac{a_i x^{i+1}}{i+1} + C$, wobei wir $C = 0$ setzen wollen. Testen Sie in `PolynomTest` diese Methode, indem Sie R' wieder integrieren. Bis auf den konstanten Term a_0 müsste als Ergebnis R herauskommen.

Aufgabe 7 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Schreiben Sie eine Klasse `Punkt` zu einem 2D-Punkt mit ganzzahligen x- und y-Koordinaten. Sehen Sie einen Konstruktor zum Erzeugen eines solchen Punktes vor, dem zwei ganzzahlige Werte übergeben werden. Die Klasse soll weiterhin eine Methode `public String toString()` enthalten, die zu einem Punkt einen String der Form `(x,y)` erzeugt, wobei x,y die entsprechenden Punktangaben zu diesem Punkt sind.

Schreiben Sie weiterhin eine Klasse `Weg`, die Wege beschreiben kann. Ein Weg hat $n > 0$ Wegpunkte $p_1, ..., p_n$, die jeweils ein Punkt sind. Nutzen Sie dazu in der Klasse `Weg` sinnvoll die Punkt-Klasse im Zusammenhang mit einem Feld, das zu jedem Zeitpunkt die gleiche Länge hat wie die Länge des aktuellen Weges. Es gibt keine Längenbeschränkung für Wege!
Hinweis: In einer Referenzvariablen lässt sich die Referenz auf ein Feld durch eine Referenz auf ein anderes (längeres) Feld ersetzen.

Sehen Sie folgende Methoden in der Klasse `Weg` vor, die genau nach diesen Vorgaben existieren müssen:

- Es gibt genau einen Konstruktor, der zu zwei int-Werten x, y einen Weg mit genau einem Punkt erzeugt. Direkt über einen Konstruktor lassen sich also nur Wege der Länge 1 erzeugen.
- Es gibt eine Instanzmethode `public int getAnzahl()`, die die Anzahl der Wegpunkte des Weges liefert.
- Es gibt eine Instanzmethode `public void verlaengern(Weg w)`, die zu dem eigenen Weg (sprich das Bezugsobjekt) einen weiteren Weg anhängt, indem alle Punkte des zweiten Weges an den eigenen Weg angehängt werden. Dadurch wird also die eigene Instanz verändert.
  Beispiel: Zum Weg $w1 = (0,0)$ wird der Weg $w2 = (1,1)$ angehängt: `w1.verlaengern(w2)`. Danach gilt $w1 = (0,0) - (1,1)$, er enthält also zwei Punkte.
  Hängt man an diesen Weg wiederum den Weg $w3 = (3,4)$ an (`w1.verlaengern(w3)`), so gilt anschließend $w1 = (0,0) - (1,1) - (3,4)$.
- Es gibt eine Methode `public String toString()`, die einen Weg als String liefert in der Form $(x_1, y_1) - (x_2, y_2) - ... - (x_n, y_n)$, wobei die x_i, y_i die ganzzahligen Koordinaten des entsprechenden Punktes sind. Nutzen Sie dabei sinnvoll die Methode `toString` der Punkt-Klasse.

Schreiben Sie in einer weiteren Testklasse **`WegTest`** ein **`main`**. Erzeugen Sie darin einen Weg $w1 = (1,1)$ und einen Weg $w2 = (2,2)$. Lassen Sie beide Wege nacheinander auf dem Bildschirm ausgeben.

Dann verlängern Sie $w1$ um $w2$ und lassen wieder den Weg $w1$ auf dem Bildschirm ausgeben.

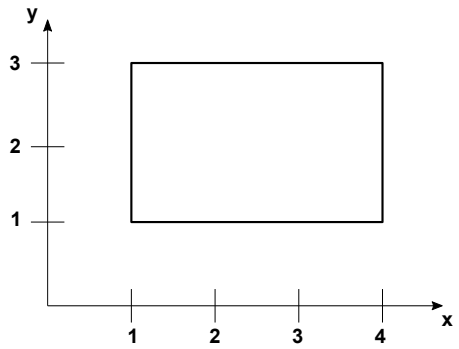
Die Gesamtausgabe muss dann exakt so sein:

```
(1,1)
(2,2)
(1,1)-(2,2)
```

Hinweis: Es gibt nicht eine Methode in irgendeiner dieser Klassen, die *als Argument* ein Feld übergeben bekommt. Denken Sie in Wegen und Punkten (und Instanzen)!

Aufgabe 8 :
Gegeben sei eine Klasse **`Punkt`** (siehe Vorlesung). Gehen Sie davon aus, dass diese Klasse zwei Instanzmethoden **`getX()`** und **`getY()`** habe, die die entsprechende Koordinate als Ganzzahl zurückgeben.

Schreiben Sie eine Klasse **`Rechteck`**, die ein achsparalleles Rechteck in der Ebene beschreibt (dessen Seiten also parallel zu der x- und y-Achse des Koordinatensystems liegen). Die Klasse beschreibe das Rechteck durch die x-Koordinaten seines "linken" und "rechten" Randes sowie die y-Koordinaten seines "unteren" und "oberen" Randes. Die Koordinaten seien ganzzahlig. Siehe folgende Abbildung für ein Beispiel. Dort ist der linke Rand bei 1, der rechte Rand bei 4, der untere Rand bei 1 und der obere Rand bei 3.



1. Geben Sie der Klasse einen Konstruktor, der die 4 Koordinaten als Argumente erhält und initialisiert.
2. Geben Sie der Klasse einen Konstruktor, der zwei **`Punkt`**-Objekte als Argumente erhält, welche diagonal gegenüberliegende Eckpunkte des Rechtecks beschreiben, und der die Koordinaten entsprechend initialisiert. Für das Beispiel wären das also die Punkte (1,1) und (4,3), jeweils als Objekte. Aus diesen Punkten lassen sich leicht die notwendigen internen Rechteckdaten im Konstruktor berechnen.
3. Schreiben Sie Instanzmethoden **`eckeLinksUnten`**, **`eckeLinksOben`**, **`eckeRechtsUnten`**, **`eckeRechtsOben`**, die jeweils ein **`Punkt`**-Objekt mit dem entsprechenden Wert zurück geben.
4. Schreiben Sie eine Instanzmethode **`enthaelt`**, die ein **`Punkt`**-Objekt als Argument nimmt und einen Wahrheitswert zurückgibt, der beschreibt, ob der Punkt innerhalb des Rechtecks (incl. Rand) liegt. Überlegen Sie sich, wie dieser Test aussehen muss.
5. Schreiben Sie eine Instanzmethode **`enthaelt`**, die ein **`Rechteck`**-Objekt als Argument nimmt und einen Wahrheitswert zurückgibt, der beschreibt, ob jenes Rechteck (incl. Rand) innerhalb des durch die Instanz beschriebenen Rechtecks (incl. Rand) liegt.
6. Schreiben Sie eine Instanzmethode **`huelle`**, die ein **`Punkt`**-Objekt als Argument nimmt und ein **`Rechteck`**-Objekt zurückgibt, das gerade so groß (und nicht größer) ist, dass es das durch die Instanz beschriebene Rechteck und den Punkt enthält.
7. Schreiben Sie eine Instanzmethode **`huelle`**, die ein **`Rechteck`**-Objekt als Argument nimmt und ein **`Rechteck`**-Objekt zurückgibt, das gerade so groß (und nicht größer) ist, dass es die Rechtecke enthält, die durch die Instanz und das Argument beschrieben sind.

Hinweis: In den Teilaufgaben sind einfache Überlegungen notwendig, auf die man zum Beispiel anhand einer einfachen Skizze leicht selbst kommen kann.

Aufgabe 9 (1 Punkt) :
Dies ist eine Praktomataufgabe!

Beschreiben Sie in einer Klasse **`Person`** eine Person. Zu einer Person gehört ein Nachname. Über einen Konstruktor lässt sich eine Person erzeugen, wobei man den Namen als String übergeben muss. Sehen Sie eine Getter-Funktion vor, um den Namen auslesen zu können. Eine Methode **`public String toString()`** liefert (ebenfalls) den Nachnamen. Beispiel: **`Meier`**

Hinweis: Diese Klasse hat also mindestens diese Konstruktoren/Methoden:

```
Person(String name);
public String getName();
public String toString();
```

Ein Fußballspieler (Klasse **`Fussballspieler`**) ist eine Person, die ein jährliches Gehalt bekommt (ein ganzzahliger Wert), das man zusätzlich zu einem Namen bei der Konstruktion als zweites Argument angeben muss. Sehen Sie eine Getter-Methode für das Einkommen vor. Eine Methode **`public String toString()`** liefert den Namen und das Gehalt (als einfache Zahl, ohne Währungszeichen etc.) durch ein Leerzeichen getrennt als String. Beispiel: **`Hummels 1000000`**

Hinweis: Diese Klasse hat also mindestens diese Konstruktoren/Methoden:

```
Fussballspieler(String name, int einkommen);
public int getEinkommen();
public String toString();
```

Geben Sie eine Klasse **`Mannschaft`** an, die aus genau 5 Fussballspielern besteht (Hallenfußball). Es gibt einen Konstruktor, der aus genau 5 einzelnen Fußballspielern (Java-Objekte werden übergeben, nicht deren Namen!) eine Mannschaft erzeugt. Es werden also auch 5 Argumente an den Konstruktor übergeben. Sehen Sie eine Methode **`public int einkommen()`** vor, die das Gesamteinkommen dieser Mannschaft liefert (Summe aller Einzeleinkommen). Weiterhin soll es eine Methode **`public String toString()`** geben, die die Aufstellung der Mannschaft in der folgenden Form liefert (als String):

```
1. Horn
2. Kimmich
3. Hummels
4. Reus
5. Draxler
```

Hinweis: **`\n`**

Hinweis: Diese Klasse hat also mindestens diese Konstruktoren/Methoden:

```
Mannschaft(Fussballspieler s1, Fussballspieler s2, Fussballspieler s3, Fussballspieler s4,
           Fussballspieler s5);
public int einkommen();
public String toString();
```

Geben Sie eine Klasse **`Stadion`** mit einem **`main`** an. Im **`main`** werden zwei Mannschaften erzeugt und 45.000 Zuschauer (dies sind einfache Personen). Als Namen der Zuschauer geben Sie **`Zuschauer-i`** an, wobei i der i-te Zuschauer ist, bei 1 beginnend. Bei den Fußballspielern geben Sie **`Spieler-x-i`** an, wobei x **`rot`** (1. Mannschaft) oder **`blau`** (2. Mannschaft) ist und mit i wiederum die Spieler durchgezählt werden (bei 1 beginnend). Als Einkommen des i-ten Spielers geben Sie für die rote Mannschaft $10.000 * i$ an, für die blaue Mannschaft $20.000 * i$.

Lassen Sie anschließend auf dem Bildschirm die Namen der roten Mannschaft ausgeben (Format siehe oben), eine Leerzeile, dann die Namen der blauen Mannschaft, eine Leerzeile und dann zu beiden Mannschaften jeweils das Gesamteinkommen. Geben Sie anschließend noch den Namen des ersten Zuschauers aus.

Ihre Ausgabe müsste also für die oben gezeigten Beispieldaten wie folgt aussehen:

```
1. Spieler-rot-1
2. Spieler-rot-2
3. Spieler-rot-3
4. Spieler-rot-4
5. Spieler-rot-5
```

1. Spieler-blau-1
2. Spieler-blau-2
3. Spieler-blau-3
4. Spieler-blau-4
5. Spieler-blau-5

150000
300000
Zuschauer-1