Heinrich Heine Universität Düsseldorf

**Programmierung – WS 2025/26**
Dr. Markus Brenneis
Abgabefrist für Feedback:
24.11.25, 10 Uhr

# 6. Übungsblatt

## Aufgabe 1: Rechteck nach VL 9

Schreiben Sie eine Klasse `Rectangle`, welche ein Rechteck mit beliebigen, ganzzahligen Seitenlängen repräsentiert. Die Seitenlängen könnten Sie z. B. in privaten Instanzvariablen `width` und `height` speichern.

Implementieren Sie einen (paket-privaten) Konstruktor `Rectangle(int width, int height)`, welcher Breite und Höhe des Rechtecks entgegennimmt und die Instanzvariablen entsprechend setzt, sowie einen zweiten Konstruktor ohne Argumente, bei dem `width` und `height` jeweils mit 0 initialisiert werden (Sie überladen also den Konstruktor).

Implementieren Sie außerdem die folgenden Instanz-Methoden:

- `int area()`: Gibt die Fläche des Rechtecks zurück.
- `int perimeter()`: Gibt den Umfang des Rechtecks zurück.
- `boolean isSquare()`: Gibt genau dann `true` zurück, wenn das Rechteck ein Quadrat ist.
- `public String toString()`: Gibt eine String-Repräsentation des Rechtecks zurück, die aus `*`-Symbolen besteht. Hierbei sollen zwei Sterne immer durch ein Leerzeichen getrennt sein (vgl. Beispiel unten).

Sie müssen in dieser Aufgabe keine Fehlerfälle (z. B. negative Längen) beachten. Verwenden Sie nur *private* Instanzvariablen.

*Es ist nicht notwendig, aber empfehlenswert, dass Sie zum Testen eine Klasse mit main-Methode schreiben und dort prüfen, ob Ihre Klasse* `Rectangle` *wie gewünscht funktioniert. Das könnte z. B. wie folgt aussehen:*

Beispiel zum Testen — java

```java
Rectangle rectangle = new Rectangle(5, 3);
Rectangle square = new Rectangle(2, 2);

System.out.println(rectangle.isSquare());
System.out.println(rectangle.perimeter());
System.out.println(rectangle.area());
System.out.println(rectangle.toString());

System.out.println(square.isSquare());
System.out.println(square);
```

```
false
16
15
* * * * *
* * * * *
* * * * *

true
* *
* *
```

*Die Sternchen-Ausgabe darf (muss aber nicht) mit einem Zeilenumbruch am Ende und/oder Leerzeichen am Zeilenende abschließen.*

## Aufgabe 2: Zulassung nach VL 9

Für die folgende Aufgabe gibt es bereits eine Lösung in den vorgegebenen Dateien zu diesem Übungsblatt. Ihre Aufgabe: Verunstalten Sie den Code, sodass er gegen möglichst viele Regeln für guten Programmierstil verstößt, die Sie im Laufe der Vorlesung kennengelernt haben.[1] Der Code soll aber weiterhin für gleiche Eingaben die gleiche Ausgabe wie zuvor generieren.

Geben Sie alle zum geänderten Programm gehörenden Java-Dateien mit einem Kommentar `// Feedback gewünscht` ab, wenn Sie Feedback erhalten wollen.[2]

> Für das Organisationsteam einer Vorlesung wurde ein Java-Programm `Zulassung` entwickelt, das anhand einer Punktetabelle ausgibt, welche Studis die Zulassung erreicht haben. Die Zulassung ist erreicht, wenn auf der ersten Hälfte (abgerundet) der Blätter mind. 20 % der Gesamtpunktzahl (Zulassung Teil I), auf den übrigen Blättern mind. 20 % der Gesamtpunktzahl (Teil II) und auf allen Blättern mind. 50 % der Gesamtpunktzahl erreicht wurden (Gesamtzulassung).
>
> Eine Punktetabelle ist in folgendem Format gespeichert:
>
> ```
> Name,Blatt 1,Blatt 2,Blatt 3,Blatt 4,Blatt 5,Blatt 6,Blatt
>   7,Blatt 8
> Tai Becker,2,10,13,3,4,5,7,18
> Sascha Maier,0,0,0,20,17,20,18,2
> Kim Müller,20,20,18,20,10,0,0,19
> Kari Nguyen-Kim,1,10,15,4,8,0,9,12
> Katara Schmidt,0,0,0,20,17,20,20,20
> ```
>
> Im Quelltext ist diese Tabelle fest als String hinterlegt.[3] Das Programm erhält beim Aufruf als Argument die maximal erreichbare Gesamtpunktzahl (Summe aller Blätter); falls kein Argument angegeben wird, gibt es eine Fehlermeldung. Das Programm gibt dann wie folgt nach dem Namen (sortiert wie in der Einga-

[1] Auf der Kursseite finden Sie eine Zusammenfassung der Regeln.

[2] Je nachdem, was sie so alles „anstellen“, darf Ihr Programm am Ende auch nur aus einer einzigen Datei `Zulassung.java` bestehen. Wenn Sie mehrere Dateien abgeben, können diese auch gleichzeitig in einem Schritt hochgeladen werden.

[3] Wir schauen uns später an, wie wir die Daten aus einer Datei einlesen können.

be) aus, ob die Zulassungen Teil I und II sowie die Gesamtzulassung erworben wurden:

```
Tai Becker,true,true,true
Sascha Maier,false,true,false
Kim Müller,true,true,true
Kari Nguyen-Kim,true,true,false
Katara Schmidt,false,true,false
```

Katara Schmidt hat also nur den zweiten Teil der Zulassung geschafft, aber weder Teil I, noch die Gesamtzulassung.

## Aufgabe 3: Elektrische Widerstände nach VL 10

*Lesen Sie sich zuerst die Aufgabe komplett durch, um einen Überblick über alle Anforderungen zu erhalten. Falls Teile der Aufgabenstellung nicht verständlich sind, melden Sie sich z. B. im Forum.*

Widerstände (engl. resistors) sind elektrische Bauteile, deren Widerstandswert (resistance) in Ohm (Ω) angegeben wird. Der Widerstandswert ist eine reelle Zahl (Kommazahl). Mehrere Widerstände können kombiniert werden, um einen größeren oder kleineren Gesamtwiderstand zu erhalten. In dieser Aufgabe wollen wir mehrere Klassen schreiben, mit denen es möglich ist, den Gesamtwiderstand einer Kombination von einzelnen Widerständen zu berechnen.

Einzelne Widerstände lassen sich wie folgt kombinieren:

- Ein einzelner Widerstand – sein Gesamtwiderstand ist sein Widerstandswert selbst.
- Zwei oder mehr Widerstände $R_1, \ldots, R_n$ können hintereinander geschaltet werden (Reihenschaltung, Series Circuit). Der Widerstandswert $R$ der Reihenschaltung beträgt in diesem Fall:
$$R = R_1 + \cdots + R_n$$
In dieser Übungsaufgabe betrachten wir nur den Fall $n = 2$, also $R = R_1 + R_2$.
- Zwei oder mehr Widerstände $R_1, \ldots, R_n$ können nebeneinander geschaltet werden (Parallelschaltung, Parallel Circuit). Für den Widerstandswert $R$ der Parallelschaltung gilt in diesem Fall:
$$\frac{1}{R} = \frac{1}{R_1} + \cdots + \frac{1}{R_n}$$
In dieser Aufgabe sollen Sie nur den Fall $n = 2$ umsetzen; dann lässt sich die Formel vereinfachen zu:
$$R = \frac{R_1 \times R_2}{R_1 + R_2}$$

1. Definieren Sie ein Interface `Resistor` für Widerstände (die potentiell aus Einzelwiderständen bestehen) mit den folgenden Methoden:
   - `double resistance()`: gibt den Widerstandswert des Widerstands zurück
   - `int resistorCount()`: gibt die Gesamtzahl der einzelnen Widerstände innerhalb des Widerstands zurück

2. Definieren Sie eine Klasse `SingleResistor`, welche das Interface `Resistor` (sinnvoll) implementiert. Der Widerstandswert soll bei der Erzeugung (als Parameter an den Konstruktor) angegeben werden und anschließend unveränderlich sein.
3. Definieren Sie eine Klasse `SeriesCircuit`, die `Resistor` implementiert und einen Gesamtwiderstand bestehend aus zwei in Reihe geschalteten Widerständen repräsentiert. Der Konstruktor erwartet zwei beliebige Widerstände (also Objekte beliebiger Klassen, die `Resistor` implementieren); `SeriesCircuit` soll nur diesen einen Konstruktor haben.
4. Definieren Sie eine Klasse `ParallelCircuit`, die `Resistor` implementiert und einen Gesamtwiderstand bestehend aus zwei parallel geschalteten Widerständen repräsentiert. Der Konstruktor erwartet zwei beliebige Widerstände; `ParallelCircuit` soll nur diesen einen Konstruktor haben.
5. *Bonusaufgabe:* Im Programmierpraktikum 1 werden Sie das Open-Closed-Prinzip (OCP) kennenlernen. Das OCP besagt, dass Code offen für Erweiterungen und geschlossen für Änderungen sein soll. Ein `SeriesCircuit`-Konstruktor kann ein `ParallelCircuit`-Objekt entgegennehmen, obwohl `ParallelCircuit` erst nach `SeriesCircuit` implementiert worden ist. Drücken Sie in eigenen Worten aus, inwiefern hier `SeriesCircuit` offen für Erweiterungen ist und gleichzeitig für die diese Erweiterung nicht geändert werden muss. Wie spielen hier Interfaces eine Rolle? *Wenn Sie individuell Feedback zu Ihrer Antwort erhalten wollen, geben Sie Ihre Antwort als Datei* `bonus.txt` *ab; schreiben Sie oben in die Datei* `// Feedback gewünscht`.

Benutzen Sie in dieser Aufgabe, wie in Java üblich, den Datentyp `double` für Kommazahlen. Sichtbarkeiten sollen so klein wie möglich sein.

Beispiel für die Verwendung der Klassen:

Beispiel `java`

```java
SingleResistor r1 = new SingleResistor(100);
SingleResistor r2 = new SingleResistor(200);
SingleResistor r3 = new SingleResistor(300);

ParallelCircuit r23 = new ParallelCircuit(r2, r3);
SeriesCircuit r123 = new SeriesCircuit(r1, r23);

System.out.println(r123.resistance());    // 220.0
System.out.println(r123.resistorCount()); // 3
```

*Tipp:* Testen Sie Ihre Implementierung selbst mit einer `main`-Methode. Sie können dafür die folgende Schaltung mit sieben einzelnen Widerständen nachbauen (Gesamtwiderstand ca. 873,56 Ω):