HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Heinrich-Heine-Universität Düsseldorf
Mathematisch-Naturwissenschaftliche Fakultät
Institut für Informatik, Abteilung Betriebssysteme
Prof. Dr. Michael Schöttner
Kevin Beineke
Ludmila Himmelspach
Stefan Nothaas

12. Übungsblatt „Programmierung“ im Wintersemester 2016/17

Abgabe bis 30. Januar 2017, 07:59 Uhr über das Abgabesystem

Aufgabe 1: Interfaces — 5 Punkte

Geben Sie für jede der folgenden Aussagen an, ob sie korrekt oder inkorrekt ist (ohne Begründung).

a) Interfaces können Variablen vererben

b) Es können mehrere Interfaces von einer Klasse implementiert werden

c) Es ist möglich eine Methode als `static` zu deklarieren

d) Es ist möglich eine Methode als `protected` zu deklarieren

e) Von einem Interface können Instanzen angelegt werden

Aufgabe 2: Vererbung & Polymorphie — 10 Punkte

a) Erstellen Sie einen Ableitungsbaum für die Klassen eines Biologieprogramms. Die Klassen, die im Ableitungsbaum auftauchen sollen sind:

Pflanze, Katze, Hai, Tier, Blauwal, Adler, Säugetier, Lebewesen, Maiglöckchen, Pinguin, Fisch, Sonnenblume, Vogel, Lachs

Erstellen Sie einen sinnvollen Ableitungsbaum, der alle genannten Klassen enthält. Sie sollen lediglich ein Bild / eine Zeichnung abgeben, keinen Code und auch keine Variablen und Methoden.

b) Im folgenden Programm sind einige Fehler versteckt. Finden Sie 5 der Fehler und erklären Sie diese.

```
1          public class Fahrzeug {
2            private int maxSpeed;
3
4            public Fahrzeug(int maxSpeed) {
5              this.maxSpeed = maxSpeed;
6            }
7
8            protected int getMaxSpeed() {
9              return maxSpeed;
10           }
11
12           public void move() {
13             System.out.print("moves on ");
14           }
15         }
16
17         public final class Landfahrzeug extends Fahrzeug {
18           public void move() {
19             System.out.print("moves on land");
```

```
20            }
21          }
22
23          public class Wasserfahrzeug {
24            public final void move() {
25              super.move();
26              System.out.print(" water");
27            }
28          }
29
30          public final class Traktor extends Landfahrzeug {
31            public final int maxSpeed = 40;
32
33            public Traktor() {
34              super();
35              super.maxSpeed = this.maxSpeed;
36            }
37
38            public void move() {
39              System.out.print("Traktor");
40              this.move();
41            }
42          }
43
44          public class Amphibienfahrzeug extends Landfahrzeug, Wasserfahrzeug {
45            public void move() {
46              System.out.print("Amphibienfahrzeug ");
47              super.move();
48              System.out.print(" with " + getMaxSpeed());
49            }
50          }
```

Aufgabe 3: Elektrische Widerstände — 15 Punkte

Widerstände sind elektrische Bauteile. Ihr Widerstandswert wird in Ohm (Ω) gemessen. Aus Widerständen lassen sich sogenannte Widerstandsnetze (mit eigenem Gesamtwiderstand) formen:

- Ein einzelner Widerstand ist das kleinste mögliche Netz.
- Zwei oder mehr Widerstände $R_1, \ldots, R_n$ können hintereinander geschaltet werden (Serienschaltung). Der Widerstandswert des Netzes beträgt in diesem Fall:

$$R = R_1 + \cdots + R_n$$

- Zwei oder mehr Widerstände $R_1, \ldots, R_n$ können nebeneinander geschaltet werden (Parallelschaltung). Der Widerstandswert des Netzes beträgt in diesem Fall:

$$R = \frac{R_1 \times \cdots \times R_n}{R_1 + \cdots + R_n}$$

Implementieren Sie Klassen zur Repräsentation derartiger Widerstandsnetze.

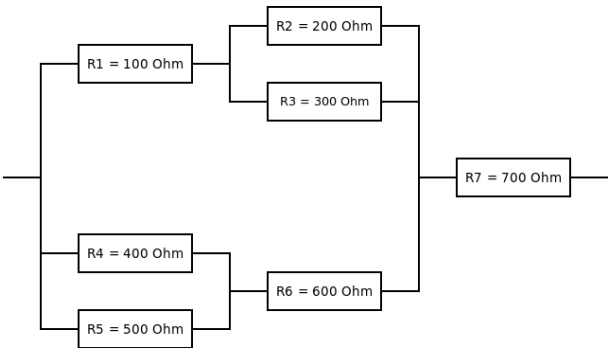
a) Definieren Sie ein Interface `Net` für die Repräsentation eines Wiederstandsnetz mit den folgenden Methoden:
Die Methode `ohm` gibt den Widerstandswert (als double) des Netzes zurück.
Die Methode `resistorCount` gibt die Anzahl der Widerstände innerhalb des Netzes zurück.

b) Definieren Sie eine Klasse `Resistor`, welche das Interface `Net` implementiert. Der Widerstandswert soll bei der Erzeugung angegeben werden und anschließend unveränderlich sein.

c) Definieren Sie eine Klasse `Serial`, die ein Wiederstandsnetz mit einer Serienschaltung repräsentiert. Der Konstruktor erwartet zwei Widerstandsnetze und speichert diese ab.

d) Definieren Sie eine Klasse `Parallel`, die ein Wiederstandsnetz mit einer Parallelschaltung repräsentiert. Der Konstruktor erwartet zwei Widerstandsnetze und speichert diese ab.

Schreiben Sie ein Programm `TestNet`, dass das folgende Widerstandsnetz aufbaut und den Gesamtwiderstand, sowie die Anzahl der Widerstände ausgibt.



Aufgabe 4: Fahrzeuge - Interfaces & Polymorphie 30 Punkte

In dieser Aufgabe soll mithilfe von Interfaces, Vererbung und Polymorphie Fahrzeuge und Verhalten (mit Einschränkungen) modelliert werden. Ein `PKW` sowie `LKW` und `Motorrad` sind `Fahrzeug`e. Spezialisierungen von `PKW`s sind ein `SUV` und ein `Coupe`. Zusätzlich kann ein `Anhaenger` entweder an einen `PKW` oder `LKW` angekoppelt werden.

Alle von Ihnen implementierten Klassen müssen zu den Attributen passende Getter-Methoden (und keine Setter-Methoden) sowie Konstruktoren zum Initialisieren der Attribute aufweisen.

a) Implementieren Sie die Klassen `Fahrzeug` (mit Attributen `sitze`, `raeder`, `gewicht`), `PKW`, `LKW` (mit Attribut `gewichtLadung`), `Motorrad`, `SUV` (mit Attribut `hatAllrad`) und `Coupe` (mit Attribut `hatFaltdach`). Achten Sie auf eine vollständige und korrekte Vererbungsstruktur.

b) Schreiben Sie ein Interface `Gesamtgewicht` mit einer Methode `getGesamtgewicht`. Implementieren Sie das Interface `Gesamtgewicht` in der Klasse `Fahrzeug` und überschreiben Sie ggf. die Methode des Interfaces in den Unterklassen von `Fahrzeug`, damit von jedem Fahrzeugtyp das korrekte Gesamtgewicht ermittelt wird.

c) Implementieren Sie eine Klasse `Anhaenger` mit den Attributen `anzahlAchsen`, `gewicht`, `gewichtLadung`. Der Anhänger soll das Interface `Gesamtgewicht` implementieren.

d) Schreiben Sie ein Interface `Anhaengerkupplung` welches Funktionalität zum Kuppeln (`void ankuppeln(Anhaenger a)`) und entkuppeln (`Anhaenger entkuppeln()`) eines Anhängers definiert sowie eine Methode `boolean gekuppelt()` bereitstellt.

e) Implementieren Sie das Interface `Anhaengerkupplung` in den Klassen `PKW` und `LKW` und passen Sie die Klassen so an, dass diese einen Anhänger besitzen können. Passen Sie die Methode `gesamtGewicht` hierbei entsprechend an.

f) Erstellen Sie in der `main`-Methode einer Test-Klasse folgende Instanzen: 2x LKW, 3x SUV, 2x Coupe, 1x Motorrad, 2x Anhänger. Hängen Sie einen Anhänger an einen LKW und den anderen an einen SUV. Ermitteln Sie das Gesamtgewicht aller erstellten Fahrzeuge. Nutzen Sie dabei die Vorteile von Polymorphie und Interfaces.