1 Aufgabe

Gegeben seien die nachfolgenden Java-Klassen:

```
abstract class GObjekt
{
    static int objektAnzahl = 0;
    protected int posX;
    protected int posY;

    abstract void zeichnen(Graphics g);
    static void objektHinzufuegen(GObjekt objekt)
    {
        // ...
    }
    static void objektEntfernen(GObjekt objekt)
    {
        // ...
    }
}
public class Rechteck extends GObjekt
{
    void zeichnen(Graphics g)
    {
        // ...
    }
}
public class Ellipse extends GObjekt
{
    protected int posX;
    protected int posY;

    void zeichnen(Graphics g)
    {
        // ...
    }
}
public class Kreis extends Ellipse
{
    void Zeichnen(Graphics g)
    {
        // ...
    }
}
```

Verdeutlichen Sie (wenn möglich) an diesem Beispiel die Begriffe Oberklasse, Unterklasse, Einfachvererbung, Mehrfachvererbung, abstrakte Klasse, Überschreiben, Verbergen, Polymorphismus, Klassenattribut, Objektattribut, Klassenoperation und Objektoperation. Testen Sie spielerisch den Zugriff auf Attribute und Methoden der verschiedenen Ebenen der Vererbungshierarchie.

2 Aufgabe

Gegeben sei die folgende Klasse Auto:

```
public class Auto {
  protected double neuwagenpreis;
  protected int baujahr;
  protected String modell;

  public Auto(double neuwagenpreis, int baujahr, String modell){
    this.neuwagenpreis = neuwagenpreis;
    this.baujahr = baujahr;
    this.modell = modell;
}

public double getPreis() {
    return neuwagenpreis;
  }

}
```

Erweitern Sie diese Klasse so, dass eine neue Klasse GebrauchtAuto entsteht, die zusätzlich die bereits gefahrenen Kilometer speichert. Schreiben Sie den dazu notwendigen Konstruktor und überschreiben Sie die Methode double getPreis() so, dass nach jeweils 20000 gefahrenen Kilometern der Neuwagenpreis des Wagens um 10% sinkt. Beachten Sie, dass der Händler einen Mindestpreis von 2000 für jeden Gebrauchtwagen angesetzt hat.

3 Aufgabe

Modellieren Sie die Klassen *AudioCD* und *VideoDVD* als Unterklassen einer gemeinsamen Oberklasse MedienKunstwerk. Folgende Daten müssen für CDs und für DVD's gespeichert werden:

- String titel
- int katalogNummer
- double preis
- int lauflaenge

Für CDs muss weiterhin gespeichert werden:

- String interpret
- int anzahlTracks
- String musikRichtung

Für eine DVD muss zusätzlich gespeichert werden:

- String hauptDarsteller
- String genre
- String format

a) Programmieren Sie die Klassen MedienKunstwerk, VideoDVD und AudioCD. Entscheiden Sie, welches Attribut zu welcher Klasse gehören soll. Die Attributte sollen alle als private deklariert werden. Programmieren Sie für die Klassen Konstruktoren, mit denen die Objektfelder initialisiert werden. Programmieren Sie Observatormethoden getTitel(), getKatalogNummer() etc, mit denen man die Objektfelder abfragen kann.
b) Versehen Sie die Klasse MedienKunstwerk mit einer Methode leseDaten(), die die Klasse javapack.Einlesen benutzt, um die Attribute der Klasse mit Werten zu besetzen, die der Benutzer auf der Konsole eingibt. In den Unterklassen AudioCD und VideoDVD soll die Methode derart überladen werden, dass auch die zusätzlichen Objektfelder besetzt werden. Soweit möglich, soll dabei auf die Methode der Oberklasse zurückgegriffen werden
c) Testen Sie die Klassen.

4 Aufgabe 3

Beantworten Sie die Fragen. Zur Beantwortung der Fragen 4) und 5) benötigen Sie die Dateien `UnterKlasse.java` und `OberKlasse.java`. Testen Sie Ihre Antworten

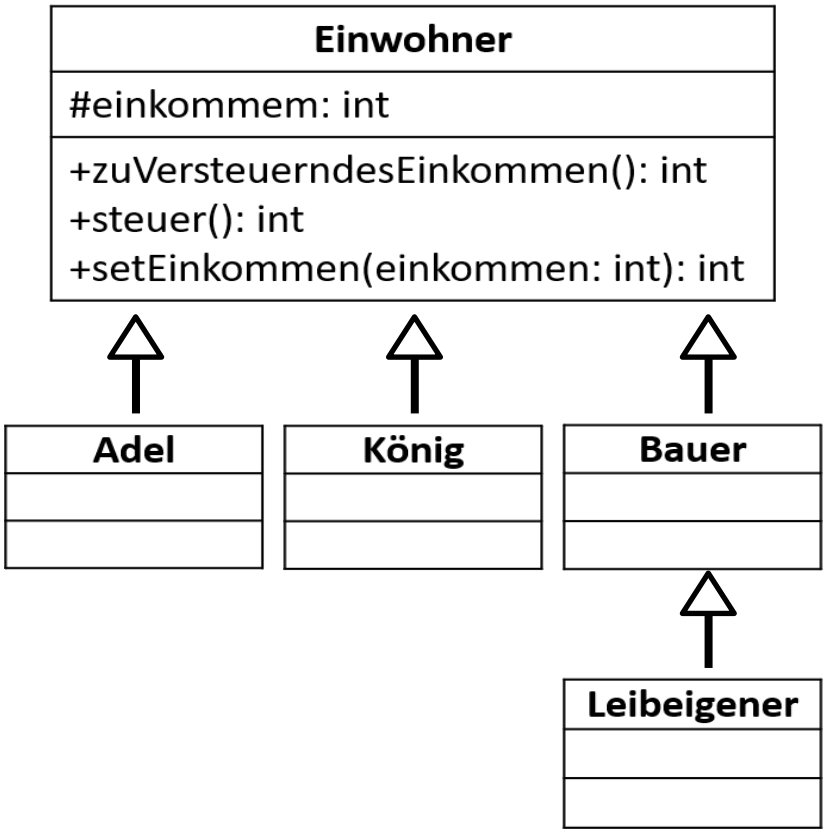
1. Können Objekte einer Klasse UnterKlasse auf Attribute zugreifen, die in der Oberklasse OberKlasse von UnterKlasse als private deklariert sind?
2. Wird in Java ein Konstruktor der Form Konto(String inhaber) vererbt?
3. Es ist manchmal notwendig, in einem Konstruktor explizit einen Konstruktor der direkten Oberklasse aufzurufen. Wie müssen Sie in Java einen solchen Aufruf formulieren?
4. Warum darf man einer Variablen vom Typ OberKlasse ein Objekt vom Typ UnterKlasse zuweisen?
5. Überlegen Sie sich, welche Ausgabe die main-Methode von UnterKlasse erzeugen wird. Notieren Sie Ihre Vermutung, rufen Sie dann die Methode auf. Stimmt das Ergebnis mit Ihrer Vermutung überein? Versuchen Sie, das Ergebnis zu erklären.

5 Aufgabe

In einem mittelalterlichen Königreich soll das Finanz- und Steuerwesen auf EDV umgestellt werden. Die verschiedenen Bevölkerungsgruppen werden durch die folgende Klassenhierarchie modelliert:

#: protected

+: public



Das Attribut *einkommen* gibt das tatsächliche Jahreseinkommen des Einwohners in Talern an. Die Methoden *zuVersteuerndesEinkommen()* und *steuer()* sollen die für jeden Einwohner des Königreiches korrekte Werte gemäß den folgenden königlichen Vorschriften liefern:

1. Sofern dieses Gesetz nichts Gegenteiliges aussagt, hat jeder Einwohner sein gesamtes Jahreseinkommen zu versteuern.

2. Jeder Einwohner hat 10% seines zu versteuernden Einkommens als Steuer zu entrichten. Der Steuerbetrag wird auf ganze Taler abgerundet, jedoch beträgt die Steuer immer mindestens 1 Taler.
3. Der König zahlt auch für sein steuerpflichtiges Einkommen keine Steuern.
4. Für Angehörige des Adels beträgt die Steuer mindestens 20 Taler.
5. Bei Leibeigenen sind 12 Taler des Jahreseinkommens steuerfrei.

Da jährliche Änderungen bei der Steuerberechnung zu erwarten sind, darf die Grundregel (2.) änderungsfreundlich nur an einer Stelle der Klassenhierarchie implementiert werden.

- Implementieren Sie die Klassenhierarchie! Überlegen Sie sich zunächst, wie die Methoden in der Klasse Einwohner implementiert werden müssen. Welche Methoden müssen in den Unterklassen überschrieben werden?
- Überlegen Sie sich, welche Ausgabe main() der Klasse Koenigreich (Koenigreich.java) erzeugt! Testen Sie.

6 Aufgabe

In dieser Übung sollen die beiden Klassen `Kreis` und `Quadrat` implementiert werden. Hierzu leiten beide Klassen von der abstrakten Basisklasse `GeometrischeFigur` ab und werden mit Hilfe der Klasse `TestBerechnung` getestet. Die beiden Klassen haben die Aufgabe, die Fläche und den Umfang eines Kreises bzw. Quadrats zu berechnen.

	Umfang	Fläche
Kreis	$2 * \pi * r$	π * r^2
Quadrat	4 * a	a^2

Die Kreiszahl π ist in der Klasse `Math` definiert.

```
// Datei: GeometrischeFigur.java
public abstract class GeometrischeFigur
{
	protected abstract double berechneFlaeche();
	protected abstract double berechneUmfang();

	public void print(){
		System.out.println ("Die Fläche beträgt: " +
				    berechneFlaeche());
		System.out.println ("Der Umfang beträgt: " +
				    berechneUmfang());
		System.out.println();
	}
}

// Datei: TestBerechnung.java
public class TestBerechnung
{
	public static void main (String [] args)
	{
		Kreis kreisRef = new Kreis (5);
		Quadrat quadratRef = new Quadrat (10);
		kreisRef.print();
		quadratRef.print();
	}

}
```

7 Aufgabe

Die Klassen `Pkw` und `Motorrad` sollen von der Klasse `Fahrzeug` abgeleitet werden. In der Klasse `FahrzeugTest` sollen die Klassen `Pkw` und `Motorrad` getestet werden. Das folgende Java-Programm enthält die Klassen `Fahrzeug`, `Pkw`, `Motorrad` und `FahrzeugTest`. Die fehlenden und zu ergänzenden Teile des Programms sind durch „//>. . . . .“ gekennzeichnet. Lesen Sie zuerst die Fragen nach dem Programm, bevor Sie das Programm vervollständigen!

```
import java.util.*;
import javapack.Einlesen;

public class FahrzeugTest {
	public static void main(String args[]) {
		System.out.println("Start des Programms");
		// Anlegen eines Arrays aus 6 Fahrzeugen
		//> . . . . .
		// Die ersten 3 Elemente des Arrays sollen mit Pkws
		// gefüllt werden
		System.out.println();
		System.out.println("3 Pkws");
		//> . . . . .
		// Die drei letzten Elemente mit Motorrädern füllen
		System.out.println();
		System.out.println("3 Motorräder");
		//> . . . . .
		// Geben Sie in einer Schleife für alle Array-Elemente
		// die entsprechenden Datenfelder aus
		//> . . . . .
		// Ermittlung des Gesamtwerts aller Fahrzeuge
		double summe = 0;
		//> . . . . .
		System.out.println();
		System.out.println("Gesamtwert aller Fahrzeuge: " + summe);
	}

}
// Datei: Fahrzeug.java

class Fahrzeug {
	private float preis;
	private String herstellerName;

	public Fahrzeug() {
		System.out.println();
		System.out.print("Geben Sie den " + "Herstellernamen ein: ");
		herstellerName = Einlesen.liesString("Hesrtellername> ");
		System.out.print("Geben Sie den Preis ein: ");
		preis = Einlesen.liesFloat("Preis> ");
	}

	public void print() {
		System.out.println();
		System.out.println("Herstellername: " + herstellerName);
		System.out.println("Preis : " + preis);
	}
	// Methode getPreis();
	//> . . . . .
}
```

```
// Datei: Pkw.java
class Pkw extends Fahrzeug {
	private String fahrzeugtyp = "Pkw";
	private String modellBezeichnung;

	public Pkw() {
		//> . . . . .// Aufruf des Konstruktors
		// der Basisklasse
		System.out.print("Geben Sie die " + "Modellbezeichnung ein: ");
		modellBezeichnung = Einlesen.liesString("Modellbezeichnung> ");
	}

	public void print() {
		//> . . . . . .
	}
}

// Datei: Motorrad.java
class Motorrad extends Fahrzeug {
	private String fahrzeugtyp = "Motorrad";

	public void print() {
		//> . . . . . .
	}
}
```

7.1.1 Fragen/Aufgaben

a) Schreiben Sie die Methode `getPreis()` der Klasse `Fahrzeug`.
b) Vervollständigen Sie den Konstruktor der Klasse `Pkw`.
c) Überschreiben Sie in der Klasse `Pkw` die Methode `print()` der Klasse `Fahrzeug`. Die Methode `print()` der Klasse `Pkw` soll alle Datenfelder eines Objektes der Klasse `Pkw` unter Zuhilfenahme der Methode `print()` der Basisklasse ausgeben. Ergänzen Sie die Methode `print()` der Klasse `Pkw`. Ergänzen Sie in analoger Weise die Methode `print()` der Klasse `Motorrad`.
d) Ergänzen Sie die fehlenden Teile der Klasse `FahrzeugTest`.

Fragen zum Wiederholen/ Selbststudium/ Ausprobieren

Vererbung

1. Frage

Welche der untenstehenden Bemerkungen trifft zu?

a) Die Beziehung zwischen einer Superklasse und einer Subklasse nennt sich has-a-Relationship.
b) Dies ist eine korrekte Formulierung für eine Vererbungsbeziehung zwischen Klassen: `public class Kaktus implements Pflanze { }.`
c) Eine Superklasse erbt Methoden von der Subklasse.
d) Die Beziehung zwischen einer Superklasse und einer Subklasse nennt sich is-a-Relationship.
e) Eine Subklasse erbt Methoden von der Superklasse.
f) Dies ist eine korrekte Formulierung für eine Vererbungsbeziehung zwischen Klassen: `public class Kaktus extends Pflanze { }.`
g) In Java gibt es keine Vererbungsbeziehungen.
h) Besteht z. B. die Klasse Tisch aus Tischbeinen und einer Tischplatte, so handelt es sich um eine has-a-Relationship.