Übung 10, Lösungsvorschlag

TECHNISCHE UNIVERSITÄT DARMSTADT

Aufgabe 1 (Wissensfrage)

a) Worin unterscheiden sich konkrete und abstrakte Klassen?

b) Wie kann in Java eine Art Mehrfachvererbung umgesetzt werden, obwohl **eine** Sub-Klasse immer nur von **einer** Ober-Klasse erben kann?

Lösung Aufgabe 1

a) Von abstrakten Klassen können, im Gegensatz zu konkreten Klassen, keine Objekte erzeugt werden. (Es können aber Objekte von erbenden konkreten Klassen erzeugt werden.) Außerdem deklarieren abstrakte Klassen oft abstrakte Methoden, bei denen keine Implementierung, sondern nur der Methodenkopf festgelegt ist. Die Implementierung erfolgt erst in erbendenden konkreten Klasse einer abstrakten Klasse. Bei konkreten Klassen ist eine Methode ohne Implementierung nicht möglich.

b) Interfaces erlauben eine Art Mehrfachvererbung. In Verbindung mit Polymorphismus können Klassen auf Interfaces arbeiten und nicht mit dem konkreten Datentyp der Klassen. Da eine Klasse beliebig viele Interfaces implementieren kann, wird damit so etwas wie Mehrfachvererbung möglich

Aufgabe 2 (LiveTicker)

a) Schreiben Sie ein Interface `LiveTicker`. Das Interface soll eine öffentliche Methode `stelleDar()` definieren. Die Methode verfügt über keine Parameter und gibt auch keinen Wert zurück.

a) Schreiben Sie nun eine Klasse `FussballTicker`, die das Interface `LiveTicker` implementiert. Die Klasse `FussballTicker` soll über die beiden privaten Attribute `heimPunkte` und `gastPunkte` mit ganzzahligem Wertebereich verfügen.
Weiterhin soll die Klasse über eine öffentliche Methode `aktualisiere(…)` verfügen, die zwei ganzzahlige Werte übergeben bekommt und diese den beiden Attributen zuweist. Die Methode `aktualisiere(…)` gibt keinen Wert zurück.
Die Methode `stelleDar()` aus dem Interface `LiveTicker` soll in der Klasse `FussballTicker` so realisiert werden, dass eine Bildschirmausgabe des Spielstands erfolgt. Dabei soll zunächst der Text "Spielstand: ", dann der Punktestand der Heimmannschaft und schließlich der Punktestand der Gäste (durch einen Doppelpunkt abgetrennt) ausgegeben werden.

Lösung Aufgabe 2

a)

```
1  package gdp.uebung10.aufgabe2;
2
3  public interface LiveTicker {
4        void stelleDar();
5  }
```

b)

```
1  package gdp.uebung10.aufgabe2;
2
3  public class FussballTicker implements LiveTicker {
4
5        private int heimPunkte;
6        private int gastPunkte;
7
8        public void aktualisiere(int heimPunkte, int gastPunkte) {
9              this.heimPunkte = heimPunkte;
10             this.gastPunkte = gastPunkte;
11       }
12
13       public void stelleDar() {
14             System.out.println("Spielstand: " + heimPunkte + " : " +
   gastPunkte);
15       }
16 }
```

Aufgabe 3 (Programmieren – Verwaltung von Sparkonten)

Wie das Leben so spielt, erhalten Sie schon wieder eine neue Anforderung für das Programm zur Verwaltung von Bankkonten und müssen dies nun abändern.

Die Anforderung lautet, dass eine weitere Klasse `Tagesgeldkonto` implementiert werden soll. Tagesgeldkonten und Sparkonten haben gemeinsam, dass ihr Guthaben in regelmäßigen Abständen verzinst wird. Sparkonten werden mit 0,4% und Tagesgeldkonten mit 0,1% verzinst. Genau wie Sparkonten soll also auch die Klasse `Tagesgeldkonto` über ein entsprechendes statisches Attribut für den Zinssatz verfügen.

Im Sinne der Objektorientierung gibt es nun zwei Lösungsansätze: Eine gemeinsame abstrakte Oberklasse `VerzinsbaresKonto` für beide Klassen erstellen oder ein gemeinsames Interface `Verzinsbar`.

Erörtern Sie zuerst die Vor- und Nachteile für beide Lösungsansätze.

a) Für den Lösungsansatz mit einer abstrakten Klasse `VerzinsbaresKonto` soll die abstrakte Klasse von `Konto` erben, sodass eine Vererbungshierarchie entsteht. Setzen Sie diese Vererbungshierarchie für die Klassen `Tagesgeldkonto` und `Sparkonto` fort, sodass sie als konkrete Klassen von `VerzinsbaresKonto` erben.

b) Für den zweiten Lösungsansatz soll das Interface `Verzinsbar` ausschließlich die Methode `verzinsen()` deklarieren. Somit erben die Klassen `TagesgeldKonto` sowie `Sparkonto` Sub-Klassen von `Konto`, implementieren hingegen zusätzlich das neu geschriebene Interface.

Erstellen Sie **für beide Lösungen** jeweils ein Hauptprogramm (main-Methode) in der Klasse `Bank`, legen die folgenden Konten in einem Konto[] Array an. Führen Sie die folgenden Buchungen aus und lassen sich die Kontostände nach jeder Buchung ausgeben. Schreiben Sie außerdem eine weitere, statische Methode, für welche ein Array an verzinsbaren Konten übergeben wird. Nutzen Sie hierzu ein weiteres Array vom Datentyp VerzinsbaresKonto[] bzw. Verzinsbar[], und machen Sie vom **Polymorphismus** gebrauch. Verzinsen Sie anschließend alle verzinsbaren Konten innerhalb des Hauptprogramms. Die Signaturen der Methoden sehen wie folgt aus:

a) Lösung mit abstrakter Klasse: `verzinseKonten(VerzinsbaresKonto[] konten)`

b) Lösung mit Interface: `verzinseKonten(Verzinsbar[] konten)`

Konto	Einzahlung	Auszahlung
giro1 = new Girokonto (1001, 300)		
giro1	1000	
giro1		1200
tag1 = new Tagesgeldkonto(1002)		
tag1	2000	
spar1 = new Sparkonto(1003)		
spar1	5000	
alle verzinsbaren Konten		verzinsen()

Lösung Aufgabe 3

Beide Lösungen haben ihre Vor- und Nachteile. Eine abstrakte Klasse ermöglicht z.B. die Implementierung neuer Methoden in der abstrakten Klasse, die dann vererbt werden. Interfaces können hingegen Methoden nur deklarieren, die dann in allen implementierenden Klassen neu implementiert werden müssen. Hingegen erfordert die abstrakte Klasse einen Eingriff in die Klassenhierarchie, so dass es schwieriger wird, den Code aufgrund der Vererbungsstufen nachzuvollziehen. Hier bieten Interfaces eine „minimalinvasive“ Lösung.

a) Lösung mit abstrakter Klasse:

```
public class Konto {
	public int kontonummer;
	public double kontostand;

	public Konto(int nr) {
		kontonummer = nr;
		kontostand = 0.0;
	}

	public boolean einzahlen(double betrag) {
		if (betrag > 0) {
			kontostand = kontostand + betrag;
			return true;
		} else {
			return false;
		}
	}

	public boolean auszahlen(double betrag) {
		if (betrag > 0 && kontostand >= betrag) {
			kontostand = kontostand - betrag;
			return true;
		} else {
			return false;
		}
	}

	public void druckeKontoauszug() {
		System.out.println("Konto " + kontonummer + ": " + kontostand);
	}
}
```

```java
public class Girokonto extends Konto {
	public double dispoLimit;

	public Girokonto(int nr, double dispo) {
		super(nr);
		dispoLimit = dispo;
	}

	public boolean auszahlen(double betrag) {
		if (betrag > 0 && (kontostand + dispoLimit) >= betrag) {
			kontostand = kontostand - betrag;
			return true;
		} else {
			return false;
		}
	}
}
```

```java
public abstract class VerzinsbaresKonto extends Konto {

	public VerzinsbaresKonto(int nr) {
		super(nr);
	}

	public abstract void verzinsen();
}
```

```java
public class Sparkonto extends VerzinsbaresKonto {
	static double zinssatz;

	public Sparkonto(int nr) {
		super(nr);
	}

	public void verzinsen() {
		double zinsen = kontostand * zinssatz;
		einzahlen(zinsen);
	}

	public static void jahresabschluss(Sparkonto[] konten) {
		for (int i = 0; i < konten.length; i++) {
			konten[i].verzinsen();
		}
	}
}
```

```java
public class Tagesgeldkonto extends VerzinsbaresKonto {
	static double zinssatz;

	public Tagesgeldkonto(int nr) {
		super(nr);
	}

	public void verzinsen() {
		double zinsen = kontostand * zinssatz;
		einzahlen(zinsen);
	}
}
```

```java
public class Bank {
	public static void main(String[] args) {
		Konto[] konten = new Konto[3];

		Tagesgeldkonto.zinssatz = 0.001; // 0.1%
		Sparkonto.zinssatz = 0.004; // 0.4%

		konten[0] = new Girokonto(1001, 300);
		konten[1] = new Tagesgeldkonto(1002);
		konten[2] = new Sparkonto(1003);

		// Girokonto
		konten[0].einzahlen(1000.0);
		konten[0].druckeKontoauszug();
		konten[0].auszahlen(1200.0);
		konten[0].druckeKontoauszug();

		// Tagesgeldkonto
		konten[1].einzahlen(2000.0);
		konten[1].druckeKontoauszug();

		// Sparkonto
		konten[2].einzahlen(5000.0);
		konten[2].druckeKontoauszug();

		if (konten[1] instanceof VerzinsbaresKonto && konten[2] instanceof
	VerzinsbaresKonto) {
			VerzinsbaresKonto[] vk = { (VerzinsbaresKonto) konten[1],
			(VerzinsbaresKonto) konten[2] };
			verzinseKonten(vk);
		}
		/* Beachten Sie, dass VerzinsbaresKonto[] vk By-Ref an die Methode
		* verzinseKonten() übergeben wird.
		* Nach dem Aufruf von verzinseKonten(vk) ändern sich die Kontostände
		* innerhalb der Konto-Objekte */
		konten[1].druckeKontoauszug();
		konten[2].druckeKontoauszug();
	}

	public static void verzinseKonten(VerzinsbaresKonto[] konten) {
		for (int i = 0; i < konten.length; i++) {
			konten[i].verzinsen();
		}
	}
}
```

b) Lösung mit Interface:

```
public class Konto {
	public int kontonummer;
	public double kontostand;

	public Konto(int nr) {
		kontonummer = nr;
		kontostand = 0.0;
	}

	public boolean einzahlen(double betrag) {
		if (betrag > 0) {
			kontostand = kontostand + betrag;
			return true;
		} else {
			return false;
		}
	}

	public boolean auszahlen(double betrag) {
		if (betrag > 0 && kontostand >= betrag) {
			kontostand = kontostand - betrag;
			return true;
		} else {
			return false;
		}
	}

	public void druckeKontoauszug() {
		System.out.println("Konto " + kontonummer + ": " + kontostand);
	}
}
```

```
public class Girokonto extends Konto {
	public double dispoLimit;

	public Girokonto(int nr, double dispo) {
		super(nr);
		dispoLimit = dispo;
	}

	public boolean auszahlen(double betrag) {
		if (betrag > 0 && (kontostand + dispoLimit) >= betrag) {
			kontostand = kontostand - betrag;
			return true;
		} else {
			return false;
		}
	}
}
```

```
public interface Verzinsbar {
	public void verzinsen();
}
```

```java
public class Sparkonto extends Konto implements Verzinsbar {
	static double zinssatz;

	public Sparkonto(int nr) {
		super(nr);
	}

	public void verzinsen() {
		double zinsen = kontostand * zinssatz;
		einzahlen(zinsen);
	}

	public static void jahresabschluss(Sparkonto[] konten) {
		for (int i = 0; i < konten.length; i++) {
			konten[i].verzinsen();
		}
	}
}
```

```java
public class Tagesgeldkonto extends Konto implements Verzinsbar {
	static double zinssatz;

	public Tagesgeldkonto(int nr) {
		super(nr);
	}

	public void verzinsen() {
		double zinsen = kontostand * zinssatz;
		einzahlen(zinsen);
	}
}
```

```java
public class Bank {
	public static void main(String[] args) {
		Konto[] konten = new Konto[3];

		Tagesgeldkonto.zinssatz = 0.001; // 0.1%
		Sparkonto.zinssatz = 0.004; // 0.4%

		konten[0] = new Girokonto(1001, 300);
		konten[1] = new Tagesgeldkonto(1002);
		konten[2] = new Sparkonto(1003);

		// Girokonto
		konten[0].einzahlen(1000.0);
		konten[0].druckeKontoauszug();
		konten[0].auszahlen(1200.0);
		konten[0].druckeKontoauszug();

		// Tagesgeldkonto
		konten[1].einzahlen(2000.0);
		konten[1].druckeKontoauszug();

		// Sparkonto
		konten[2].einzahlen(5000.0);
		konten[2].druckeKontoauszug();

		if (konten[1] instanceof Verzinsbar && konten[2] instanceof
	Verzinsbar) {
			Verzinsbar[] v = { (Verzinsbar) konten[1], (Verzinsbar)
		konten[2] };
			verzinseKonten(v);
		}
		// Beachten Sie, dass Verzinsbar[] v By-Ref an die Methode
		// verzinseKonten() übergeben wird.
		// Nach dem Aufruf von verzinseKonten(v) ändern sich die Kontostände
		// innerhalb der Konto-Objekte
		konten[1].druckeKontoauszug();
		konten[2].druckeKontoauszug();
	}

	public static void verzinseKonten(Verzinsbar[] konten) {
		for (int i = 0; i < konten.length; i++) {
			konten[i].verzinsen();
		}
	}
}
```