Übung 10

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?

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.

b) 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.

※抽象类和接口都不能被实例化(使用 new Interface or Abstract),
但都可以创建变量. eg: Abstract a = new concrete();
Interface i = new concrete();

※做类型转换时, Abstract a = (Abstract) A1;
a可以调用实例类A1当中所有的方法与特征,
然后由于做了类型转换,抽象变量a可以放入抽象数组 Abstract[]
当中!

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)`

Interface ganz ein Attribut?

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()

Aufgabe 1:

a) Abstrakte Klassen werden oft als Ober-Klasse verwendet, wenn Sub-Klassen teilweise gleiches Verhalten aufweisen, aber unterschiedliche Implementierungen erfordern.

- Bei der abstrakte Klassen kann das Objekt nicht erzeugt werden.
- Abstrakte Klasse kann als statischer Datentyp verwendet werden.
- Es deklariert oft abstrakte Methode!

b) Interface wird für die Mehrfachvererbung implementiert. Eine Subklasse muss von einige Oberklasse erben, aber es kann von verschiedene Interface erben!