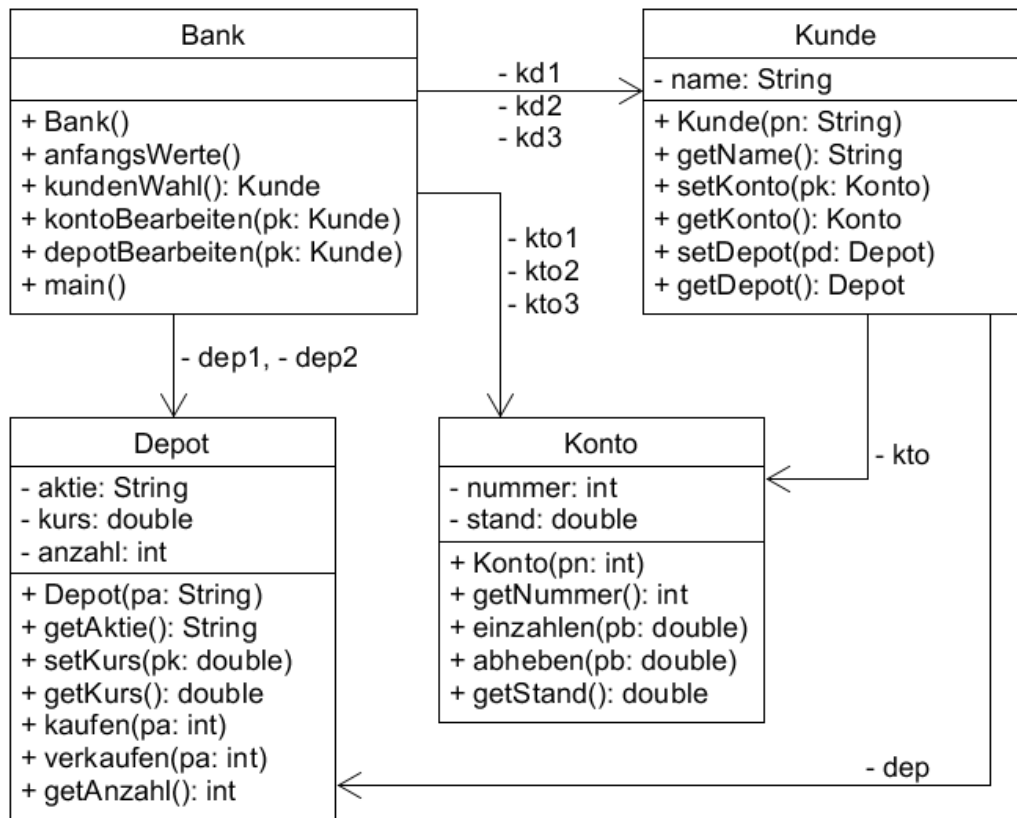


Aufgabe

Im folgenden Klassendiagramm ist die Software für eine Bank dargestellt, in dem außer Konten auch Kunden und Aktiendepots modelliert sind:



Jedes **Kunden**-Objekt verwaltet den Namen des Kunden. Kunden haben in der Regel ein Konto, und können außerdem ein Aktiendepot haben (d.h. das Depot ist optional).

In diesem einfachen Modell können Kunden maximal ein Konto und ein Aktiendepot haben.

Jedes **Konto** hat eine Kontonummer und einen Kontostand (der hier nicht negativ sein darf).

Ein **Depot** verwaltet Anteile einer Aktie (z.B. „VW“ oder „Bosch“). Die Aktie hat einen Kurs, also ein Geldbetrag, zu dem man jeweils einen Anteil der Aktie kaufen bzw. verkaufen kann.

Ein Depot-Objekt speichert die Anzahl der Anteile, die der Kunde zur Zeit besitzt.

Verwende die ausgeteilte BlueJ-Vorlage. Die Klassen Konto und Depot sind bereits implementiert.

- a) Implementiere die Klasse **Kunde** mit allen im Diagramm dargestellten Methoden:
 - i. Der Name des Kunden wird mit dem Konstruktor gesetzt.
Es gibt keine set-Methode, aber eine get-Methode zum Abfragen des Namens.
 - ii. Eine Referenz auf ein Konto bzw. ein Depot kann man einem Kunden-Objekt mithilfe entsprechender set-Methoden zuweisen. Für die Referenzen auf ein Konto bzw. ein Depot gibt es auch je eine get-Methode. Da nicht alle Kunden Depots haben, gibt es weiterhin eine Methode hatDepot(), die true zurückgibt, falls der Kunde ein Depot hat, und sonst false.
(Frage: woran kann die Methode das erkennen?)
- b) Implementiere die Klasse **Bank**:
Gib die Deklarationen der Referenzen auf Kunden, Konten und Depots an und implementiere den Konstruktor, der die entsprechenden Objekte erzeugt. Für die benötigten Daten (Namen, Kontonummern und Aktiennamen) setze Beispielwerte.
- c) Implementiere die Methode **anfangsWerte()**. Diese ordnet jedem Kunden ein Konto, und zwei von ihnen und ein Depot zu. Außerdem setzt sie Kurse für die Aktien und zahlt auf die Konten einen Anfangsbetrag ein.
- d) Implementiere die Methode **kontoBearbeiten()**.
Der Methode wird eine Referenz auf ein Kunden-Objekt als Parameter übergeben.
Sie gibt zunächst den Namen des Kunden und seine Kontodaten (Nummer und Stand) aus.
Der Kunde soll die Wahl zwischen Ein- und Auszahlung haben und kann dann entsprechend einen Betrag eingeben, der auf das Konto eingezahlt bzw. davon abgehoben wird.
Zum Schluss wird der neue Kontostand ausgegeben.
Die Methode soll die möglichen Sonderfälle sinnvoll behandeln.

```

BlueJ: Konsole - Aufg.2
Optionen
Kunde: Dennis
Kontonummer: 1001 Kontostand: 500.0
Möchten Sie einzahlen (1) oder abheben (2)? 1
Welchen Betrag einzahlen? 1000
Neuer Kontostand: 1500.0
    
```

e) Implementiere die Methode **depotBearbeiten()**.

Der Methode wird eine Referenz auf ein Kunden-Objekt als Parameter übergeben.
 Sie gibt zunächst den Kundennamen sowie dessen Konto- und Depotdaten aus:
 Kontonummer und -stand, Name der Aktie, Kurs und Anzahl der bereits gekauften Anteile.
 Der Kunde hat die Wahl, Aktien zu kaufen oder zu verkaufen. Entsprechend erhöht bzw.
 verringert sich die Anzahl der Anteile. Außerdem wird beim Kauf der entsprechende Geldbetrag
 vom Konto abgeboben, beim Verkauf eingezahlt.
 Wie in Aufgabe d) muss die Methode die möglichen Sonderfälle behandeln.

```
BlueJ: Konsole - Aufg.2
Optionen
Kunde: Dennis
Kontonummer: 1001 Kontostand: 1500.0
Aktie: Siemens Anzahl: 0 Kurs: 100.0
Möchten Sie kaufen (1) oder verkaufen (2)? 1
Wie viele Anteile? 7
Anteile gekauft, 700.0 abgeboben.
```

f) Implementiere die Methode **kundenWahl()**.

Sie liefert eine Referenz auf ein Kunden-Objekt als Rückgabewert.
 Der Benutzer soll einen Namen eintippen. Die Methode vergleicht das mit den Namen der drei
 Kunden-Objekte. Bei einer Übereinstimmung wird das entsprechende Objekt zurückgegeben,
 sonst muss der Benutzer einen neuen Namen eintippen.

g) Implementiere die **main-Methode**.

Sie ruft zuerst die Methode `anfangsWerte()` auf.
 Dann wiederholt sie (bis der Benutzer das Programm abbricht):
 Die Methode `kundenWahl()` liefert eine Referenz auf ein Kunden-Objekt.
 Falls der Kunde ein Aktien-Depot hat, kann er wählen, ob er sein Konto oder sein Depot
 bearbeiten möchte – ansonsten folgt automatisch die Bearbeitung des Kontos.
 Mit den Methoden `kontoBearbeiten()` bzw. `depotBearbeiten()` wird die gewählte Aktion
 ausgeführt.
 Dann wird der Benutzer gefragt, ob ein weiterer Kunde bearbeitet werden soll, und die Schleife
 geht in die nächste Wiederholung oder bricht ab.