zu Kapitel 7

Aufgabe 1
(Kontofuehrung.java)

Schreiben Sie eine Klasse Konto, welche folgende Eigenschaften hat: einen Kontoinhaber (nur den Namen als String), einen Kontostand (als float) und eine Kontonummer (der Einfachheit halber ein long). Ein Konto wird unter Angabe des Namens des Kontoinhabers, einer Kontonummer und mit dem Kontostand = 0,0 Euro angelegt. Man kann Geldbeträge einzahlen und abheben. Diese verändern logischerweise den Kontostand. Ein Kontoauszug kann gedruckt werden, das soll bedeuten, dass Kontoinhaber, -nummer und der Kontostand am Bildschirm ausgegeben werden.

Wenn die Klasse Konto kompilierbar ist, schreiben Sie eine Klasse Kontofuehrung. In deren main()-Funktion legen Sie zwei Konten (von Herrn Maier und Frau Mueller) an, initialisieren diese und tätigen dann ein paar Einzahlung und Abhebungen. Mittels Anschauen des Kontoauszugs können Sie die Geldbewegungen kontrollieren.

Aufgabe 2
(RaumBelegung.java)

Programmieren Sie für ein ganz einfaches Raumbelegungstool eine Klasse Raum. Das Raumbelegungstool soll (der Einfachheit halber nur für einen Tag) einen Raum stundenweise als besetzt oder frei verwalten. Sie können sich die Raumbelegung anzeigen lassen, den Raum für eine Stunde reservieren oder eine Reservierung löschen. Damit Sie nicht komplett "daneben liegen", ist unten die Skizze eine Klassendeklaration der Klasse Raum gegeben. Trotzdem ist es sinnvoll, erst einmal selbst zu überlegen, wie so eine Klasse aussehen könnte.

Wenn die Klasse Raum eigenständig kompilierbar ist, dann schreiben Sie ein main-Programm, welches in Art eines Raumbelegungstool den Benutzer wahlweise einen Raum reservieren oder eine Reservierung löschen lässt. Der Belegungsplan kann auf Wunsch auch komplett angezeigt werden.

Tipp: So könnte eine Klasse Raum aussehen:

```
class Raum
{
      // Attribute:
      private boolean[] stunde;  // ein Array fuer 24 Stunden, gibt an
                                 // ob der Raum frei oder besetzt ist
      // Methoden:
      public void init () {...}
      public void belegen(int zeit) {...}
      public void loeschen(int zeit) {...}
      public void belegungsplanDrucken () {...}
}
```

Aufgabe 3
(ZeitTest.java)

Erstellen Sie für eine Digitaluhr eine Klasse Zeit, welche Uhrzeiten in der Form ss:mm repräsentiert. Gültige Zeiten liegen zwischen 00:00 Uhr und 23:59 Uhr. Die Klasse Zeit soll eine Funktion print() besitzen, welche eine Zeit beispielsweise in der Form 07:05 ausgibt. (Und zwar genau so, also mit führender Null.) Weiterhin besitzt die Klasse Zeit einen Konstruktor, mit dem eine Initialisierung einer Zeit unter Angabe der Stunden und Minuten möglich ist. Wenn beim Anlegen eines Zeit-Objektes keine Zeit angegeben wird, soll die Zeit automatisch mit 00:00 initialisiert werden. Wird beim Initialisieren eine ungültige Uhrzeit angegeben (also z.B 24:30 Uhr oder 3:67 Uhr) soll der Konstruktor die Zeit auf 00:00 setzten und eine entsprechende Fehlermeldung erzeugen. Testen nicht vergessen!

Aufgabe 4
(RechteckListe.java)

a) Programmieren Sie eine Klasse Rechteck, die ein einfaches Rechteck mit einer ganzzahligen Länge und einer ganzzahligen Breite beschreibt. Die Klasse bietet nach außen folgende Schnittstellen an:
   - einen Konstruktor, der Länge und Breite übergeben bekommt
   - flaeche() gibt die Grundfläche des Rechtecks an
   - istQuadrat() überprüft, ob das Rechteck ein Quadrat ist.

b) Schreiben Sie ein Programm RechteckListe, das 1000 Rechteck-Objekte mit zufälligen Kantenlängen im Bereich 1-10 erzeugt und diese in einem Array speichert. Aus diesen Rechtecken errechnen Sie die durchschnittliche Fläche eines Rechtecks und bestimmen, wie viele von den Rechtecken Quadrate sind.

Tipp: Zufallszahlen erzeugen Sie mittels der statischen Methode random() in der Klasse Math. Lesen Sie dazu die Beschreibung z.B. in der Java-SUN-Dokumentation durch.

Aufgabe 5
(Bruch.java)

Brüche (z.B. 1/3) werden in Java meist mittels float-Variablen dargestellt. Das kann aber in mathematischen Anwendungen zu Rundungsfehlern führen, welche man vermeiden kann, wenn man eine eigene Klasse Bruch verwendet. Die Klasse Bruch hält intern den Zähler und den Nenner eines Bruchs als Integer-Zahlen. Ein Bruch wird unter Angabe seines Zählers und seines Nenners erzeugt. Für einen Bruch gilt, dass er Zähler und Nenner immer in gekürzter Form speichert und ein negatives Vorzeichen immer im Zähler hält. (Bei Eingabe von 5 und -10 wird also -1 und 2 gespeichert. Die Klasse Bruch bietet die vier Grundrechenarten als statische Methoden und eine Methode toString(), welche einen Bruch in der Form "<z>/<n>" zurückgibt.

Aufgabe 6
(AmpelSchaltung.java)

Programmieren Sie eine einfache Ampelschaltung. Hierzu soll eine Klasse Ampel die Konstanten ROT, ROT-GELB, GRUEN und GELB besitzen, sowie ein Element "farbe", welches den aktuellen Zustand der Ampel enthält. Die Klasse Ampel besitzt folgende Methoden:
- **`String getFarbe();`**
  gibt den aktuellen Zustand der Ampel als String zurück;
- **`void schalte();`**
  setzt den Ampelzustand jeweils auf den nächsten sinnvollen Zustand, also von ROT nach ROT-GELB nach GRUEN usw.

Im Mainprogramm legen Sie eine Ampel an und schalten diese in einer Endlosschleife jeweils auf Druck der Returntaste. In Ermangelung einer graphischen Oberfläche geben Sie jeweils die Farbe der Ampel (also ROT, ROT-GELB etc.) als String aus.

Tipp:
Um die Ampel auf Tastendruck zu schalten, sollten Sie mittels der Methode nextLine() der Klasse Scanner einfach einen leeren String von der Tastatur abholen.