Programmierstarthilfe WS 2008/09
Fakultät für Ingenieurwissenschaften und Informatik

9. Blatt
Für die Woche vom 15.12. bis zum 19.12.2008 (KW 51)

Organisatorisches

Die Webseiten zur Veranstaltung sind unter `http://www.uni-ulm.de/in/mi/lehre/2008ws/programmierstarthilfe.html` zu finden.

Nachdem wir letzte Woche bereits mit Klassen und Objekten gearbeitet haben, stehen diese Woche statische Attribute und `public` und `private` im Vordergrund.

1 Aufgaben

1.1 Kontoverwaltung

Zur Verwaltung bei einer Bank sollen Konto-Objekte erzeugt werden.

a) Überlege dir, welche Eigenschaften ein Objekt vom Typ `Konto` enthalten muss und welche davon beim Anlegen eines neuen Kontos bekannt sein müssen. Schreibe einen Konstruktor, der diese Werte entgegennimmt und alle anderen Werte sinnvoll belegt.

b) Erweitere deine Klasse `Konto`. Überlege dir, auf welche Werte eines Konto-Objekts man von außen nicht zugreifen können sollte, und setze die Sichtbarkeiten dementsprechend. Stelle anschließend die Methoden bereit, die man nun zum Benutzen des Kontos braucht.

c) Die Bankleitzahl soll für alle Konten identisch sein, aber dennoch Teil des Kontos. Wenn die Bankleitzahl von außen geändert wird, soll das in allen Konten geschehen. Wie realisiert man das? Setze es um, und teste.

d) Erweitere deine Klasse `Konto` um ein Attribut `pin`.

   i) Der PIN soll bei der Erstellung eines Objektes im Konstruktor übergeben werden und sich während der Lebensdauer des Objektes nicht mehr ändern. Welche Sichtbarkeit sollte für `pin` gewählt werden?

   ii) Ändere die Klasse so ab, dass Transaktionen wie Abheben nur noch bei Übergabe des korrekten PINs funktioniert. Wurde ein falscher PIN übergeben, sollen die entsprechenden Methoden `false` zurückgeben.

*e) Implementiere eine zusätzliche Methode zum Überweisen. Überlege dir was nötig ist, damit ein Konto-Objekt den Betrag eines anderen Konto-Objekts verändern kann.

1.2 Website-Rating

In einer Klasse `Website` sollen die Adresse einer Website und eine zugehörige Bewertung zwischen null und fünf Sternen gespeichert werden können. Schreibe die Klasse und benutze `public` und `private` um den Zugriff auf die Klassen-Attribute sinnvoll zu steuern. Nach außen hin sollen nur drei Methoden sichtbar sein: Eine zum Abrufen der aktuellen Bewertung, eine, welche die Stimmenanzahl zurückgibt und eine zum Hinzufügen einer neuen Bewertung. Verhindere, dass von außen direkt an den Werten manipuliert werden kann, oder dass fehlerhafte Werte angegeben werden (z.B. 10 Sterne oder -5 Sterne).

2 Bonusaufgaben

Durch die Aufgaben oben hast du ein Verständnis für das neue Konzept dieses Blatts bekommen. Durch die Bonusaufgaben kannst du nun deine Kenntnisse vertiefen.

2.1 Filmobjekte

Du willst Objekte vom Typ `Film` erzeugen. Zu jedem Film sollen Titel, Erscheinungsjahr, Regisseur und Genre gespeichert werden können. Schreibe zu dieser Klasse einzelne Konstruktoren und erzeuge aus einer Testklasse Objekte:

- Einen Konstruktor, der einen Film nur mit Titelangabe anlegt, das Erscheinungsjahr wird in diesem Fall auf -1 gesetzt, Regisseur und Genre erhalten den Wert “unbekannt”
- Einen Konstruktor, der Titel, Erscheinungsjahr und Genre enthält, Regisseur wird auf “unbekannt” gesetzt.
- Einen Konstruktor, bei dem alle Werte übergeben werden.

Warum ist es nicht möglich, anschließend noch einen Konstruktor zu schreiben, dem man Titel, Erscheinungsjahr und Regisseur übergeben kann?

2.2 Geldangelegenheiten

Für meinen verdienten Urlaub möchte ich mit verschiedenen Währungen umgehen können. Da mich spätestens bei der Umrechnung in Yen das Kopfrechnen verlässt, hätte ich gerne eine Klasse `Geld`, deren Objekte Ausgaben verschiedener Geldbeträge und Währungen darstellen. Die Anwendung sieht dann etwa so aus:

```
1 Geld hotelkosten = new Geld(); hotelkosten.set(823.45, "USD");
2 Geld mietwagen = new Geld(); mietwagen.set(567.30, "GBP");
3 System.out.println("Mietwagen:   "+mietwagen
4     +", das sind "+mietwagen.toEur()+" EUR");
5 System.out.println("Hotelkosten: "+hotelkosten
6     +", das sind "+hotelkosten.toEur()+" EUR");
7 if (mietwagen.teurerAls(hotelkosten))
```

```
8       System.out.println("Der Mietwagen ist zu teuer.");
9   System.out.println("Gesammtkosten: "
10      +mietwagen.add(hotelkosten));
```

Schreibe die Klasse `Geld`, die mit diesen Methodenaufrufen sinnvoll umgehen kann.

Hinweis: Implementieren die Methode `String toString()` in der Klasse um eine Ausgabe zu machen.

2.3 Vektoren

Die Klasse `Vektor` soll einen mathematischen Vektor darstellen, der aus drei Komponenten besteht, die jeweils als `float`-Wert gespeichert werden sollen.

a) Implementiere die Klasse `Vektor`.

b) Füge der Klasse `Vektor` einen Konstruktor `public Vektor()` hinzu, den man benutzen kann, um einen Nullvektor zu erstellen.

c) Füge der Klasse `Vektor` einen Konstruktor `public Vektor(float a, float b, float c)` hinzu, den man benutzt um einen Vektor (a, b, c)zu erstellen.

d) Erstelle für die Klasse `Vektor` eine Methode `ausgabe()`, die die drei Werte der Komponenten des Vektors ausgibt (oder überschreibe die `toString()`-Methode).

e) Implementiere die Methode `multipliziere(float skalar)` für die Klasse, die komponentenweise die Werte des Vektors mit dem übergebenen `float`-Wert multipliziert.

f) Implementiere die Methode `addiere(Vektor vektor2)` für die Klasse `Vektor`, die komponentenweise die Werte des übergebenen zweiten Vektors zu den Werten des Vektors hinzuaddiert.

g) Implementiere die Methode `betrag()` für die Klasse `Vektor`, die den Betrag des Vektors ausgeben soll. Falls dir nicht bekannt ist, wie der Betrag gebildet wird, suche im Internet nach der passenden Formel hierfür.

h) Füge der Klasse `Vektor` einen Konstruktor `public Vektor(float laenge)` hinzu, den man benutzt um einen Vektor der Länge `laenge` mit drei gleichen Komponenten zu erstellen. (Hinweis: für einen Vektor (a, a, a) ist die Länge $laenge = a * \sqrt{3}$.)

2.4 Erd-Koordinaten

Die Angabe von Positionen auf der Erdoberfläche erfolgt meistens durch zwei Erdmittelpunktswinkel, den Längengrad (oft longitude λ) und den Breitengrad (oft latitude φ). Für die computerbezogene Handhabbarkeit der Koordinaten kann es sinnvoll sein, diese Koordinaten in ein kartesisches Korrdinatensystem umzurechnen. Hierfür bietet sich ein objektorientierter Ansatz an:

a) Implementiere zunächst die Klasse `Winkel`, die einen Winkel intern als `double` im Bogenmaß speichert. Der Winkel soll über die Methoden `double getRad()` und

`setRad(double)` im Bogenmaß und über die Methoden `double getGrad()` bzw. `setGrad(double)` in Grad-Skala abgefragt bzw. gesetzt werden können. Überprüfe jeweils die Gültigkeit des Wertebereichs.

b) Schreibe jetzt unter Verwendung der Winkel-Klasse die Klasse `GeoKoordinaten`, die Längen- und Breitengrad als Winkel-Objekte speichert. Implementiere auch die Methode `double[] getKartesischeKoordinaten()`, die die Position im kartesischen Koordinatensystem als `double`-Array der Länge 3 zurückgibt.

c) Schreibe eine Main-Methode und teste die Koordinantenumrechnung für die geographischen Pole ($\varphi = \pm 90°$) und den Schnittpunkt des Äquators mit dem 0-ten Längengrad ($\varphi = 0°$, $\lambda = 0°$) und der Datums-/Zeitgrenze ($\varphi = 0°$, $\lambda = 180°$).

Benutze für pi die Konstante aus der `Math`-Klasse. Darin befinden sich auch die benötigten trigonometrischen Funktionen `Math.sin()` bzw. `Math.cos()`.

Hilfreiche Links:
`http://de.wikipedia.org/wiki/Geographische Koordinaten`
`http://de.wikipedia.org/wiki/Kugel`

3 Für das nächste Blatt

Über die Weihnachtszeit behandeln wir keinen neuen Stoff sondern bieten euch ein Bonusblatt mit kleinen abgeschlossenen Projekten zum Üben des bisherigen Stoffs an.