Programmierstarthilfe SS 2008
Fakultät für Ingenieurwissenschaften und Informatik

7. Blatt
Für den 9. und 10.6.2008

Organisatorisches

Um auf die Mailingliste aufgenommen zu werden schicke einfach eine Mail an `guido.de-melo@uni-ulm.de`.

Die Webseiten zur Veranstaltung sind unter `http://www.uni-ulm.de/in/mi/lehre/ss2008/programmierstarthilfe.html` zu finden.

1. Klassen und Objekte

1.1. Geometrische Figuren

- Erstelle eine Klasse `Punkt`:
  - Attribute: `int x`, `int y`
  - Methoden: keine
- Erstelle eine Klasse `Linie`:
  - Attribute: `Punkt a`, `Punkt b`
  - Methoden: `double getLaenge()`
- Erstelle eine Klasse `Rechteck`:
  (Hier wird keine zweite Linie verwendet, um auszuschließen, dass sich die Linien gar nicht berühren, oder nicht rechtwinklig aufeinander stehen)
  - Attribute: `Linie grundseite`, `int hoehe`
  - Methoden: `double getFlaeche()`
- Erstelle eine Klasse `Dreieck`: *
  - Attribute: `Punkt a`, `Punkt b`, `Punkt c`
  - Methoden: `double getFlaeche()`

Schreibe eine Klasse mit main-Methode und teste die obigen Klassen.

*etwas kniffelig

Hinweis: die Wurzel von x berechnet man mit `Math.sqrt(x)`. Das Ergebnis ist ein `double`. Die Strecke zwichen zwei Punkten lässt sich mit Pythagoras berechnen ($a^2 + b^2 = c^2$). Die Fläche eines Dreiecks ist $\frac{g \cdot h}{2}$ mit Grundseite g und Höhe $h \perp g$. Die Fläche eines Rechtecks ist natürlich $a \cdot b$.

1.2. 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.

1.3. Eine eigene Datums-Klasse

In einigen Aufgaben hast du wahrscheinlich bereits die Klasse `GregorianCalendar` verwendet. In dieser Aufgabe schreiben wir uns eine eigene Datums-Klasse.

a) Schreibe eine Klasse `Datum` mit den Variablen `jahr`, `monat` und `tag`, und der Methode `setDatum(int j, int m, int t)`. Die Methode soll ihre Parameter j, m und t (für Jahr, Monat und Tag) zuerst auf ihre Gültigkeit überprüfen, und sie dann in den entsprechenden Variablen speichern. Sollte ein Wert ungültig sein, soll er durch 1 ersetzt werden und eine Fehlermeldung ausgegeben werden. Teste deine Datums-Klasse in der main-Methode, indem du einige Daten erzeugst und Werte darin speicherst.

b) Ergänze deine Datums-Klasse um die Methoden `boolean istFrueherAls(Datum d)`, die `true` zurückgibt falls im Objekt gespeicherte Datum früher als das als Parameter übergebene ist, und `void istSoundsoLangHer(Datum aktuellesDatum)` das auf der Konsole den Zeitunterschied der beiden Daten ausgibt.

1.4. 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`

1.5. Buchsammlung

Eine kleine Sammlung von Büchern soll in Java implementiert werden. Hierfür soll eine Klasse `Buch` geschrieben werden. Die Klasse soll die Attribute Titel, Autor, ISBN-Nummer als `String` sowie Seitenzahl als `Integer` und Preis als `Float` besitzen.

a) Implementiere die Klasse `Buch`.

b) Implementiere die Methode `ausgabe()` für die Klasse `Buch`, die einen String zurückgeben soll, in dem Titel, Autor und ISBN-Nummer tabellarisch angegeben werden.

c) Erzeuge nun zum Testen eine neue Java-Datei und erstelle in der `main`-Methode fünf Buch-Objekte mit den Daten deiner Lieblingsbücher. Speichere die `Buch`-Objekte anschließend in ein Array für Objekte vom Typ `Buch`. Lasse nun alle im Array enthaltenen Bücher ausgeben. Nutze hierfür, dass die Objekte bereits eine Methode `ausgabe()` besitzen.

d) Berechne die Summe der Preise aller Bücher sowie die Anzahl aller Seiten der Buchsammlung und lasse sie ausgeben.

e) * Aus einer ISBN-Nummer lassen sich viele Informationen über Bücher ablesen. Steht an der ersten Stelle eine 0 oder 1, so ist das Buch englisch, eine 3 steht hingegen für

deutsch. Erweitere nun die Klasse `Buch` um eine Methode `erkenneSprache()`, die als String die Sprache zurückgeben soll, in welcher das Buch laut ISBN-Nummer ist. Achtung: Es gibt auch ein erweitertes Format für ISBN-Nummern, bei denen vor der eigentlichen Nummer noch 978- oder 979- angehängt ist. In diesem Fall muss die fünfte Stelle untersucht werden.
Erweitere nun deine Ausgabe-Methode um die zusätzliche Angabe der Sprache des Buches.

Hilfreiche Links:
`http://de.wikipedia.org/wiki/Internationale_Standardbuchnummer`

2. Bonusaufgaben

2.1. 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) Erstelle für die Klasse `Vektor` eine Methode `ausgabe()`, die die drei Werte der Komponenten des Vektors ausgibt.

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

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

d) 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.

2.2. *Eine Matrix-Klasse

Wir haben jetzt schon einige Dinge mit Matrizen gemacht, wie zum Beispiel Matrizenaddition und -multiplikation, Spiegelung an einer Diagonalen oder das Anlegen und Füllen einer Wellenmatrix. Mit der Objektorientierung haben wir nun die Möglichkeit unsere Methoden zu einer Klasse `Matrix` zu sammeln, die alle diese Dinge als Methoden verfügen kann und zudem ein geeignetes Speicherobjekt für Matrizen darstellt. Diese Klasse kann man später immer wieder benutzen und um Funktionen erweitern (Matrizen wird man immer brauchen, überall).

Implementiere die Klasse `Matrix` nach deinen eigenen Vorstellungen. Mach dir dafür zu den folgenden Punkten Gedanken. Es gibt hierbei keine richtigen oder falschen Antworten, sondern es geht darum, wie du die `Matrix`-Klasse haben möchtest.

- Wie sollen die Werte in der Matrix gespeichert werden? Welchen Datentyp willst du zu Grunde legen? Wie sollen die Werte von außen gesetzt und abgefragt werden?
- Woher weiß das Matrix-Objekt, welche Dimension die Matrix haben soll? Wie speichert man die Dimension? Macht es Sinn die Dimension während der Nutzung zu verändern? Was ist mit nicht-quadatischen Matrizen? Wie bekommt man aus dem Objekt während der Nutzung die Dimension, falls man sie braucht?
- Soll es eine Methode geben, die die Matrix ausgibt? Soll das die `toString()`-Methode sein? Was macht man dann mit Zeilenumbrüchen?
- Welche der eingangs erwähnten Methoden hast du schon implementiert und möchtest du einbauen? Wie sind diese zu ändern, damit sie mit dem Matrix-Objekt arbeiten können? Gibt es noch weitere Methoden, die du einbauen willst?