Programmierstarthilfe WS 2008/09
Fakultät für Ingenieurwissenschaften und Informatik

8. Blatt
Für die Woche vom 8.12. bis zum 12.12.2008 (KW 50)

Organisatorisches

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

Diese Woche beginnen wir mit den Konzepten der Objektorientierung. Bitte achte diesmal bei der Formatierung deines Codes auf folgende Punkte:

- Klassennamen schreiben wir immer groß.
- Namen von Methoden beginnen immer mit einem kleinen Buchstaben.
- Attribute und Variablen werden ebenfalls klein geschrieben um sie von Klassen zu unterscheiden.

Als erstes führen wir neue Datentypen ein indem wir eigene Klassen deklarieren. Die erste Aufgabe ist zu den komplexen Zahlen. Durch verwenden eines Konstruktors sehen wir, dass die Attribute der erzeugten Objekte komfortabler gefüllt werden können. Durch die statischen Methoden sehen wir, wie man viele Operationen mit den Objekten einer Klasse anstellen kann.

Bei der Aufgabe mit der Buchsammlung und den geometrischen Figuren verwenden wir nichtstatische Methoden um direkt mit den Objekten zu arbeiten.

1 Aufgaben

1.1 Eine eigene Datums-Klasse für Sexagesimal-Kalender

In einer früheren Aufgabe hast du wahrscheinlich bereits die Klasse `GregorianCalendar` verwendet. In dieser Aufgabe schreiben wir uns nun eine eigene Datums-Klasse für eine andere Kalenderform. Der sogenannte Sexagesimal-Kalender besteht aus sechs Monaten mit jeweils zehn Wochen. Eine Woche in diesem Kalendersystem hat außerdem nur 6 Tage. Die Beachtung von Schaltjahren und Ausgleichstagen ignorieren wir bei dieser Aufgabe.

a) Schreibe eine Klasse `Datum` mit den Attributen `jahr`, `monat` und `tag`, und einem entsprechenden Konstruktor.

b) Schreibe nun eine statische Methode `pruefeDatum`, welche ein Datumsobjekt übergeben bekommt und prüft, ob dieses Datum gültig ist oder nicht.

c) Ergänze dein Programm um eine weitere statische Methode `berechneAbstandInTagen`, welche nun zwei verschiedene Datumsobjekte übergeben bekommt und berechnet, wie weit beide Daten auseinander liegen. Der Unterschied soll in Tagen zurückgegeben werden.

1.2 Komplexe Zahlen

Mit Hilfe von Klassen lassen sich in Java beliebige neue Datentypen erstellen, beispielsweise auch die Komplexen Zahlen. Die Komplexen Zahlen erweitern die reelen Zahlen derart, dass die Gleichung $x^2 = -1$ gelöst werden kann. Dies wird dadurch erreicht, dass die reellen Zahlen um eine imaginäre zweite Dimension erweitert werden. Jede Komplexe Zahl c besteht also aus einem Realteil c_{Re} und Imaginärteil c_{Im}, so dass gilt: $c = (c_{Re}, c_{Im})$.

a) Schreibe eine Klasse `ComplexNumber`, mit der sich eine Komplexe Zahl darstellen lässt. Erweitere die Klasse um einen Konstruktor, so dass man den Real- und Imaginärteil direkt bei der Erstellung angeben kann. Überlade diesen Konstruktor zusätzlich mit einem zweiten Konstruktor, der nur den Realteil als Parameter übergeben bekommt. In diesem Fall soll der Imaginärteil automatisch auf 0 gesetzt werden.

b) Unter Beachtung spezieller Rechenregeln ist es möglich, mit Komplexen Zahlen wie gewohnt zu rechnen:

$a + b = (a_{Re} + b_{Re},\ a_{Im} + b_{Im})$
$a - b = (a_{Re} - b_{Re},\ a_{Im} - b_{Im})$
$a \cdot b = (a_{Re} \cdot b_{Re} - a_{Im} \cdot b_{Im},\ a_{Re} \cdot b_{Im} + a_{Im} \cdot b_{Re})$

Schreibe statische Methoden für die Addition, die Subtraktion und die Multiplikation. Die Methoden sollen jeweils zwei Komplexe Zahlen als Parameter übergeben bekommen und als Ergebnis wieder Komplexe Zahlen zurück geben.

c) Implementiere eine weitere statische Methode `print`, die eine Komplexe Zahl auf der Konsole ausgibt. Ist der Imaginärteil der Zahl gleich 0, so soll nur der Realteil ausgegeben werden, andernfalls Real- und Imaginärteil.

d) Berechne das Ergebnis des Ausdrucks $\big((12,5) - (15,7) + (3,3)\big) \cdot \big((8,1) + (-7,0) + 1\big)$ und gib es auf der Konsole aus.

1.3 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) Erzeuge nun zum Testen eine neue Klasse, die die `main`-Methode enthält, und erstelle darin fünf Buch-Objekte mit den Daten deiner Lieblingsbücher. Implementiere die Methode `ausgabe(Buch b)`, die einen String zurückgeben soll, in dem Titel, Autor und ISBN-Nummer tabellarisch angegeben werden.

c) Speichere die `Buch`-Objekte anschließend in einen Array für Objekte vom Typ `Buch`. Lasse nun alle im Array enthaltenen Bücher ausgeben. Hier hilft deine Methode `ausgabe(Buch b)`.

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

1.4 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()`
- Für Fortgeschrittene: Erstelle eine Klasse `Dreieck`: (* Bonus)
  - Attribute: `Punkt a`, `Punkt b`, `Punkt c`
  - Methoden: `double getFlaeche()`

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

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 $a \cdot b$.

2 Bonusaufgaben

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

2.1 Familienplanung

Definiere eine Klasse `Familie`.

a) Deine Klasse soll ein Array mit den Namen der Familienmitglieder haben, wobei die ersten beiden Arrayeinträge für die Elternteile reserviert sind und die darauffolgenden für die Kinder. Lege das Array groß genug an, so dass Familienzuwachs möglich ist. Die unbelegten Plätze im Array sollen mit dem String `frei` markiert werden. In einem zweiten Array speichere in der gleichen Reihenfolge die Geburtsdaten ab, freie Felder markiere mit `-1`. Außerdem soll es einen Zähler für die Anzahl der Kinder geben.

b) Lege die Methode `zuwachs` an. Welche beiden Parameter brauchst du?

c) Schreibe außerdem eine Methode, die dir die Familienmitglieder mit ihren Geburtsjahren ausgibt.

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?

3 Für das nächste Blatt

Nächste Woche werden wir noch mehr mit Objekten arbeiten. Wir führen statische Attribute ein. Lies dazu das Kapitel *Objektorientierung* im Skript zu Ende.