Programmierstarthilfe SS 2008

Fakultät für Ingenieurwissenschaften und Informatik

8. Blatt

Für den 16. und 17.6.2008

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. Konstruktoren

In der letzten Woche habt ihr Objekte angelegt und dann mit Werten gefüllt, etwa so:

```
1 Buch informatiker_buch = new Buch();
2 informatiker_buch.autor = "Douglas Adams";
3 informatiker_buch.titel = "Per Anhalter durch die Galaxis";
```

Dabei ist `Buch()` eine sehr spezielle Methode, die nur durch das Schlüsselwort `new` aufgerufen werden kann, genauso heißen muss, wie die Klasse. Diese Methode nennt man Konstruktor.

Jede Klasse besitzt automatisch einen Konstruktor, der nicht sehr viel tut. Aber als Programmierer kann man einer eigenen Klasse auch einen Konstruktor schreiben, der etwas mehr kann. Ein Beispiel für unsere Buch-Klasse wäre:

```
1 public Buch() {
2     titel = "namenlos";
3     autor = "autorenlos";
4 }
```

Der Konstruktor hat wie ihr seht, keinen spezifizierten Rückgabewert und auch keine `void`. Aber er kann trotzdem Parameter entgegennehmen. Das wird oft genutzt, um das Objekt mit Startwerten zu füllen, etwa so:

```
1 public Buch(String buch_autor, String buch_titel) {
2     titel = buch_titel;
3     autor = buch_autor;
4 }
```

Wenn ich jetzt ein Objekt vom Typ `Buch` erzeugen will, kann ich direkt im Konstruktor schon Titel und Autor übergeben. Um zum Beispiel "den Anhalter" von oben anzulegen, reicht Folgendes aus:

```
1 Buch informatiker_buch = new Buch("Douglas Adams",
2     "Per Anhalter durch die Galaxis");
```

Eine Klasse kann mehrere Konstruktoren haben, die sich nur in ihren Parametern unterscheiden. Unsere Klasse Buch könnte z.B. die Konstruktoren `public Buch()` und `public Buch(String buch_autor, String buch_titel)` und `public Buch(String buch_autor, String buch_titel, int seitenzahl)` haben, so dass man für jeden Zweck den passenden Konstruktor hat.

1.2. public und private

Jedes Attribut einer Klasse und jede Methode kann eine der Eigenschaften `public` oder `private` erhalten. Hat sie keine davon, gilt sie etwa als `public` (erst bei Benutzung von Paketen ergibt sich hier ein Unterschied).

Wenn ein Attribut oder eine Methode `private` ist, kann man es/sie außerhalb der Klasse nicht benutzen, bei `public` schon. Das nutzt man vor allem, um dem Anwender, der die Objekte erzeugt (außerhalb der Klasse) zu verbieten die Werte in der Klasse unsinnig zu setzen.

Wenn ihr euch zum Beispiel an die Artikel aus dem Vortrag in der letzten Woche erinnert, so wollten wir verhindern, dass man eine negative Anzahl speichern kann. Dafür haben wir eine Methode `setAnzahl(int neue_anzahl)` geschaffen, über die die Anzahl in Zukunft ausschließlich gesetzt werden soll. Diese Methode machen wir `public` (also öffentlich nutzbar), dafür wird das Attribut `anzahl` `private`. Nun kann nicht mehr mit `artikel.anzahl` auf das Attribut zugegriffen werden. Dann sieht der Artikel etwa so aus:

```
1 class Artikel {
2     String name;
3     private int anzahl;
4     float preis;
5
6     public void setAnzahl(int neue_anzahl) {
7         if (neue_anzahl < 0) System.out.println("Zu wenig.");
8         else anzahl = neue_anzahl;
```

```
9       }
10  }
```

Probiert nun ruhig aus, was bei Aufrufen von etwa `artikel.anzahl = 10;` passiert.

1.3. static

Das Schlüsselwort `static` kann man, wie ihr schon wisst, vor Methoden setzen. Die Methode ist dann statisch, das heißt unabhängig davon ob ein Objekt angelegt ist oder nicht.

Deshalb braucht man auch (im Gegensatz zu nicht-statischen Methoden) kein Objekt, um die Methode zu nutzen. Um zum Beispiel `Math.sqrt(2)` aufzurufen muss ich kein Objekt der Java-Klasse `Math` erzeugen, ein Aufruf wie

```
1  Math meine_mathematik = new Math();
```

ist also unnötig. Das ist auch logisch, denn warum soll sich die Wurzelrechnung in Java von der in der Mathematik unterscheiden?

Dafür dürfen statische Methoden nicht mehr auf Daten ihres Objekts arbeiten. So etwas wie

```
1  anzahl = neue_anzahl;
```

aus der Artikel-Methode von oben ginge also nicht, wenn `setAnzahl` statisch wäre, denn `anzahl` ist ein Attribut des Objekts Artikel. Daher ist `setAnzahl` aus gutem Grund nicht statisch, denn die Anzahl eines Artikels zu setzen hängt ganz offensichtlich mit dem konkreten Artikel zusammen.

1.4. Referenzen

Letzte Woche war die Rede davon, dass zwei Objektvariablen auf dasselbe Objekt verweisen können, da in den Variablen nur die Referenz auf das Objekt, also die Adresse, gespeichert ist. Ihr habt auch schon erfahren, dass es bei der Parameterübergabe an eine Methode einen Unterschied zwischen primitiven Datentypen (wie z.B. `int` und `float`) und Objektdatentypen (wie z.B. Arrays oder Objekte selbstangelegter Klassen) gibt. Bei primitiven Datentypen wird nur der Wert kopiert (das nennt man "call by value"), Objektdatentypen werden als Referenzen übergeben, d.h. die Adresse des Objekts wird übergeben und das Objekt selbst kann in der Methode verändert werden ("call by reference").

Wofür kann das gut sein? Wenn du beispielsweise eine Klasse `Vector` hast, die zwei verschiedene Konstruktoren und die Methoden `print()` und `multiply(float,Vector)` besitzt, kannst du deinen leeren Ergebnisvektor übergeben und in der Methode füllen lassen:

```
1 Vector v = new Vector(1.0f, 2.3f, 1.2f);
2 Vector result = new Vector();
3 v.multiply(2.2, result);
4 result.print();
```

Die Ausgabe am Ende gibt dir dann den Ergebnisvektor aus.

Eine besondere Referenz ist die `null`-Referenz. Mit dem Schlüsselwort `null` erzeugt man eine leere Referenz, die auf keine Daten zeigt. Setzt man alle Referenzen auf ein Objekt auf `null` dann löscht Java die zugehörigen Daten aus dem Speicher. Achtung, versucht man über die `null`-Referenz auf Eigenschaften oder Methoden zuzugreifen kommt es zu einem Laufzeitfehler:

```
1     Artikel a = new Artikel();
2     a = null; // a ist leere Referenz
3     a.anzahl = 5; // Null-Pointer-Fehler und Programmabsturz
```

2. Aufgaben

2.1. Kontoverwaltung

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

2.2. Nochmal Vektoren

Die Klasse `Vektor` soll einen mathematischen Vektor darstellen, der aus drei Komponenten besteht, die jeweils als `float`-Wert gespeichert werden sollen. (Hinweis: Falls du die Vektoraufgabe auf dem letzten Blatt bearbeitet hast, kannst du jetzt deine Klasse `Vektor` einfach erweitern. Dies ist aber keine Voraussetzung.)

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 einen Konstruktor `public Vektor(float a, float b, float c)` hinzu, den man benutzt um einen Vektor $\begin{pmatrix} a \\ b \\ c \end{pmatrix}$ zu erstellen.

d) 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 $\begin{pmatrix} a \\ a \\ a \end{pmatrix}$ ist die Länge $laenge = a * \sqrt{3}$.)

Teste alle deine Konstruktoren, indem du in der Mainmethode einige Vektorobjekte erstellst und ausgibts.

2.3. 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?

2.4. Nochmal Kontoverwaltung

Erweitere hier deine Klasse `Konto`. Überlege dir auf welche Werte eines Kontoobjekts man von außen nicht zugreifen sollte, und setze die Rechte dementsprechend. Stelle anschließend die Methoden bereit, die man nun zum Benutzen des Kontos braucht.

2.5. Website-Rating

Über eine Klasse `Website` sollen zu einer Website die Adresse und die zugehörige Bewertung zwischen null und fünf Sternen gespeichert werden. Schreibe die Klasse und lenke den zugriff über `public` und `private`. Nach außen hin sollen nur drei Methoden sichtbar sein, eine zum Abrufen der aktuellen Bewertung, eine, die 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).

3. Bonusaufgaben

3.1. 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, sodass 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? Schreibe außerdem eine Methode, die dir die Familienmitglieder mit ihren Geburtsjahren ausgibt.