

Übungsblatt 3

Besprechung in der Vorlesung am 23.03.2006

1 Spiel, Satz, Sieg

Es sei folgende einfache Klasse gegeben, die zur Speicherung von Daten über Tennisspieler (z.B. bei einem Turnier) verwendet werden könnte:

```
public class TennisSpieler {  
    public String name;    // Name des Spielers  
    public int punkte;    // Punkte in der Weltrangliste  
    public int punktDifferenz (int punkte) {  
        // Differenz der Punkte mit mathematischer Betragsbildung  
        return Math.abs(punkte - this.punkte);  
    }  
}
```

1. Zeichnen Sie das UML-Klassendiagramm für diese Klasse.
2. Was bewirken die nachfolgenden Anweisungen? Warum benötigt man die zweite Anweisung überhaupt?

```
TennisSpieler williams;  
williams = new TennisSpieler();
```

3. Erläutern Sie die Bedeutung der `this`-Referenz anhand der Methode `punktDifferenz`.
4. Wie erfolgt der Zugriff auf die Attribute und Methoden dieser Klasse?
5. Was versteht man unter einem Konstruktor? Wie würde ein geeigneter Konstruktor für die Klasse `TennisSpieler` aussehen, der die Attribute des Spielers bei der Objekterzeugung initialisiert? Wenn Sie die Klasse um diesen Konstruktor ergänzen, ist dann die folgende Anweisung noch zulässig?

```
TennisSpieler williams = new TennisSpieler();
```

6. Was ist der Unterschied zwischen Attributen (Instanzvariablen) und Klassenvariablen?
7. Erweitern Sie die Klasse `TennisSpieler` um ein Attribut namens `verfolger`, das eine Referenz auf einen weiteren Tennisspieler (den unmittelbaren Verfolger in der Weltrangliste) darstellt, und um ein Attribut `startNummer`, das es ermöglicht, allen Tennisspielern (z.B. bei der Erzeugung eines neuen Objektes für

die Teilnehmerliste eines Turniers) eine eindeutige ganzzahlige Nummer zuzuordnen.

8. Erweitern Sie die Klasse `TennisSpieler` um eine Klassenvariable namens `folgeNummer`, die die jeweils nächste zu vergebende Startnummer enthält.
9. Modifizieren Sie den Konstruktor der Klasse `TennisSpieler` so, dass er jeweils eine entsprechende Startnummer vergibt und die Klassenvariable `folgeNummer` entsprechend erhöht. Geben Sie auch eine Überladung dieses Konstruktors an, die es ermöglicht, bei der Objekterzeugung auch noch den Verfolger in der Weltrangliste anzugeben.
10. Wie verändert sich der Wert der Variablen `startNummer` und `folgeNummer` in den Objekten `federer`, `schuettler` und `agassi` mit den nachfolgenden Anweisungen?

```
TennisSpieler federer, schuettler, agassi;
federer = new TennisSpieler ("R. Federer", 167);
schuettler = new TennisSpieler ("R. Schuettler", 226, federer);
agassi = new TennisSpieler ("A. Agassi", 236, schuettler);
```

11. Erweitern Sie die Klasse `TennisSpieler` um eine Methode namens `istLetzter`, die genau dann den Wert `true` liefert, wenn das `TennisSpieler`-Objekt keinen Verfolger in der Weltrangliste hat.
12. Erweitern Sie die Klasse `TennisSpieler` um die folgende Methode:

```
public String toString () {
    String printText = name + " (" + startNummer + ")";
    if (verfolger != null)
        printText = printText + " liegt vor " + verfolger.toString();
    return printText;
}
```

Diese Methode ermöglicht es, dass man Objekte der Klasse innerhalb von Zeichenkettenausdrücken (also auch in Ausgabeanweisungen) automatisch in Strings umwandeln kann. Was würden die folgenden Zeilen dann ausgeben?

```
System.out.println(federer);
System.out.println(schuettler);
System.out.println(agassi);
```

13. Wie kann man vermeiden, dass ein Programmierer bei der Bearbeitung der Objekte der Klasse `TennisSpieler` die (von den Konstruktoren automatisch generierten) Startnummern überschreibt? Wie kann man trotzdem lesenden Zugriff auf die Startnummern ermöglichen?

2 Mensch, ärgere Dich nicht!

1. Schreiben Sie eine Klasse `Mensch`, die private Attribute beinhaltet, um den Namen, das Alter und das Geschlecht (als boolescher Wert, wobei `true` "weiblich" bedeutet) eines Menschen zu speichern.
2. Deklarieren Sie in `Mensch` zwei konstante Klassenvariablen `MAENNLICH` und `WEIBLICH`, die die oben beschriebenen booleschen Codes für die beiden Geschlechter enthalten.
3. Statten Sie die Klasse mit einem Konstruktor aus, der als Parameter den Namen, das Alter und das Geschlecht übergeben bekommt und die entsprechenden Attribute des Objekts mit diesen Werten belegt. Es sollte damit möglich sein, Objekte wie folgt zu instanzieren:

```
Mensch adam = new Mensch ("Adam", 21, Mensch.MAENNLICH);
Mensch eva = new Mensch ("Eva", 20, Mensch.WEIBLICH);
```

4. Statten Sie die Klasse mit öffentlichen "Gettern" und "Settern" aus, die den Zugriff auf die privaten Attribute von außen erlauben.
5. Überschreiben Sie die von `Object` geerbte `toString`-Methode so, dass sie eine Zeichenkette zurückliefert, die sich aus dem Namen, dem Alter und dem Geschlecht des Menschen zusammensetzt. Achten Sie darauf, dass Sie die Signatur von `Object.toString` exakt übernehmen – was würde geschehen, wenn Sie `Mensch.toString` eine andere Signatur gäben?
6. Schreiben Sie zum Testen Ihrer Klasse `Mensch` eine einfache Klasse `TestMensch`, die in ihrer `main`-Methode mit Objekten der Klasse `Mensch` arbeitet und den Konstruktor und die Methoden von `Mensch` testet. Testen Sie dabei auch, ob der Compiler wirklich Direktzugriffe auf die privaten Attribute verweigert, und ob für ein Objekt `m` der Klasse `Mensch` tatsächlich die `toString`-Methode aufgerufen wird, um die Anweisung `System.out.println(m);` auszuführen.
7. Angenommen, wir deklarieren und initialisieren zusätzlich zu `adam` und `eva` noch folgende Variablen:

```
Mensch adamKlon = adam;
Mensch evaKlon = new Mensch ("Eva", 20, Mensch.WEIBLICH);
```

Welche booleschen Werte haben dann die folgenden Ausdrücke? Warum?

```
adam == eva
adam == adamKlon
eva == evaKlon
```

3 Inkrement und kein Ende

Gegeben sei das folgende Java-Programm:

```
public class Plus {
    public static void main (String args []) {
        int a = 1, b = 2, c = 3, d = 4;
        System.out.println(++a);
        System.out.println(a);
        System.out.println(b++);
        System.out.println(b);
        System.out.println(++c + (++c));
        System.out.println(c);
        System.out.println((d++) + (d++));
        System.out.println(d);
    }
}
```

Vollziehen Sie das Programm nach, und überlegen Sie sich, welche Werte ausgegeben werden. Verifizieren Sie danach Ihre Ergebnisse durch einen Programmlauf.

4 Dosenravioli sind nahrhaft...

...und deshalb machen Sie gerade Urlaubsvertretung für einen Verpackungsingenieur bei der Firma *Raviolita*. Dieser hat Ihnen kurz vor seiner Abreise in den Spontanurlaub noch das Programm (bzw. die Klasse) *Raviolita* hinterlassen:

```
public class Raviolita {

    public static void main (String[] args) {

        final double PI = 3.141592;
        double u, h;
        u = ___; // geeignete Testwerte einbauen
        h = ___; // geeignete Testwerte einbauen

        // hier die fehlenden Deklarationen ergaenzen
        // hier die fehlenden Berechnungen ergaenzen
        // hier die fehlenden Ausgaben ergaenzen
    }
}
```

Dieses Programm führt Berechnungen durch, die bei der Herstellung von Konservendosen aus einem Blechstück mit

- Länge u (Umfang der Dose in Zentimetern) und
- Breite h (Höhe der Dose in Zentimetern)

anfallen. Dieses Programm sollen Sie nun so vervollständigen, dass es ausgehend von den Variablen u und h und unter Verwendung der Konstanten π (bzw. $PI = 3.141592$) die folgenden Werte berechnet und ausgibt:

- den Durchmesser des Dosenbodens: $d_{boden} = \frac{u}{\pi}$
- die Fläche des Dosenbodens: $a_{boden} = \pi \cdot \left(\frac{d_{boden}}{2}\right)^2$
- die Mantelfläche der Dose: $a_{mantel} = u \cdot h$
- die Gesamtfläche der Dose: $a_{gesamt} = 2 \cdot a_{boden} + a_{mantel}$
- das Volumen der Dose: $v = a_{boden} \cdot h$

Testen Sie Ihr Programm mit vernünftigen Daten für u und h .