Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 11

Bearbeitungszeit: 23.06. bis 29.06.2008

Aufgabe 1: Lebensmittel-Pakete

In den bisherigen Übungsaufgaben haben wir nur Klassen verwendet, die im selben Verzeichnis lagen. Bei einem größeren Projekt mit vielen Klassen werden dann allerdings die Orientierung und die Verwaltung der Klassen schwieriger.

Um diesen Schwierigkeiten entgegenzuwirken, wird das Konzept der **Packages** eingeführt. Ein Package ist eine Sammlung mehrerer logisch zusammengehörender Klassen. Packages haben einen eindeutigen Namen, der den normalen Konventionen für Bezeichner entspricht (in der Praxis üblicherweise Kleinbuchstaben). Packages können geschachtelt werden, also weitere „Subpackages“ enthalten. Die Namen geschachtelter Packages werden durch Punkte getrennt.

Eine Klasse **`ArrayList`**, die im Subpackage **`util`** des Packages **`java`** liegt, wird also vollständig als **`java.util.ArrayList`** bezeichnet – das ist auch ein reales Beispiel, denn alle Klassen der Java-API sind in Packages organisiert!

Sie können Klassen aus einem Package verwenden, indem Sie den vollständigen Namen der Klasse verwenden (**`java.util.ArrayList x = new java.util.ArrayList();`**).

Einfacher wird es, wenn Sie ganz am Anfang Ihrer Datei (noch vor **`class`**) die **`import`**-Klausel verwenden (**`import java.util.ArrayList;`**), dann können Sie die Klasse ohne den Package-Pfad verwenden (**`ArrayList x = new ArrayList();`**).

Wenn Sie mehrere Klassen importieren wollen, können Sie das entweder einzeln tun oder „**`*`**“ verwenden: **`import java.util.*;`** importiert alle Klassen aus dem Package **`java.util`** – aber nicht auch aus Subpackages von **`java.util`**! **`import java.*;`** liefert also nicht das gewünschte Ergebnis.

Wenn Sie eigene Klassen in ein Package einordnen wollen, muss jede Datei vor **`class`** und **`import`** mit **`package <packagename>;`** beginnen. Aber Achtung: die Namen von Klassen, Interfaces und Packages müssen eindeutig sein, **u**nd jede Klasse kann nur Mitglied genau eines Packages sein.

Außerdem werden Packagestrukturen normalerweise auf Verzeichnisstrukturen des Betriebssystems abgebildet, die Dateien liegen also in Unterverzeichnissen des Projektverzeichnisses. Darum kümmert sich glücklicherweise BlueJ.

Struktur-Beispiel für eine eigene Klasse:

```
package allginf.test;
import java.util.ArrayList;
import ...;
public class Test ... {
    ...
}
```

Probieren Sie nun das Anlegen und Arbeiten mit Packages in einem einfachen Beispiel aus:

a) Starten Sie BlueJ.

b) Erzeugen Sie in einem neuen Projekt ein Package **`lebensmittel`** (rechte Maustaste). In der deutschen BlueJ-Version werden Packages als „Pakete“ bezeichnet.

c) Doppelklicken Sie auf das neue Symbol, um in das neue Package zu wechseln.

d) Erzeugen Sie in **`lebensmittel`** die beiden Packages **`suessigkeiten`** und **`obst`**.

e) Schreiben Sie Klassen **`Apfel`**, **`Orange`**, **`Schokolade`**, **`Kekse`** und **`Brot`** mit jeweils einer statischen Methode **`void print()`**, die den Namen des jeweiligen Lebensmittels auf dem Bildschirm ausgibt. Ordnen Sie die Klassen in die von Ihnen angelegten neuen Packages gemäß ihrer Lebensmittelgruppe ein.

   Jede Klasse muss mit einer Package-Deklaration beginnen (z.B. **`package lebensmittel.obst;`**). Das erledigt zwar BlueJ für Sie, schauen Sie sich diese Zeile jedoch genau an, später müssen Sie diese Befehle selbst einfügen.

f) Legen Sie in Ihrem BlueJ-Projekt (außerhalb der Packages) eine Klasse **`Einkaufskorb`** mit einer Methode **`public static void test()`** an. In dieser können Sie die **`print`**-Methode von einigen der Lebensmittelklassen aufrufen und damit z.B. folgende Ausgabe erzeugen:

   ```
   In meinem Einkaufskorb befinden sich:
   Brot
   Schokolade
   Orange
   ```

   Denken Sie an evtl. notwendige **`import`**-Anweisungen am Anfang der Datei.

g) Erweitern Sie, wenn Sie möchten, ihre Package-Struktur um weitere Lebensmittelgruppen-Subpackages und Lebensmittelklassen.

Aufgabe 2: Züge

Züge bestehen aus einer Lokomotive und mehreren Waggons. Bei den Lokomotiven gibt es als spezielle Ausprägung die Diesel- und die Elektro-Lokomotive. Bei Waggons gibt es die Spezialisierungen Schlafwagen und Speisewagen.

Lokomotiven charakterisieren sich außerdem durch eine Typ-Nummer und ihre Länge, während Waggons Länge, Passagierkapazität und Wagennummer als Eigenschaft besitzen.

Ihre Aufgabe ist es, Züge mit Hilfe von Java-Klassen, die in Paketen liegen, zu beschreiben. Als Vorgabe finden Sie auf unserer Webseite ein vorbereitetes BlueJ-Projekt mit der Klasse **`Locomotive`** im Package **`train.locomotive`** und der Klasse **`Car`** im Package **`train.car`**. Schauen Sie sich die beiden Klassen genau an und gehen Sie dann wie folgt vor:

a) Erstellen Sie im vorgegebenen Projekt die Klassen **`DieselLocomotive`** und **`ElectricLocomotive`**. Beide sollen von **`Locomotive`** erben (**`extends`**) und im Package **`train.locomotive`** sein. In beiden Klassen müssen Sie den Konstruktor und die Methode **`toString()`** überschreiben.

b) Erstellen Sie die Klassen **`DiningCar`** und **`SleepingCar`**. Beide sollen von **`Car`** erben und im Package **`train.car`** sein. Wieder müssen Sie den Konstruktor und die Methode **`toString()`** überschreiben.

c) Schreiben Sie im Package **`train`** eine Klasse **`Train`**. Diese Klasse soll folgende Methoden anbieten:

   1. Einen Konstruktor, der eine Lokomotive als Parameter erwartet und einen ziemlich kurzen Zug – nur aus einer Lok ohne Wagen bestehend – erzeugt.
   2. **`void add(Car car)`** soll einen Waggon am Ende des Zuges anhängen.
   3. **`int getCapacity()`** soll die gesamte Passagierkapazität berechnen und zurückgeben.
   4. **`int getLength()`** soll die gesamte Zuglänge berechnen und zurückgeben.
   5. **`Car removeLast()`** soll den letzten Waggon vom Zug abhängen und als Ergebnis zurückgeben. Falls der Zug keine Waggons hat, soll **`null`** zurückgegeben werden.
   6. **`String toString()`** soll einen Text folgender Art zurückgeben:
      `ElectricLocomotive Typ: 3672 <-> Car Nr 1 <-> DiningCar Nr 3 <-> Car Nr 4 <-> SleepingCar Nr 7`

   Verwenden Sie zur Verwaltung der Waggons in einer Liste die Klasse **`ArrayList`** oder **`LinkedList`** aus dem Package **`java.util`** (siehe Java-API!). Sie benötigen die Methoden **`void add(Object o)`**, **`Object get(int index)`**, **`Object remove(int index)`** und **`size()`**. Bei den Methoden **`get`** und **`remove`** müssen Sie einen **TypeCast** vornehmen, um die Methoden der **`Car`**-Objekte aufrufen zu können (Beispiel: **`Car car = (Car) cars.get(3);`**).

d) Erstellen Sie in ihrem Projekt (außerhalb der Packages) eine Klasse **`TrainTest`** mit einer Methode **`public static void test()`**, in der Sie einen Zug erzeugen, mehrere Waggons anhängen und die Funktion aller Methoden überprüfen.

Viel Spaß!