Technische Universität Darmstadt

Fachbereich Informatik

Prof. Dr. Andreas Koch

Allgemeine Informatik 2 im SS 2008

Übungsblatt 4

Bearbeitungszeit: 05.05. bis 11.05.2008

Aufgabe 1: Die Rechteck-Welt

Die Rechteck-Welt besteht aus einem zweidimensionalen Koordinatensystem. Ihre Bewohner sind Rechtecke, deren Seiten parallel zu den Koordinatenachsen sind. Jedes Rechteck ist durch vier Werte charakterisiert: x- und y-Koordinate der linken unteren Ecke, Breite und Höhe. Alle vier Werte sind ganzzahlig und nicht-negativ.

a) Starten Sie BlueJ und erstellen Sie eine neue Klasse **`Rectangle`**, die die Bewohner der Rechteck-Welt repräsentiert. Löschen Sie, wie auch schon in den letzten Übungen, die vordefinierten Inhalte der Klasse.

b) Stellen Sie sicher, dass ein Objekt Ihrer Klasse die vier Werte speichern kann, die ein Rechteck ausmachen. Achten Sie dabei auf sinnvolle Variablennamen.

c) Schreiben Sie nun einen Konstruktor, der vier Parameter erwartet und damit Koordinaten, Breite und Höhe des Rechtecks initialisiert. Achten Sie auch hier auf sinnvolle Parameternamen – das können dieselben sein wie in b), allerdings benötigen Sie dann das Schlüsselwort **`this`**, um auf die Attribute der Klasse zuzugreifen. Beispiel: **`this.name = name;`** weist dem Attribut **`name`** den Wert des Parameters **`name`** zu.

Wenn sich in der Rechteckwelt Rechtecke begegnen, wollen sie sich näher kennen lernen. Oft fragt man da nach den charakterisierenden Eigenschaften.

d) Schreiben Sie also Methoden **`getWidth()`**, **`getHeight()`**, **`getLeft()`**, **`getRight()`**, **`getBottom()`** und **`getTop()`**, die entsprechend ihres Namens die Breite, Höhe und die Begrenzungen (links, rechts, unten, oben) eines Rechtecks zurückgeben.

Besonders interessant ist, ob es sich bei einem Rechteck um ein Quadrat handelt.

e) Schreiben Sie also auch eine Methode **`boolean isSquare()`**, die die Antwort auf diese Frage liefert.

Auch in der Rechteckwelt finden sich Rechtecke, die in einer Beziehung zueinander stehen. Dazu fragt man ein Rechteck, ob es in einer bestimmten Beziehung zu einem anderen Rechteck steht.

f) Schreiben Sie eine Methode **`boolean isContained(Rectangle r)`**, die **`true`** zurückgibt, wenn das gesamte Rechteck, auf dem die Methode aufgerufen wird, innerhalb des Rechtecks **`r`** liegt.

g) Schreiben Sie außerdem eine Methode **`boolean isDisjoint(Rectangle r)`**, die **`true`** zurückgibt, wenn dieses Rechteck und **`r`** sich weder berühren noch ineinander liegen.

h) Sie können sich gerne auch noch weitere Methoden ausdenken, die etwas über ein Rechteck (z.B. Flächeninhalt) oder Beziehungen zwischen Rechtecken (z.B. „größer als“) aussagen!

i) Erstellen Sie schließlich eine weitere Klasse **`RectangleTest`** mit einer **`main`**-Methode, die mehrere Rechtecke erzeugt und Eigenschaften sowie Beziehungen erfragt und auf dem Bildschirm ausgibt.

Aufgabe 2: Straßenbahn

Sicherlich sind Sie in Darmstadt schon Straßenbahn gefahren. Jede Straßenbahn hat eine Liniennummer, sie kann fahren und wieder anhalten und es können Passagiere ein- und aussteigen (allerdings nicht während der Fahrt).

Implementieren Sie eine Java-Klasse, die die genannten Eigenschaften und Funktionen einer Straßenbahn simuliert. Gehen Sie dabei wie folgt vor:

a) Erstellen Sie eine neue Klasse **`Tram`**. Löschen Sie bitte wieder die vordefinierten Inhalte.

b) Deklarieren Sie in der Klasse Variablen, die den Zustand einer Straßenbahn darstellen:

`int line`	Die Liniennummer der Straßenbahn
`boolean driveMode`	**`true`**, falls die Straßenbahn fährt, ansonsten **`false`**
`int passengers`	Die Anzahl der Passagiere in der Straßenbahn

c) Die Klasse soll einen Konstruktor besitzen, der die Liniennummer als Argument bekommt und die Variablen initialisiert – zu Beginn ist die Straßenbahn natürlich leer und nicht in Fahrt. Außerdem soll eine Betriebsbereitschaftsmeldung auf dem Bildschirm erscheinen.

d) Schreiben Sie folgende Methoden und deren beschriebene Funktionalitäten:

`void drive()`	Die Straßenbahn wird in Fahrt versetzt.
`void halt()`	Die Straßenbahn wird gestoppt.
`void getIn(int n)`	Es steigen **n** Passagiere ein.
`void getOut(int n)`	Es steigen **n** Passagiere aus.

Geben Sie in jeder der Methoden auf dem Bildschirm entsprechende Meldungen über den Fahrtmodus der Bahn und die Anzahl der ein- bzw. aussteigenden Passagiere aus. Geben Sie außerdem Fehlermeldungen aus, wenn unsinnige Methoden aufgerufen werden: es sollte nicht möglich sein, dass Passagiere ein- oder aussteigen, während die Straßenbahn fährt, außerdem können **n** Passagiere nur dann aussteigen, wenn so viele in der Bahn sind.

e) Verwenden Sie zum Testen die bereitgestellte Klasse **`TramTest`**. Importieren Sie diese in BlueJ (**Bearbeiten → Klasse aus Datei hinzufügen…** bzw. **Edit → Add Class from File…**). Die darin enthaltene **`main`**-Methode sollte eine Bildschirmausgabe wie diese liefern:

```
Linie 1: Betriebsbereit.
Linie 7: Betriebsbereit.
Linie 1: In Fahrt.
Linie 7: Es sind 5 Passagiere eingestiegen.
Linie 7: In Fahrt.
Linie 7: Aussteigen nicht moeglich - Bahn in Fahrt.
Linie 7: Gestoppt.
Linie 7: Es sind 4 Passagiere ausgestiegen.
Linie 1: Einsteigen nicht moeglich - Bahn in Fahrt.
Linie 1: Gestoppt.
Linie 1: Zu wenig Passagiere in der Bahn - Vorgang abgebrochen.
```