Programmieren I -Schmiedecke

Experiment X4: BallWorld

Dies ist ein Experimetierlabor, in dem Sie in eine vorgegebene Umgebung "hinein programmieren". Die Ablaufkontrolle des Programms liegt diesma nicht in Ihrer Hand: Sie liefern unserer neuen Experimentierumgebung "Ballworld" Klassen, aus denen sie sich Ihre Spielzeuge (Objekte) erzeugt. Damit diese Zusammenarbeit gelingt, gibt es Spielregeln - Interfaces -, an die Sie sich bei der Programmierung Ihrer Klassen halten müssen. - Wie immer in diesem Experimenten müssen Sie viel denken und wenig schreiben, also bereiten Sie sich gut auf das Labor vor....

Lernziele: Klassen und Objekte, Interfaces, Vererbung; Objektverhalten und -interaktion.

Vorbereitung

Lesen Sie den gesamten Aufgabentext und bearbeiten Sie die 7 Vorbereitungsfragen.

Nachbereitung: Laborprotokoll

Halten Sie das Protokoll kurz (ca. 2 Seiten Text zuzüglich Screenshots etc), aber erklären Sie jeweils ihre Programm-Strategie und gehen Sie auf Schwieirgkeiten und Erkenntnisse ein. Wie lange haben Sie für die Experimente gebraucht? Was war schwer zu verstehen?

Zur Erinnerung: Jede Hilfe ist erlaubt (Internet, Kommilitonen, Omas, ...), ***aber Sie müssen sie im Protokoll benennen!***

Einführung: Das Experimentierfeld "BallWorld"

Die Idee von BallWorld ist, ein Umgebung zu bieten, in der sich viele Ball-Objekte unterschiedlicher Bauart "tummeln" können. Allen Objekten gemeinsam ist, dass sie kreisrund sind, aber sie sind Instanzen unterschiedlicher Klassen, d.h. sie sind von unterschiedlicher Bauart. Ihre Aufgabe besteht darin, die verschiedenen "Bauarten" zu implementieren, und zwar als Klassen, die das gemeinsame Interface "Ball" implementieren. In den Klassen programmieren Sie den Startzustand der Objekte und ihr Verhalten, d.h. wie sie sich im Laufe der Zeit oder aufgrund von Interaktionen verändern. **Aber Sie instanziieren Ihre Ball-Objekte nicht selbst!** Das Experimentierfeld entält die main-Methode und übernimmt es, Objekte der Klassen zu erzeugen, anzuzeigen und periodisch zu "aktualisieren". Man sieht dort dann sehr gut, dass gleichartige Objekte, d.h. Instanzen derselben Klasse, sich im Prinzip gleich verhalten, aber ihre eigenen Identität haben: Dadurch, dass sie nicht alle synchron erzeugt werden und unterschiedliche Interaktionen erfahren, "entwickeln" sie sich unterschiedlich.

Wenn Sie die Experimentierumgebung Ballworld starten, erscheint eine grüne rasenähnliche Grundfläche, mehr nicht. Mit File>Launch Ball öffnet sich ein Editierfeld, in dem Sie den Namen einer Ballklasse angeben können. Dann erscheint ein Knopf, mit dem Sie Bälle dieser Klasse

erzeugen und ins Experimentierfeld "schicken" können. Aber diese Ball-Klassen müssen zunächst programmiert werden.

Wie programmiert man eine Ball-Klasse? Eine Ballklasse muss zweierlei definieren: Aussehen und Position des Balls (d.h. das, was BallWorld darstellen soll) und sein Verhalten.

Das Interface Ball

Nach allem, was Sie über Klassen gelernt haben, dürfte klar sein, dass das *Aussehen* durch *Attribute* (oder Felder) der Klasse definiert wird, und das *Verhalten* durch *Methoden*. **BallWorld soll diese Attribute und Methoden, die Sie in Ihrer Ball-Klasse definieren, benutzen, um die aus Ihrer Klasse generierten Ball-Objekte darzustellen.** Das bedeutet, dass es eine Vereinbarung darüber geben muss, wie diese Attribute und Methode "aussehen": Das Interface `Ball`.

Ihre Ball-Klasse muss das **Interface `Ball` implementieren**. Es enthält `get`-Methoden für die Attribute `radius`, `x` und `y` und zwei Verhaltensmethoden, die Ballworld aufruft, wenn der Benutzer mit der Maus in den Ball klickt oder bei angeklicktem Ball eine Taste drückt. Wichtig: diese Methoden *werden gerufen*, d.h. BallWorld ruft sie und "versorgt" sie mit den erforderlichen Parameterwerten, also den Koordinaten des Mausklicks oder dem Zeichen der gedrückten Taste!. Die Methoden userClicked() und userTyped() ruft Ballworld für jeweils das Objekt, das gerade "angeklickt" ist. Hier haben Sie die Gelegenheit, x, y, und radius des Balls zu verändern.

```
public interface Ball {
  public double getX();
  public double getY();
  public double getRadius();
  public void setWorld(World theWorld);
  public void userClicked(double atX, double atY);
  public void userTyped(char key);
}
```

Das Interface World

Natürlich müssen Sie einiges über die Umgebung wissen, in der Ihr Ball operiert, z.B. wie groß das "Spielfeld" ist. Die Umgebung wird durch ein Objekt der Klasse World beschrieben. Wenn Ihr Ball-Objekt in das Spielfeld "geschickt" wird, gibt BallWorld ihm eine Referenz auf dieses World-Objekt mit, und zwar durch Aufruf der Methode `setWorld(World theWorld)`; *ein kluges Ball-Objekt speichert diese **Referenz vom Typ World**.* Das World-Objekt liefert nicht nur die Information über die Größe des Spielfeldes; Sie können auch erfahren, welches andere Ball-Objekt Ihnen gerade am nächsten ist (falls Ihr Ball der einzige im Spiel ist, liefert die Methode `null`!), neue Bälle hinzufügen oder vorhandene aus dem Spiel nehmen.

```
public interface World {
  public double getMinX();
  public double getMinY();
  public double getMaxX();
  public double getMaxY();
  public Ball getClosestBall(Ball fromBall);
  public void addBall(Ball aNewBall);
  public void removeBall(Ball toBeRemoved);
}
```

Das Interface Animate

Schließlich können Sie Ihrem Ball-Objekt auch "Leben einhauchen", indem Sie es *animieren*. Ihr Ball muss zusätzlich das Interface `Animate` implementieren, das nur die eine Methode `act()`

enthält. BallWorld startet einen Prozess, der diese Methode immer wieder aufruft - ganz so, wie Sie es von dem Koordinatenverhalten von Etch-A-Sketch kennen. Ist die act-Methode leer, so definieren Sie passive Bälle, die allenfalls. auf Benutzerinteraktionen reagieren. Programmieren Sie in `act()` eine Positions- oder Größenveränderung, so entstehen Ball-Objekte, die sich von selbst bewegen oder verändern.

```
public interface Animate {
  public void act();
}
```

Vorbereitungsfragen:

1. Sie sollen eine Ball-Klasse namens Basic implementieren. Schreiben Sie den Klassenkopf.
2. Warum darf der Klassenrumpf von Basic nicht leer sein?
3. Versuchen Sie, den Zusammenhang zwischen den Klassen Basic, Ball, World, Animate zu beschreiben und grafisch darzustellen. (Wenn Sie nicht mit dem Computer zeichnen mögen, fotografieren Sie Ihre Papierskizze)).
4. Schreiben Sie den Rumpf von Basic: Basic-Objekte sollen in Position (0,0) sitzen, einen Radius von 5 (Pixeln) haben und nichts tun. (Was passiert, wenn mehrere Basic-Objekte ins Spiel geschickt werden?)
5. Basic braucht keine Attribute. Warum? Legen Sie trotzdem Attribute an, in denen Sie die aktuelle Position, die aktuelle Größe und die World-Umgebung speichern können.
6. Lesen Sie die Experimentieranleitung durch und entwerfen Sie die `act`-Methode und den Konstruktor für die `Ping`-Klasse.
7. Nehmen Sie an, Sie haben eine Ball-Klasse, in der Sie die Umgebung im Feld `theWorld` gespeichert haben. Was könnte bei dem folgenden Codestück schiefgehen? (Sehen Sie sich die Beschreibung von `World` oben noch einmal an!)

```
Ball closest = theWorld.getClosestBall(this);
  double xdist = closest.getX() - this.getX();
```

Laborexperimente

1. **Demo**:
   Laden Sie ballworld.jar herunter, binden Sie es *als Bibliothek* in Eclipse ein und starten Sie es.*Die main-Methode befindet sich in der Klasse ballworld.BallworldGUI*. (Sie können Ballworld auch durch Doppelklick starten.) Sie sehen das rasengrüne Spielfeld. Mit dem Menübefehl File>LoadBall können Sie einige vordefinierte Ballklassen laden, die Namen sehen Sie im Paket ballworld.. Der Effekt von LoadBall ist, dass Sie unter der Spielfläche einen Knopf erhalten, mit dem Sie Bälle dieser Bauart (dieses Typs) erzeugen und ins Spiel schicken können. Spielen Sie ein bisschen damit, um ein Gefühl für BallWorld zu bekommen. Probieren Sie auch das Studentenwerk Planet aus: Wenn Sie einen Planeten anklicken, können Sie ihm mit der +-Taste Planetoiden hizufügen.

2. **Basic:**
   Nun Definieren Sie Ihre vorbereitete Klasse Basic – im eigenen Paket . Sie wissen ja, wie Sie eine Klasse in Eclipse definieren; aber geben Sie jetzt im Class-Wizard gleich die beiden Interfaces an, die Sie definieren möchten. Benutzen Sie den Knopf "Add.." für die Interfaces. Eclipse erzeugt Ihnen so eine Klassenschablone, in der alle zu implementierenden

Methoden angelegt sind. In wenigen Schritten ist dann Basis fertig gestellt.

Kompilieren Sie Ihr Projekt neu und starten Sie es: Nun können Sie die Basic-Klasse laden und mit dem Knopf "Basic" Basic-Objekte ins Spiel schicken . Wieviele Basic-Objekte können Sie unterscheiden? (Haben Sie eine Idee, wie Sie mehr Bälle zu sehen bekommen könnten?) Reagiert Ihr Basic-Objekt auf Mausklicks? (Falls Sie Ihre Basic-Klasse in ein package gepackt haben, müssen Sie sie mit <package>.Basic laden)

3. **Kick:**
   Schreiben Sie nun einen Balltyp Kick, die wie Basic starten soll, aber etwas größer ist (10) und auf Tastatur und Maus reagiert: Immer, Wenn der Benutzer in den Kick-Ball klickt, soll die aktuelle Mausposition zum Kreismittelpunkt werden; jeder nicht genau zentrische Klick bewegt so den Ball ein wenig. Die Eingabe von 'd' soll den Kick-Ball aus dem Spiel entfernen. (Wenn 5 Kick-Bälle im Spiel sind, welcher wird entfernt?).
   Tipp: Sie können den Code von Basic kopieren, aber es gibt eine viel bessere Möglichkeit: Vererbung!

4. **Ping:**
   Definieren Sie jetzt einen animierten Balltyp Ping, der sich gemäß irgendeiner von Ihnen definierten (linaren) Regel über das Spielfeld bewegt, und an den Spielfeldgrenzen reflektiert wird.

   Überlassen Sie doch auch mal etwas dem Zufall: Mit `Math.random()` erhalten Sie Zufallswerte zwischen 0 und 1. Weisen Sie im Konstruktor den Startkoordinaten und dem Inkrement von x und y (dem Wert, um den sich der Koordinatenwert bei jedem Tick, d.h. jeder Ausführung von act() verändert) Zufallswerte zu. Jetzt verhalten sich alle Ping-Objekte verschieden.
   Tipp: mit `Math.random()-0.5` erhalten Sie Werte zwischen -0,5 und +0,5 - d.h. verschiedene Bewegungsrichtungen.

5. **Bang:**
   Schafffen Sie jetzt einen explodierenden Balltyp `Bang`, der wie Ping über das Spielfeld wandert, aber dabei langsam dicker wird (z.B: 0.03 pro Tick). Lassen Sie ihn mit einem Radius von 2 starten; bei 5 "explodiert" er, d.h. er verscchwindet vom Spielfeld.

   Ab hier für Schnelle und Experten – oder neue Fans...

6. **Feuerwerk:**
   Wenn das funktioniert, lassen Sie den Ball eine Kettenreaktion starten: Vor der Explosion schickt er zwei neue `Bang`-Bälle ins Spielfeld. Sie müssen dazu zwei `Bang`-Objekte erzeugen und mit der `World`-Methode `addBall()` dem Spielfeld hinzufügen. Damit das ganze gut aussieht, sollten die neuen `Bang`-Bälle dort starten, wo der andere Ball explodiert. Dazu benötigen Sie noch einen weiteren Kostruktor, dem Sie die Startkoordinaten mitgeben:
   `public Bang(double startX, double startY)`

7. **Weitere Experimente:**
   Jetzt haben Sie einiges an Material, mit dem Sie weiter experimentieren können. Sie lernen dabei sehr viel Programmieren!
   Hier ein paar Anregungen:
   - Lassen Sie die Explosionen enden, wenn Sie 50 Bälle im Spielfeld haben: Ab dann schrumpfen die Bälle von 5 auf 0 und verschwinden.
   - Ahmen Sie ein Feuerwerk nach, indem Sie einen kleinen Ball eine ballistische Kurve hinaufschießen und in geeigneter Höhe als Sternenregen explodieren lassen: 30 `Ball`-Objekte, die in eine zufällige Richtung fliegen und dabei imer kleiner werden (bitte nicht alle einzeln erzeugen, sondern eine Schleife verwenden!!).
   - Wer's schafft, kann versuchen, Bälle auf Ihre Nachbarn reagieren zu lassen: z.B. Jeden Ball fressen, der bis auf 1 Pixel herankommt (da ist es dann Zufall, wer wen frisst), oder

beim Zusammenstoß die Richtung ändern, oder beim Zusammenstoß explodieren... (Das riecht dann schon ein bisschen nach Videospiel...)

Hinweis zum Copyright