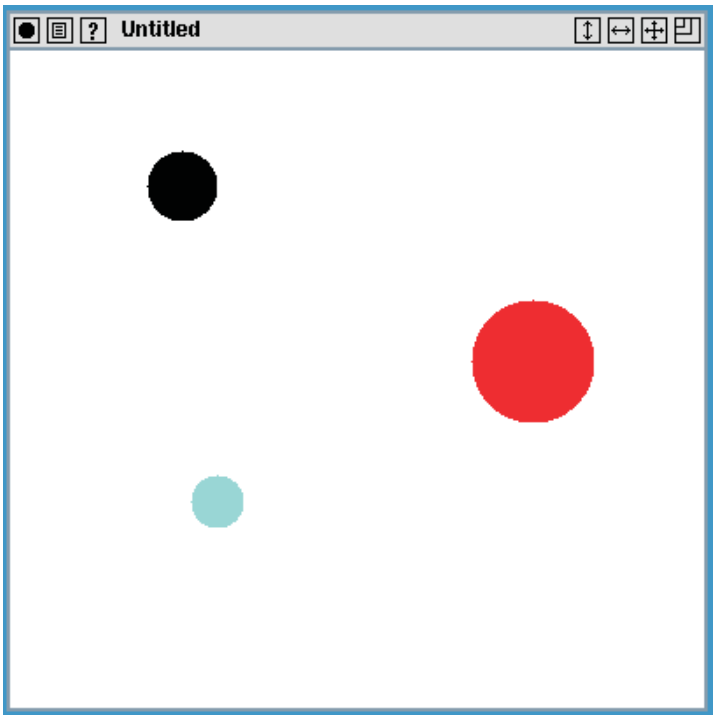
Balls in a box

September 11, 2009

Introduction

The object-oriented programming (OOP) approach will be further explored in this project. An object-oriented program is made up of a set of cooperating objects. The objects are abstract models of the "real" objects we wish to simulate. Each object has the capability of processing data as well as communicating and interacting with other objects.

The purpose of this project is to explore and extend a set of Java classes. The program that we will work with simulates balls in a box. The box consists of a window on the screen in which balls of various colors live and can be moved, resized and recolored. A snapshot of a box with three balls can look like this:



The program consists of three classes. The balls are instances of class `Ball` while the box is an instance of the class `Box`. The main program is a class called `BallsinBoxViewer`.

Explore the Classes

Look at the code in the class `Ball.java`. Try to find the answers to the following questions.

- The class has several instance fields. What are their names and types and what do they do?
- In what method does the actual drawing of the balls take place?
- Which objects do we wish to model with the different classes?
- Does the class have a constructor?
- How many methods does the class have (name them)?

Answer the same questions for the `Box` class.

Run the Program

Compile all three files in turn with the `javac` command. Run the viewer program by the command `java BallsinBoxViewer`. You should get a similar result as given in the figure above.

The execution of a java program always starts in the `main` method, which is required for the java interpreter to be able to run. The first thing that happens in the program is that we create a `JFrame` object, and we set a size for the object. The command

```
Box box = new Box();
```

creates a `Box` object called `box`. As you probably already know, you create an object from a class by calling a constructor defined in the class. In a similar way we create black, red and cyan colored balls by the following commands.

```
Box box = new Box();
Ball blackBall = new Ball(box,100,90);

Ball redBall = new Ball(box,270,200);
redBall.setColor(Color.RED);
redBall.setRadius(100);

Ball cyanBall = new Ball(box,50,350);
cyanBall.setColor(Color.CYAN);
cyanBall.setRadius(10);
```

Take a look again at the constructor for the `Ball` class. Describe the parameters as used above. Why will the black ball be black (we didn't state the color)? What radius will the black ball have?

Create two more balls in the box. Try to make one of them into a non-standard color (look in the API at the different constructors in the `Color` class). Place one of the balls in the corner of the box, what happens? What will happen if you place two balls in the same place? Try to place a ball outside the box (what is the size of the box?)

Move the Balls

To move a `Ball` object we can use a method called `step`. Take a look at the method implemented in the `Ball` class. A call to the method moves the ball on the x and y axes by given step sizes, as given in the method `setStepSizes`.

Set different step sizes for all the balls in the program (also try negative step sizes). When using the `step` function the balls are moved very quickly, so in order to see the balls move me must use a delay. This is implemented in the method `moveAllBalls`. Look up the parameters for the method and apply it to the balls in the box. For example, what happens if you at the end of the viewer class put the following line (don't forget to set the step size first!)?

```
box.moveAllBalls(5,30);
```

Improve the program

We will continue to work with the three classes `Ball`, `Box` and `BallsinBoxViewer` during the coming weeks. Can you think of something to improve the performance of the program?