BouncingBall.java

```java
/*************************************************************************
 *  Compilation:  javac BouncingBall.java
 *  Execution:    java BouncingBall
 *  Dependencies: StdDraw.java
 *  Implementation of a 2-d bouncing ball in the box from (-1, -1) to (1, 1).
 *
 *  This code is already complete. You will make an object-oriented version
 *  of it as described in the next file.
 *
 *  The version differs from the one in Booksite 1.5 in the initial position
 *  of the ball, and random factors for the velocity and radius.
 *************************************************************************/
public class BouncingBall {
    public static void main(String[] args) {

        // set the scale of the coordinate system
        StdDraw.setXscale(-1.0, 1.0);
        StdDraw.setYscale(-1.0, 1.0);

        // initial values, random velocity and size
        double rx = 0.0, ry = 0.0;                          // position
        double vx = 0.015 - Math.random() * 0.03;      // x velocity
        double vy = 0.015 - Math.random() * 0.03;      // y velocity
        double radius = 0.025 + Math.random() * 0.05; // size

        // main animation loop
        while (true)  {
            // bounce off wall according to law of elastic collision
            if (Math.abs(rx + vx) > 1.0 - radius) vx = -vx;
            if (Math.abs(ry + vy) > 1.0 - radius) vy = -vy;

            // update position
            rx = rx + vx;
            ry = ry + vy;

            // clear the background
            StdDraw.setPenColor(StdDraw.GRAY);
            StdDraw.filledSquare(0, 0, 1.0);

            // draw ball on the screen
            StdDraw.setPenColor(StdDraw.BLACK);
            StdDraw.filledCircle(rx, ry, radius);

            // display and pause for 20 ms
            StdDraw.show(20);
        }
    }
}
```

Ball.java

```
/**************************************************************
Create an object-oriented version of BouncingBall.java that
is capable of simulating any number of Ball instances. The first
program should define the following API:

public class Ball
-----------------------------------------------------------------
Ball()        create a ball at (0,0), random velocity, random size
void draw()   draw ball at current position
void move()   move using velocity and unit time increment

The second program (shown separately) will be a client
BouncingBalls that takes a command-line argument N and
creates/animates N bouncing balls.

 *************************************************************
 *  Compilation:  javac Ball.java
 *  Execution:    java Ball
 *  Dependencies: StdDraw.java
 *
 *  Object oriented implementation of a 2-d Ball, Booksite 3.4
 *************************************************************/

public class Ball {

   // declare instance variables
   _____________________________________________   // position
   _____________________________________________   // velocity
   _____________________________________________   // radius

   // other instance variables? up to you

   // constructor
   public Ball() {
      // always start ball position at (0, 0)



      // initial velocity and size generated randomly




   }


   // draw the ball, but not the background
   public void draw() {


   }
   // bounce off vertical wall by reflecting x-velocity
   private void bounceOffVerticalWall() {

   }

   // bounce off horizontal wall by reflecting y-velocity
   private void bounceOffHorizontalWall() {

   }


   // move the ball one step, but don't draw it
   public void move() {
      // bounce off wall(s) if you are near the border



      // update position using unit change in time



   }

   // test client to create and animate just 2 balls.
   // this part is already complete.

   public static void main(String[] args) {
      // create and initialize two balls
      Ball b1 = new Ball();
      Ball b2 = new Ball();

      // animate them
      StdDraw.setXscale(-1.0, +1.0);
      StdDraw.setYscale(-1.0, +1.0);
      while (true) {
         StdDraw.setPenColor(StdDraw.GRAY);
         StdDraw.filledSquare(0.0, 0.0, 1.0);
         StdDraw.setPenColor(StdDraw.BLACK);
         b1.move();
         b2.move();
         b1.draw();
         b2.draw();
         StdDraw.show(20);
      }
   }
}
```

BouncingBalls.java

```
/*************************************************************
 *  Compilation:  javac BouncingBalls.java
 *  Execution:    java BouncingBalls N
 *  Dependencies: Ball.java StdDraw.java
 *  Booksite 3.4
 *  Client to create and animate an array of N bouncing balls
 *************************************************************/

public class BouncingBalls {
   public static void main(String[] args) {

      // number of bouncing balls from command-line argument
      int N = ________________________________________(args[0]);

      // Set window coordinates between -1 and +1
      StdDraw.setXscale(-1.0, 1.0);
      StdDraw.setYscale(-1.0, 1.0);

      // create an array of N random balls
      Ball[] balls = ________________________________________
      for (int i = 0; i < N; i++)
         balls[i] = ________________________________________

      // do the animation loop
      while(true) {
         // Gray Background
         StdDraw.setPenColor(StdDraw.GRAY);
         StdDraw.filledSquare(0.0, 0.0, 1.0);

         // draw N black balls
         StdDraw.setPenColor(StdDraw.BLACK);
         for (int i = 0; i < ____; i++) {
            _____________________________________
            _____________________________________
         }
         StdDraw.show(20);
      }
   }
}

/**********************************************************************
Recommended Book Exercises: 3.2.5, 3.2.11 (Point.java code on Booksite)
**********************************************************************/
```