PROGRESSING TO ARRAYLISTS

(Part 1)

In class, we've seen how to make a ball bounce on the screen.

If you recall, from the Bounce.java program, the ball had certain properties that we stored in variables.

```
int ballx = 200;                    // x-position of ball
int bally = 100;                    // y-position of ball
int size = 20;                      // size of ball
int xspeed = 3;                     // x-speed of ball
int yspeed = 2;                     // y-yspeed of ball
Color ballColour = Color.RED;       // colour of ball
```

All numbers above are in pixels.

Note:

I sometimes call the speeds, **speedx** and **speedy** (I'm not consistent). Most often though, I use **vx** and **vy** because that's shortest and still make sense (if you remember that v=velocity)

We didn't really have a colour variable for our ball. We just chose a colour in the drawGraphics() method. But if we want to make balls that have different colours, then colour must be a property of the ball.

We need to have 6 variables to describe one ball.
If we wanted another ball, we'd need another 6 variables.
A third ball would mean that we'd have 3x6 = 18 ball variables.
This is pretty much impossible to work with. *What do we do?*

We make a Ball object. In this object we put all of the ball properties that we want. (We can also put methods to make the ball do things, but that's for later.)

```
class Ball {
        int x;                      // renamed from ballx
        int y;
        int size = 20;              // default size of ball
        int xspeed = 3;             // default x-speed of ball
        int yspeed = 2;             // default y-yspeed of ball
        Color clr = Color.RED;      // default colour of ball
}
```

So now we just have 1 variable:

Ball **b1** = new Ball();

And we can access all of the variables inside the ball by saying

```
        b1.x = 250;
        gc.setColor(b1.clr);        //use the ball colour
```

We can set all of the ball variables like this.

If we have two balls

        Ball **b2** = new Ball();

then we can do the same thing, using the variables inside b2**:**

```
        b2.x = 125;
        b2.y = 456;

        b2.x += b2.speed;
```

BUT there's an easier way: set the variables WHEN you make the ball.

Our ball class is the same, but we add something called a CONSTRUCTOR. It looks like a method, but doesn't have void in front of it, and it is the same name as the class.

```
class Ball {
        //ball variables
        int x, y;
        int size = 20;
        int xspeed = 3;
        int yspeed = 2;
        Color clr = Color.RED;

        //constructor plus local variables used in constructor
        Ball(int x, int y, int size, int vx, int vy, int colour) {
                this.x = x;
                this.y = y;
                this.size = size;
                xspeed = vx;
                yspeed = vy;
                clr = colour;
        }
}
```

If the local variable in the constructor is the same as the Ball variable, then you need to put "this" in front of the Ball variable.
Do some research to find out why this is.

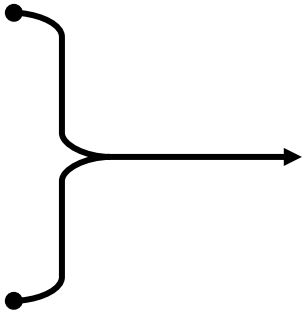
When we make the ball, we send in the values of the variables that we want it to be created with. Here, have a look:

```
Ball b1 = new Ball(100, 400, 30, 3, 3, Color.BLUE);

gc.drawOval(b1.x, b1.y, b1.size, b1.size);
```

Thus we have gone from 6 variables down to 1, with the variables all inside b1

```
int ballx;
int bally;
int size;
int xspeed;
int yspeed;
Color ballColour;
```



Ball b1;

<continued in next document>