COS126 Using Objects - 3.1

- Here is an API (Application Programming Interface) for charged particles.

```
public class Charge
--------------------------------------------------------------------------
       Charge(double x0, double y0, double q0)  // location and charge
double potentialAt(double x, double y)          // potential at (x,y) due to charge
String toString()                               // string representation
```

- Write a program that takes a **double** value `r` from the command line, creates four **Charge** objects that are each distance `r` from the center of the screen (.5, .5), and prints the potential at location (.25, .5) due to the combined four charges. All four charges should have the same unit charge.

```
 1  /*  Book Exercise 3.1.1 to create a Charge client */
 2  public class FourChargeClient {
 3     public static void main(String[] args) {
 4        // read in distance r from command line
 5        double r = Double.parseDouble(args[0]);
 6        // center of standard drawing window
 7        double cx = 0.5;
 8        double cy = 0.5;
 9
10        // construct four charges
11        Charge c1 = new Charge(cx + r, cy,     1.0);     // east
12        Charge c2 =                                      // south
13        Charge c3 =                                      // west
14        Charge c4 =                                      // north
15
16        // Compute potentials at (.25, .5)
17        double px = 0.25;
18        double py = 0.5;
19        double v1 = c1.potentialAt(              );
20        double v2 =
21        double v3 =
22        double v4 =
23
24        // compute and output total potential
25        double sum =
26        System.out.println("total potential = " + sum);
27     }
28  }
```

- Recommended Book Exercises: 3.1.13 (Booksite 3.1.25), 3.1.14 (Booksite 3.1.30).

  Booksite Exercise 3.1.14 (answer in Book on page 340)

- Look up the **Picture** API in Section 3.1 (Book p. 330 or Booksite).
- Exercise 3.1.6 (Booksite 3.1.60) Write a program that takes the name of a picture file as a command-line input, and creates three images, one that contains only the red components, one for green, and one for blue.

```
 1 /*************************************************************************
 2  *  Compilation:  javac ColorSeparation.java
 3  *  Execution:    java ColorSeparation filename
 4  *  Dependencies: Picture.java
 5  *  Reads in an image from a file, and displays the red, green, and
 6  *  blue portions in three separate windows.
 7  *************************************************************************/
 8
 9 import java.awt.Color;
10
11 public class ColorSeparation {
12    public static void main(String[] args) {
13       // read in the picture specified by command-line argument
14       Picture pic = new Picture(args[0]);
15       int width  = pic.width();
16       int height = pic.height();
17
18       // create three empty pictures of the same dimension
19       Picture R = new Picture(width, height);   // R
20       Picture G =                               // G
21                                                 // B
22
23       // separate colors
24       for (int x = 0; x < width; x++) {
25          for (int y = 0; y < height; y++) {
26             // color value of current pixel
27             Color c = pic.
28
29             int r = c.getRed();  // r
30             int g =              // g
31                                  // b
32
33             R.set(x, y, new Color(r, 0, 0));  // R
34             G                                 // G
35                                               // B
36          }
37       }
38
39       // display each one in its own window
40       R.show();    // R
41       G.           // G
42                    // B
43    }
44 }
```