Chapter 7 – Classes & Objects, Part D

These notes present some details about the Random and Date classes, as well as simple interactive GUI constructions.

The *Random* Class

1. Math.random() is useful; however, every time we use it, we get a different (random) value. In the end, this is usually what we want. However, it is hard to debug a program when different code segments are called based on a random number. In other words, the code behaves differently, depending on the random value. To debug a particular code segment, we have to run the program over and over until it randomly hits that segment again. This approach is fraught with even more problems. At least for debugging, what we need is to be able to do is reproduce the exact same sequence of random numbers each time a program runs. We call this *replicating* a set of random numbers. In other words, to be able to generate 10 random values, and when the program runs again, the same 10 random values will be generated. To do this, we will use the Random class and set the *seed* of the random number generator. Consider this code:

```
Random r = new Random();

for( int i=0; i<10; i++ )
      System.out.printf( "%.4f ", r.nextDouble() );

r = new Random();

for( int i=0; i<10; i++ )
      System.out.printf( "%.4f ", r.nextDouble() );


r = new Random(723947682);

for( int i=0; i<10; i++ )
      System.out.printf( "%.4f ", r.nextDouble() );

r = new Random(723947682);

for( int i=0; i<10; i++ )
      System.out.printf( "%.4f ", r.nextDouble() );
```

Which will display:

```
0.1839 0.0853 0.9382 0.0910 0.7460 0.3344 0.6356 0.0746 0.1718 0.0673
0.6909 0.7438 0.3007 0.9586 0.0050 0.5335 0.3716 0.8235 0.5412 0.5759

0.8480 0.6624 0.1562 0.4647 0.5716 0.5684 0.6932 0.9385 0.9671 0.3704
0.8480 0.6624 0.1562 0.4647 0.5716 0.5684 0.6932 0.9385 0.9671 0.3704
```

2. There are several type of random values that are useful. For instance a random float between 0 and 1, a random float between *a* and *b*, and a random integer between *x* and *y*, inclusive. This code shows examples of these:

```
// U(0,1), nextDouble(), nextFloat()
for( int i=0; i<8; i++ )
      System.out.printf(  "%.4f ", r.nextDouble() );
System.out.println();

// U(a,b), nextDouble(), nextFloat()
double a = 5.0, b=20.0;
for( int i=0; i<8; i++ )
{
      double rand =  a + r.nextDouble()*(b-a);
      System.out.printf(  "%.4f ", rand );
}
System.out.println();

// integer U(x,y)
int x = 5, y = 15;
for( int i=0; i<10; i++ )
{
      int rand = x + r.nextInt( y-x+1 );

      System.out.printf( "%d ", rand );
}
System.out.println();
```

3. The Random class also allows us to create random booleans:

```
for( int i=0; i<10; i++ )
{
      System.out.printf( "%b ", r.nextBoolean() );
}
```

This statement generates *true* with probability 0.5 and *false* with probability 0.5. If you want to true's and false's with different probabilities then use a different technique. Suppose that we want to generate *true* with probability *p*. Then we can generate a value between 0 and 1 and if it less than *p*, then generate *true*; otherwise, we generate *false*.

```
double trueProb = 0.3;

for( int i=0; i<10; i++ )
{
      boolean rand = r.nextDouble() < trueProb ? true : false;

      System.out.printf( "%b ", rand );
}
```

The *Date* Class

Sometimes it is useful to print out the date and time of your output. This can be accomplished by using the System and Date classes.

```
Date d = new Date( System.currentTimeMillis() );

System.out.println( d.toString() );
```

which displays something like this:

```
Mon Oct 13 15:06:12 EDT 2008
```

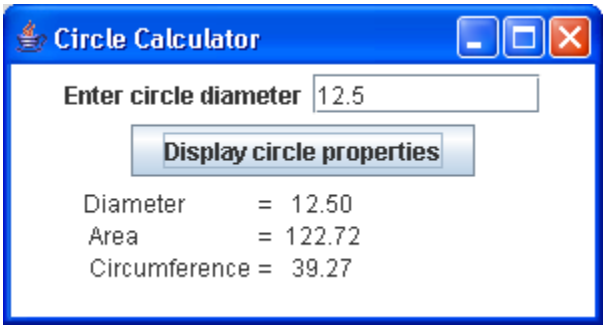
You might also want to time your program:

```
import java.util.*;

public class DateStuff
{
   public static void main(String[] args)
   {
      Date dBegin = new Date( System.currentTimeMillis() );

      for( long i=0; i<10000000; i++ )
      {
         double x = Math.sqrt( Math.pow( i, 0.25 ) );
      }

      Date dEnd = new Date( System.currentTimeMillis() );

      long bTime = dBegin.getTime();
      long eTime = dEnd.getTime();
      double seconds = (eTime-bTime)/1000.0;

      System.out.println( "Run Begin : " + dBegin.toString() );
      System.out.println( "Run End   : " + dEnd.toString() );
      System.out.println( "----------------------------------------" );
      System.out.printf(  "Total Time: %.2f sec.\n", seconds );
   }
}
```

It takes a good bit more work to do anything with Dates. For instance, you can put a Date in a Calendar and use it to get the day of the week, *etc*. Similar, you can put a Date object into a DateFormat object to obtain similar types of things. If you want to calculate the difference between two dates in days (or weeks and days, etc), you must write your own code.

Displaying GUI Components

1. Next, we consider building a simple GUI application that looks like this:



The code is below and is a bit detailed. You could use it as boiler-plate code, for now, to do very simple things. A detailed understanding of a number of things would be required to go much further with GUI.

```
import java.awt.event.*;
import javax.swing.*;

public class guiExample
{
   public static void main(String[] args)
   {
      // Create GUI controls.
      JLabel lab = new JLabel("Enter circle diameter");
      final JTextField txtDiameter = new JTextField(10);
      final JTextArea txaResults = new JTextArea("Results:", 3, 20);
      JButton btnCreateCircle = new JButton("Display circle properties");

      // Add button's ability to respond when the button is pressed.
      // Specify the code for the event handler.
      btnCreateCircle.addActionListener(

         new ActionListener() // anonymous inner class
         {
            public void actionPerformed( ActionEvent event )
            {
               // Get the diameter from the text box.
               double diameter = Double.parseDouble(txtDiameter.getText());
               // Create a circle object
               Circle circle = new Circle( diameter );
               // Build an output string.
               String msg = String.format( "Diameter\t= %7.2f\n " +
                                           "Area\t= %7.2f\n " +
                                           "Circumference\t= %7.2f",
                                           circle.getDiameter(),
                                           circle.getArea(),
                                           circle.getCircumference() );
               // Display output string.
               txaResults.setText( msg );
            } // end method actionPerformed
         } // end anonymous inner class
      ); // end call to addActionListener
```

```
        // Create a Panel
        JPanel panel = new JPanel();

        // Add GUI controls to Panel.
        panel.add(lab);
        panel.add(txtDiameter);
        panel.add(btnCreateCircle);
        panel.add(txaResults);

        // Create the Frame.
        JFrame frame = new JFrame("Circle Calculator");
        // Add Panel to Frame.
        frame.add( panel );

        // Set Frame properties.
        frame.setSize( 300, 160 );
        frame.setLocation( 500, 300 );
        frame.setDefaultCloseOperation( JFrame.EXIT_ON_CLOSE );
        // Display the Frame.
        frame.setVisible(true);
    }
}

class Circle
{
   private double diameter;

   public Circle( double diameter )
   {
      this.diameter = diameter;
   }

   public double getDiameter() { return diameter; }

   public double getArea()
   {
      return Math.pow( diameter/2, 2 ) * Math.PI;
   }

   public double getCircumference()
   {
      return 2 * Math.PI * ( diameter/2 );
   }

}
```