Programming Project 01

DUE Feb. 15. 10pt Early Bird if in by Feb 8.

Make (that means figure out what it should do, find out how, begin to write in Java, compile, fix syntax, debug, and improve LITTLE BY LITTLE): A JAVA PROGRAM THAT COMMANDS A TURTLE TO DRAW STUFF LIKE: (Upload a compilable, runnable .java file that does it!)

Five different sizes of the SAME figure! (You design how the sizes vary!)

30 pts for effective use of parametrized methods.

One near the CENTER, the other four near the corners.

You choose the letters:

I, L, J Unacceptable: too easy, done in class already!

First letter for up to 10 points: (single stroke):

C G M N S U V W Z

10 points for one of these.

Second letter, must be separated from first with space up to 40 points:

10 pts for the space between the 2 letters.

plus:

10 pts for a single-stroke letter, different from your first.

or

20 pts for a multi-stroke letter, like:

A B E F H K P Q R T X Y

or

30 pts for a curved letter, you'll have to jump ahead and code loops to approximate a curve by a sequence of short straight lines.

Multi-color and width bonus: 10pts. Use different colors and widths for different parts of each letter. Tip: Read SimpleTurtle documentation!
Use these in the body of main:

```java
World myWorld;   declaration (getting another ticket on your desk).
myWorld = new World( );   new operation, constructing one World object,
                        assignment (copying data returned from new onto
                        the ticket named myWorld).
Turtle mt;       another declaration (so you now have 2 tickets).
mt = new Turtle( myWorld ); see above, except the ticket is named mt.
    The Turtle builder must know which World holds
    the Turtle.

Only one World and one Turtle need be used in this project!

mt.penUp(); method call statement, select the penUp method of the Turtle
and perform the call operation. Moving a Turtle with its
pen up makes no trace.

mt.penDown(); Moving a Turtle with its pen down draws a line.
You need penDown and penUp in the letter-drawing methods too,
for letters like X that require 2 or more pen strokes.

mt.moveTo( xxx, yyy ); Makes the Turtle move to a spot with given x,y location

mt.setPenWidth( nn ); Sets the width of future lines the Turtle will draw.

mt.setPenColor( new java.awt.Color( 255, 0, 0) ); (1) make a Color object with
    the brightest red possible. (2) make the Turtle use that Color.

mt.drawL( 2.0 ); Call operation, calling the drawL method we coded into
    Turtle.java. With that, Turtle can draw one or more letter L

Within the Turtle.java file, say between the last } of the main method and the
last } in the class definition (just left of the last } in the file, define
the drawL method:

```java
public void drawL( double scale )
{
    this.setHeading( 0.0 );  this, for the method's code, refers to the same Turtle
    that mt refers to for main's code, since main
    used mt.drawL( 2.0 );
    setHeading( 0.0 ); makes the Turtle point up.
    this.turn( 180.0 ); turn makes a Turtle change its direction.
    Experiment with direction values to get proper letters.
    this.forward( (int) (100 * scale) );
    100 a number you figure out to control the length of a line.
    * is the operator telling the computer to multiply.
    scale is the name of the spot on the calling card where the scale factor is wri
    (100 * scale) The parentheses ( ) make the multiply be done FIRST.
    (int) drops the digits after the decimal point, converts the floating point
    (decimal with a possible fraction) number to an Java int. G&E
    programmed their forward method to only accept int distances.
}