Computer Science Fall 2012
Lab Test
Oct. 22-26, 2012

You will write the lab test during your scheduled laboratory period. Only students registered in the lab will be allowed to write at that time without prior approval.

At your lab test you will be randomly assigned **two** of the following questions, one from Part A and one from Part B. To prepare, you should become familiar with all problems by coding them, and subsequently executing them.

During the lab test, you will not be allowed any electronic media (diskettes, USB drives etc), personal notes, or network access. You will log on to a special test userid and you will not have access to your `Z:` drive. You may bring a pen or pencil and paper will be supplied if needed.

In the question script you will be given specifications, including what methods to write and the required parameters. These should be followed and implemented for full marks. In each case, you will need to write the remaining code (i.e. constructor and other methods as needed) to allow the program to run. Solutions that produce the required result but do not follow the requirements will be penalized. Part marks are awarded for correct partial implementation (e.g. only some of the required methods, not using methods where required, not using parameters as required).

You do not need to comment your code. Execution efficiency is not a concern, however proper use of the Java constructs we have discussed is expected.

You will be using the DrJava development environment. To aid you we will provide a template for Java classes using TurtleGraphics and a summary of `Math`, `Turtle` and `TurtleDisplayer` methods as attached here.

COSC 1P02
Class Template

```
package PackageName;


import Media.*;                   // for displayer and turtle
import java.awt.*;                // for Color class
import static java.lang.Math.*;  // for math constants and functions & random
import static Media.Turtle.*;    // for turtle speed constants
import static java.awt.Color.*;  // for Color constants


public class ClassName {


    // Instance variable declarations


    public ClassName ( ) {  // constructor


    };  // constructor


    // Method declarations


    public static void main ( String[] args ) {
                                               ClassName v = new ClassName();
                                               v.methodName(); };

}
```

COSC 1P02
Method Summary

`TurtleDisplayer`

method	meaning
`d = new TurtleDisplayer()`	*constructor*: creates a turtle displayer object
`d.close()`	wait until user presses `Close` button and close displayer
`d.placeTurtle(turtle)`	place `turtle` on the canvas
`d.waitForUser()`	wait until user presses `OK` before continuing

`Turtle`

method	meaning
`SLOW`, `MEDIUM`, `FAST`	*constants*: turtle speeds
`t = new Turtle()`	*constructor*: creates a turtle object
`t = new Turtle(speed)`	*constructor*: creates a turtle object drawing at `speed`.
`t.backward(units)`	move backwards `units` drawing units
`t.forward(units)`	move forward `units` drawing units
`t.left(radians)`	turn left radians `radians`
`t.moveTo(x,y)`	move turtle to coordinates `(x,y)`
`t.penDown()`	place the pen on the canvas
`t.penUp()`	raise the pen from the canvas
`t.right(radians)`	turn right radians `radians`
`t.setPenColor(color)`	change the turtle's pen color to `color`
`t.setPenWidth(width)`	change the turtle's pen width to `width`

`Math`

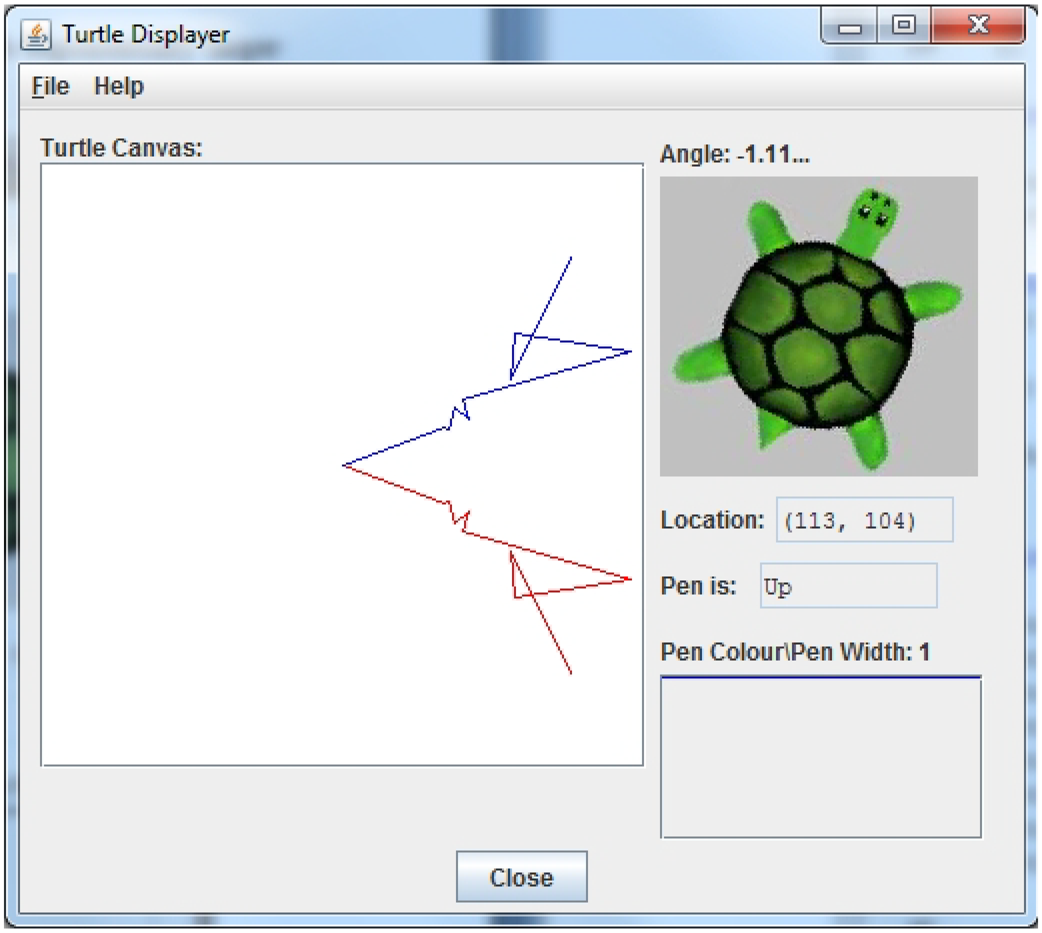
Method/Value	meaning
`E`	*constant*: the mathematical constant `e`
`PI`	*constant*: the mathematical constant π
`v = abs(x)`	returns the absolute value of `x`
`v = acos(x)`	returns the arc cosine of `x`
`v = asin(x)`	returns the arc sine of `x`
`v = atan(x)`	returns the arc tangent of `x`
`v = cos(x)`	returns the cosine of `x`
`v = log(x)`	returns the natural logarithm of `x`
`v = pow(a,b)`	returns a^b
`v = random()`	returns a random value between `0.0` and `1.0`
`v = sin(x)`	returns the sine of `x`
`v = sqrt(x)`	returns the square root of `x`
`v = tan(x)`	returns the tangent of `x`

`Color`

method	meaning
`red, green, …, RED, GREEN, …`	*constants*: color constants
`d = new Color(value)`	*constructor*: creates one of 16,777,216 colors

Part A
Question 1

Write a Java program to a mirrored random walk such as:



Angle: -1.11...

Location: (113, 104)

Pen is: Up

Pen Colour\Pen Width: 1

A random walk can be produced by selecting a random angle between 0 and 2π and a random distance between 0 and 100 and then drawing a line of that length at that angle from the turtle's current position.

The mirroring is accomplished by having two turtles drawing on the same displayer in two different colors. They draw lines of the same length, however while one turns left, the other turns right.

In preparing your solution, write a method:

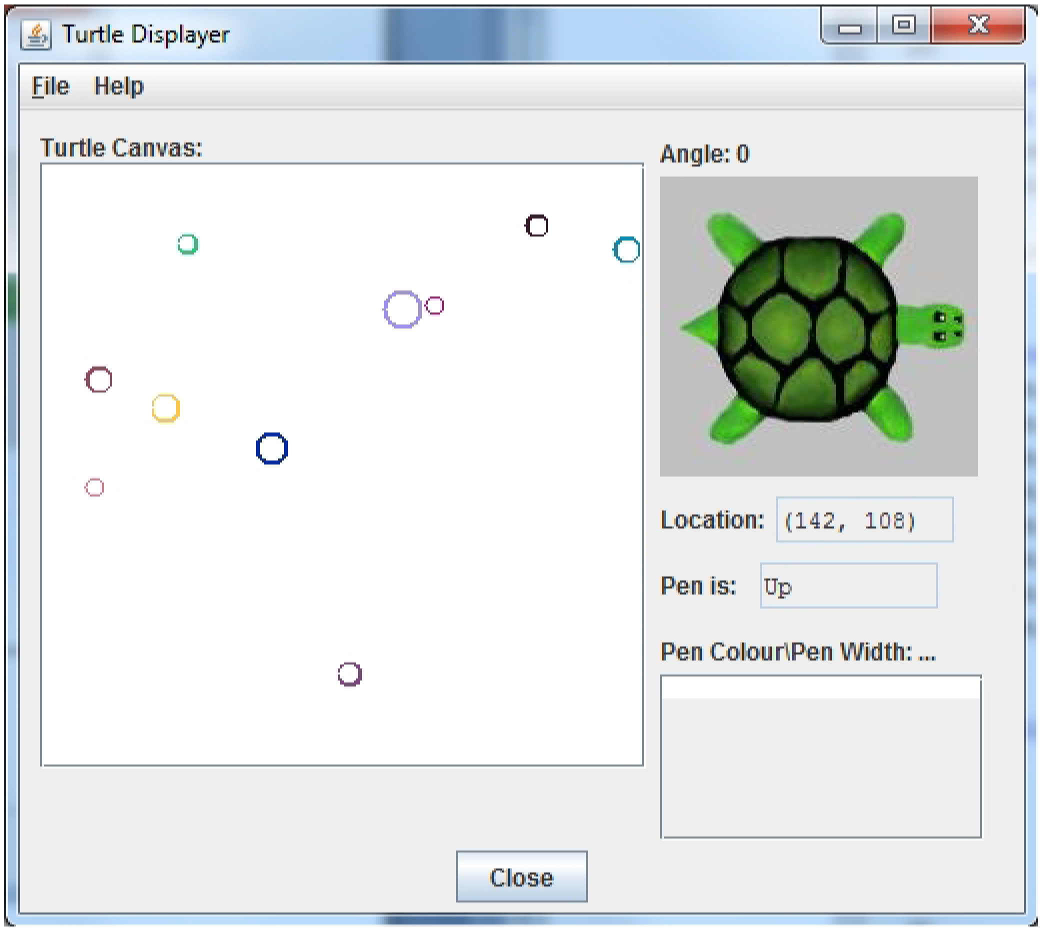
```
    private void walk ( int steps ) { …
```

which draws a mirrored random walk consisting of `steps` steps starting from the turtles' current positions.

To run your program for submission, draw a mirrored walk of 10 steps with one turtle in `red` and the other in `blue`.

Part A
Question 2

Write a Java program to draw bubbles as shown below:



The program should draw bubbles centered at random points `(x,y)` on the canvas. Each bubble should be a random diameter from 10 to 20 pixels drawn in a random color.

In preparing your solution, write a method:

```
   private void drawBubble ( int diameter ) { …
```

which draws a bubble centered on the turtle's current position. The color should be chosen at random from the `16,777,216` possible colors (use the `Color` constructor). The bubble can be drawn by drawing a dot of specified diameter in the random color and then drawing a dot whose diameter is 80% of the specified diameter in `white` at the same position. A dot can be drawn by setting the pen width and drawing a line of length `0`.

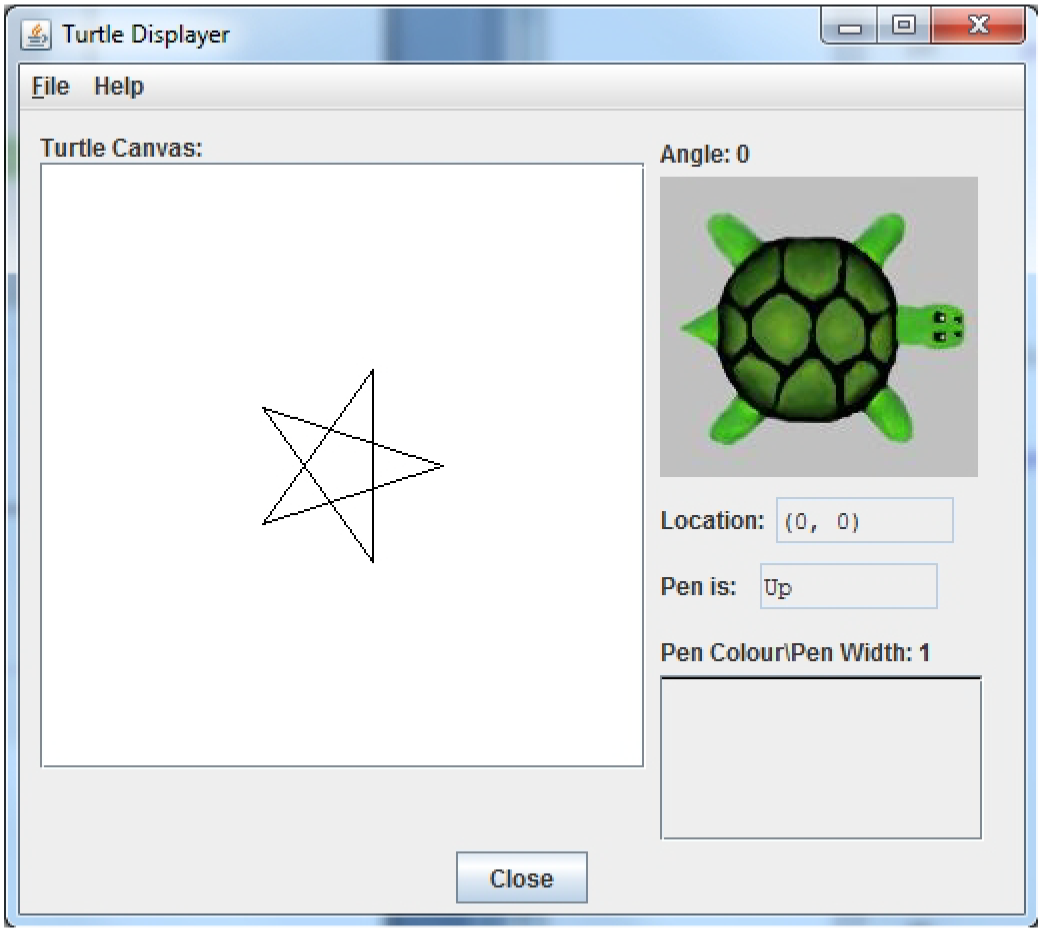
To run your program for submission, draw a series of 10 bubbles. The display is 300x300 with `(0,0)` being the center. Some part of some bubbles may be off the display.

Part A
Question 3

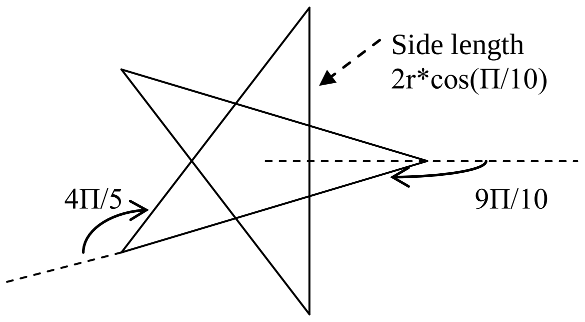
Write a Java method:

```
     private void drawPentagram ( double radius ) { …
```

which draws a pentagram (five-pointed star) with specified radius centered on the turtle position as shown below:



A pentagram has the following geometry:

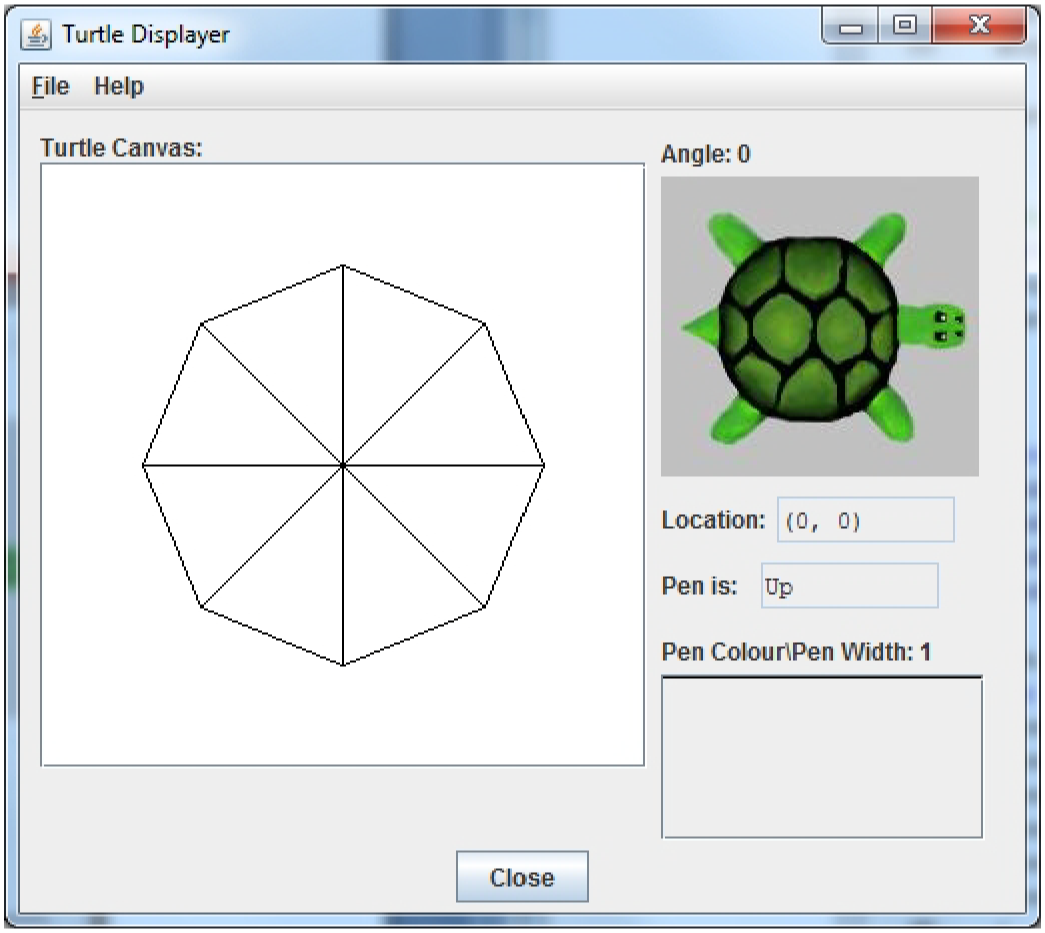


The radius (r) is the distance from the center of the star to the point.

For submission, run your program to draw a pentagram with radius 50 centered on the canvas.

Part B
Question 1

Write a Java program to draw a wedged octagon as shown below:



An octagon of radius `r` is drawn as eight isosceles triangles with common vertex (the center of the octagon). The triangles have two sides the same length—the radius `r`—and the third side (the outer side of the octagon) of length 2 r sin π/8. The angle at the center is π/4 and the base angles are 3π/8, making the exterior angles 5π/8.

In preparing your solution, write a method:

```
     private void drawTriangle ( int r ) { …
```

which draws an isosceles triangle with two sides length `r` as described above with the small angle (π/4) at the turtle's current position and the first side in the turtle's current direction. It leaves the turtle in the same orientation as it started.

Also write a method:

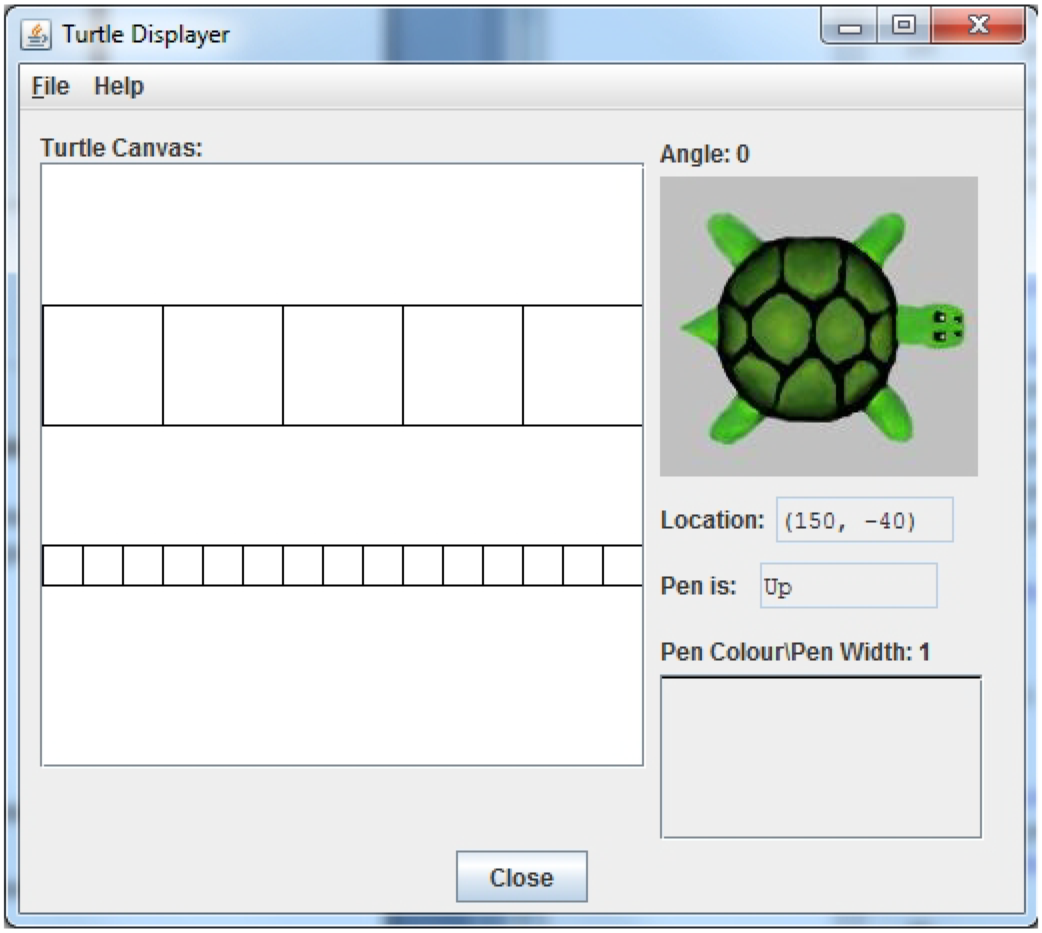
```
     private void drawOctagon ( int radius ) { …
```

which draws an octagon as described above with radius `radius`.

For submission, run the program to draw an octagon of radius 100 centered on the canvas as above.

Part B
Question 2

Write a Java program to draw squares across the width of the displayer.



Location: (150, -40)
Pen is: Up

The squares should exactly fit from one side to the other.

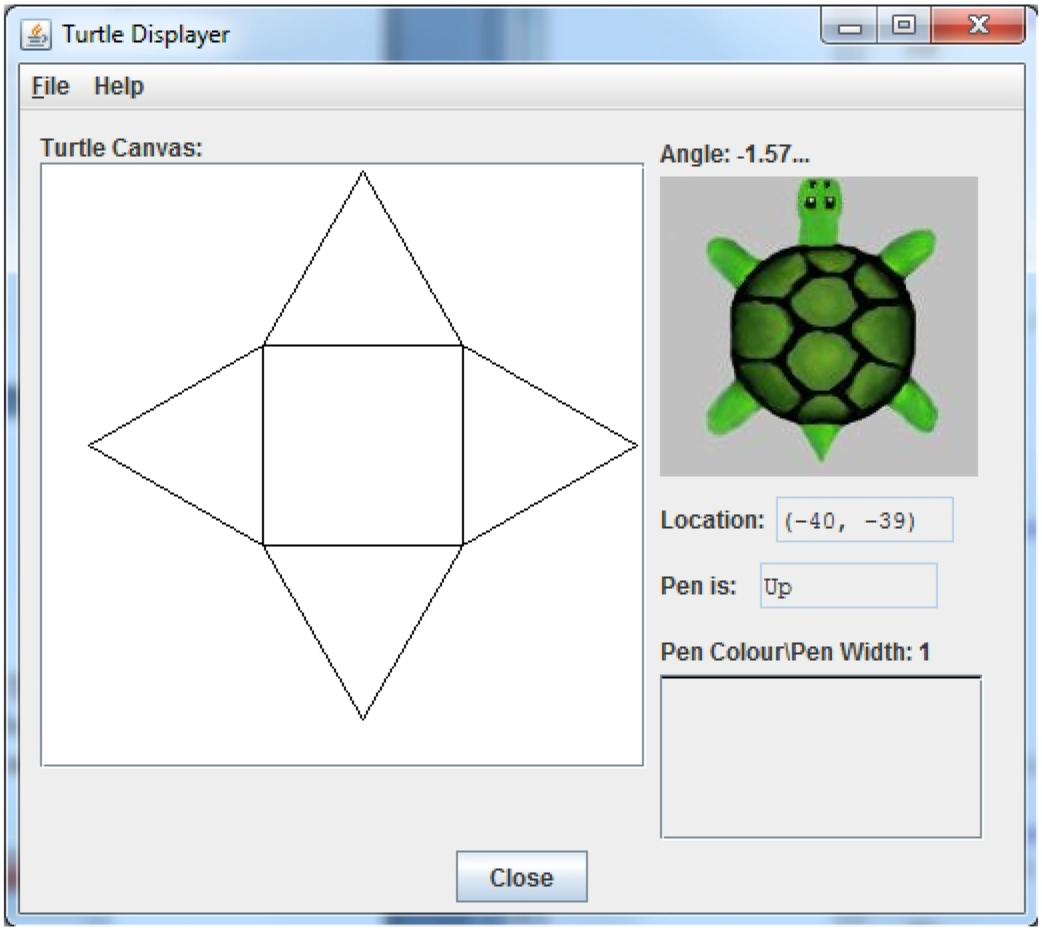
In preparing your solution, write a method:
    `private void drawSquare ( int length ) { …`
which draws a square with sides of length `length`, starting from the turtle's current position.

Also write a method:
    `private void drawNSquares ( int n ) { …`
which draws `n` appropriately sized squares across the displayer starting from the turtle's current position. The displayer is 300 units across.

To run your program for submission, draw a series of 5 squares and another series of 15 squares as above.

Part B
Question 3

Write a Java program to draw an open pyramid—what a square pyramid would look like if you folded down the sides—as seen below:



The open pyramid consists of four equilateral triangles arranged around a square.

In preparing your solution, write a method:

```
     private void drawTriangle ( int side ) { …
```

which draws a triangle with sides of length `side`, starting from the turtle's current position.

Also write a method:

```
     private void drawPyramid ( int base ) { …
```

which draws an open pyramid with base being a square `base` units on a side, consisting of four appropriately arranged triangles of appropriate size.

To run your program for submission, draw an open pyramid with base 100 as above. The exact position on the page is not critical as long as all of the pyramid can be seen.