Java Homework 4

Problem 1:

Write a Java application that simulates coin tossing. Let the program toss a coin each time the user press the “ toss ” button. Count the number of the times each side of the coin appears. Display the results by using the JTextFileds. The program should call a separate method flip that takes no arguments and returns false for tails and true for heads.

Problem 2:

Write an application that plays “guess the number” as follows: Your program chooses the number to be guessed by selecting a random integer in the range 1 to 1000. The application displays the prompt **Guess a number between 1 and 1000** next to a **JTextField.** The player types a first guess into the **JTextField** and presses the *Enter* key. If the player’s guess is incorrect, your program should display **Too high. Try again. or Too low. Try again** in the status bar to help the player “zero in”(瞄準) on the correct answer. The program should clear the **JTextField** so the user can enter the next guess. When the user enters the correct answer, display **Congratulations. You guessed the number!!** In the status bar, and clear the **JTextField** so the user can play again.

(*Note*: The guessing technique employed in this problem is similar to a *binary search*.)

Problem 3:

Write a Java **application program** that displays a text field and two buttons labeled “uppercase” and “lowercase”, when the user clicks on the uppercase button, the text changes to uppercase; when the user clicks on the lowercase button, the text changes to lowercase.

Problem 4: Fractals

A fractal is a geometrical figure, but unlike triangles, circles, and rectangles, fractals can be divided into parts, each of which is reduced-size copy of whole. Here we introduce the *Sierpinski triangle,* named after a famous Polish mathematician*.*

A *Sierpinski triangle* is created as follows:

1. Begin with an equilateral triangle, which is considered to be a *Sierpinski* fractal of order (or level) 0.
2. Connect the midpoints of the sides of the triangle of order 0 to create a *Sierpinski triangle* of order 1
3. Leave the center triangle intact(原封不動的). Connect the midpoints of the sides of the three other triangles to create a *Sierpinski triangle* of order 2.
4. You can repeat the same process recursively to create a *Sierpinski triangle*

Of order 3,4,…,and so on.

Note:

How do you draw a Sierpinski triangle of order 1? The problem can be reduced to draw three Sierpinski triangles of order 0.

How do you draw a Sierpinski triangle of order 2? The problem can be reduced to draw three Sierpinski triangles of order 1.

How do you draw a Sierpinski triangle of order *n*? The problem can be reduced to draw three Sierpinski triangles of order *n*-1.

The following file “Sierp.java” gives a Java applet that displays a Sierpinski triangle of any order. You can enter an order in a text field to display a Sierpinski triangle of specified order.

//Sierp.java

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Sierp extends JApplet implements ActionListener

{ private JTextField input = new JTextField("0", 5);

private SierpPanel sp = new SierpPanel();

public Sierp ( ) {

JPanel pl = new JPanel();

pl.add(new JLabel("Enter an order: "));

pl.add(input);

input.setHorizontalAlignment(SwingConstants.RIGHT);

add(sp);

add(pl, BorderLayout.SOUTH);

input.addActionListener( this ); }

public void actionPerformed( ActionEvent event ){

sp.setOrder(Integer.parseInt(input.getText())); }

}

class SierpPanel extends JPanel {

private int order = 0;

public void setOrder(int order){

this.order = order;

repaint(); }

public void paintComponent(Graphics g){

super.paintComponent(g);

Point p1 = new Point(getWidth()/2, 10);

Point p2 = new Point(10, getHeight()-10);

Point p3 = new Point(getWidth()-10, getHeight()-10);

display(g, order, p1, p2, p3); }

private void display(Graphics g, int order, Point p1, Point p2, Point p3){

if ( order == 0){

g.drawLine(p1.x, p1.y, p2.x, p2.y);

g.drawLine(p1.x, p1.y, p3.x, p3.y);

g.drawLine(p2.x, p2.y, p3.x, p3.y); }

else {

Point p12 = midPoint(p1, p2);

Point p23 = midPoint(p2, p3);

Point p31 = midPoint(p3, p1);

System.out.println("order="+order);

display(g, order-1, p1, p12, p31);

display(g, order-1, p12, p2, p23);

display(g, order-1, p31, p23, p3); } }

private Point midPoint(Point p1, Point p2){

return new Point( (p1.x+p2.x)/2, (p1.y+p2.y)/2); }

}

Problem:

(I). Please execute the above program by yourself. Describe what you learn from this program.

(II).Please revise the applet program in the above into **application program** and let the user use the + and－ buttons to increase and decrease the current order by 1. The initial order is 0. If the current order is 1, the decrease button is ignored.