Java Homework 3

Problem 1:

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 2: Finding the closet pair

Given a set of point, the closet-pair problem is to find the two points that are nearest to each other.

Example: (-1, 3), (-1, -1), (1, 1), (2, 0.5), (2, -1), (3, 3), (4, 2), (4, -0.5)

The points (1, 1) and (2, 0.5) are closet to each other.

Hint: There are several ways to solve this problem. An intuitive approach is to compute

the distances between all pairs of points and find the one with the minimum distance.

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.

import java.awt.\*;

import java.awt.event.\*;

import javax.swing.\*;

public class Sierp extends JFrame 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())); }

public static void main( String args[] ) // execute application

{ JFrame w = new Sierp();

w.setDefaultCloseOperation( JFrame.EXIT\_ON\_CLOSE );

w.setSize( 500, 500 );

w.setVisible( true );

}

}

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 application program in the above, 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.

Problem 5:

Using BigInteger and BigDecimal classes

a. The BigInteger and BigDecimal classes can be used to represent integers or decimal numbers of any size and precision.

b. The BigInteger and BigDecimal classes are in java.math package.

c. You can use new BigInteger(String) and new BigDecimal(String) to create an object of BigInteger and BigDecimal.

d. Use add, subtract,…methods to perform arithmetic operation

Case I. Please write a java application program to show the arithmetic operations of big data (at least 30 digits)

Case II. Please write a java application program to calculate the factorial of an integer n, the value of n is at least 45.