1.124J Foundations of Software Engineering

Problem Set 6 - Solution

Due Date: Thursday 11/2/00

Problem 1:[60%]

Part A:

Sol6_1.java

```
public class Sol6_1a
{
   public static void main(String args[])
    {
      new Plotter1();
    }
}
```

plotter.java

```
import java.io.*;
import java.util.*;

public class Plotter1
{
 Plotter1()
 {
   readData();
 }

 void readData()
  {
```

```
FileInputStream ifp;
double x[], y[];
  x = y = null;

  String fileName =  new String("data6_1");

  try{
     ifp = new FileInputStream(fileName);
    }
  catch(FileNotFoundException e)
    {
      System.out.println(" File " + fileName + " was not found.");
      return;
  }

  int i, n=0;

  InputStreamReader rd = new InputStreamReader(ifp);
  StreamTokenizer tk = new StreamTokenizer(rd);

  try {
       tk.nextToken();
       n = (int)tk.nval;
       x = new double[n];
       y = new double[n];

       for(i=0;i<n;i++)
         {
           tk.nextToken();
           x[i] = (double)tk.nval;
           tk.nextToken();
           y[i] = (double)tk.nval;
         }
       ifp.close();
     }
    catch(IOException e)
     {
      System.out.println("IOException: " + e.getMessage());
     }
    System.out.println("\n" + n + " points have been read");
        for(i=0;i<n;i++)
          {
            System.out.print("x[" + (i+1) +"]=" + x[i]);
```

```
            System.out.println("\ty[" + (i+1) +"]=" + x[i]);
          }
  }
}
```

Part B:

Sol6_1.java

```
public class Sol6_1
{
   public static void main(String args[])
    {
      new Plotter();
    }
}
```

plotter.java

```
import javax.swing.*;

import java.awt.*;
import java.awt.event.*;
import java.io.*;
import java.util.*;

public class Plotter extends JFrame
                implements ActionListener
{
   private JMenuBar menuBar ;
   private JMenu fileMenu;
   private JMenuItem importMI, exitMI;

   private JToolBar toolBar = new JToolBar();
   private JButton dashedButton, solidButton;
   private JPanel contentPane;

   private Plot plot;
```

```
  String fileName;
  FileInputStream ifp =null;

  public Plotter()
   {
   super("Problem Set 6 - Problem 3: Plotter");
     JPopupMenu.setDefaultLightWeightPopupEnabled(false);
     setSize(500,500);

   setMenuBar();
   contentPane = new JPanel() ;
   contentPane.setLayout(new BoxLayout(contentPane, BoxLayout.Y_AXIS));
   setToolBar();
   setPlot();

   setContentPane(contentPane);



   setVisible(true);

     addWindowListener(new WindowAdapter()
                 {
              public void windowClosing(WindowEvent e)
                    {
        dispose(); System.exit(0);
        }
                 }                  );
  }

 private void setMenuBar()
  {
     menuBar = new JMenuBar();

     fileMenu = new JMenu("File");
     menuBar.add(fileMenu);
     importMI = new JMenuItem("Import Data");
     importMI.addActionListener(this);
     fileMenu.add(importMI);
     exitMI = new JMenuItem("Exit");
     exitMI.addActionListener(this);
     fileMenu.add(exitMI);
```

```
    setJMenuBar(menuBar);
   }

  private void setToolBar()
   {
    toolBar = new JToolBar();

    dashedButton = new JButton(new ImageIcon("dashed.gif"));
    dashedButton.setMnemonic(KeyEvent.VK_D);
    dashedButton.setToolTipText("Dashed line");
    dashedButton.addActionListener(this);
    toolBar.add(dashedButton);
    toolBar.addSeparator();

    solidButton = new JButton(new ImageIcon("solid.gif"));
    solidButton.setMnemonic(KeyEvent.VK_S);
    solidButton.setToolTipText("Solid line");
    solidButton.addActionListener(this);
    toolBar.add(solidButton);
    toolBar.addSeparator();

    contentPane.add(toolBar);
   }

 private void setPlot()
  {
      plot = new Plot();
   contentPane.add(plot);
  }

  public void actionPerformed(ActionEvent evt)
   {
    Object src = evt.getSource();

    if(src == exitMI)
      System.exit(0);
  else if (src == importMI)
    readData();
  else if (src == dashedButton)
      plot.setStroke(Plot.dashedStroke);
  else if (src == solidButton)
      plot.setStroke(Plot.solidStroke);
   }
```

```
void readData()
{
 double x[], y[];
   x = y = null;

   fileName =  new String("data6_1");

   try{
       ifp = new FileInputStream(fileName);
     }
   catch(FileNotFoundException e)
     {
       System.out.println(" File " + fileName + " was not found.");
       return;
   }

   int i, n=0;

   InputStreamReader rd = new InputStreamReader(ifp);
   StreamTokenizer tk = new StreamTokenizer(rd);

   try {
        tk.nextToken();
        n = (int)tk.nval;
        x = new double[n];
        y = new double[n];

        for(i=0;i<n;i++)
          {
            tk.nextToken();
            x[i] = (double)tk.nval;
            tk.nextToken();
            y[i] = (double)tk.nval;
          }
        ifp.close();
      }
     catch(IOException e)
      {
        System.out.println("IOException: " + e.getMessage());
      }
     System.out.println("\n" + n + " points have been read");
   for(i=0;i<n;i++)
       {
```

```
            System.out.print("x[" + (i+1) +"]=" + x[i]);
            System.out.println("\ty[" + (i+1) +"]=" + x[i]);
          }
  plot.setPoints(x,y);
    }

}
```

plot.java

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



public class Plot extends JPanel
{
 final static float dash[] = {10.0f};
   final static BasicStroke dashedStroke = new BasicStroke(1.0f,
                                       BasicStroke.CAP_BUTT,
                                       BasicStroke.JOIN_MITER,
                                       10.0f, dash, 0.0f);
   final static BasicStroke solidStroke = new BasicStroke(1.0f);
   BasicStroke stroke = solidStroke;

 double x[]=null, y[]=null;
   int xx[], yy[];
 double minX, maxX, minY, maxY;

 public Plot()
     {
  super();

       setBackground(Color.blue);
     setForeground(Color.black);
 }

 void setStroke(BasicStroke s)
 {
  stroke = s;
  repaint();
```

```
}

   public void paintComponent(Graphics g)
  {
 super.paintComponent(g);
       Graphics2D g2 = (Graphics2D) g;
       int h = getHeight();
 int w = getWidth();
 int i, x1, y1, x2, y2, n;
 double absMaxY;

 if(x==null)
  return;

 n = x.length;

 x1 = (int)(w*0.1);
 x2 = (int)(w*0.9);
 y1 = y2 = h/2;

 g2.setStroke(solidStroke);
 g2.drawLine(x1, y1, x2, y2);
 g2.drawString("x="+minX,(int)(w*0.01), (int)(h*0.51));
 g2.drawString("x="+maxX,(int)(w*0.91), (int)(h*0.51));
 g2.drawString("y="+maxY,(int)(w*0.05), (int)(h*0.10));
 g2.drawString("y="+minY,(int)(w*0.05), (int)(h*0.95));

 g2.setStroke(stroke);

 if(Math.abs(maxY)>Math.abs(minY))
    absMaxY = Math.abs(maxY);
 else
    absMaxY = Math.abs(minY);

for(i=0;i<n;i++)
    {
   xx[i] = (int) (x1 + (x[i]-minX)/(maxX-minX)*0.8*w);
   yy[i] = (int) (y1 - y[i]/absMaxY * 0.4 * h);
    }

 g2.drawPolyline(xx, yy, xx.length);

  }
```

```
 void setPoints(double x[], double y[])
 {
  int i , n ;

  n = x.length;
  this.x = new double[n];
  this.y = new double[n];
  xx = new int[n];
  yy = new int[n];

  maxX = x[0];
  minX = x[0];
  maxY = y[0];
    minY = y[0];

       for(i=0;i<n;i++)
     {
            this.x[i] = x[i];
     if(maxX < x[i])
      maxX = x[i];
     else if(minX > x[i])
      minX = x[i];

     this.y[i] = y[i];
     if(maxY < y[i])
      maxY = y[i];
     else if(minY > y[i])
      minY = y[i];
       }
  repaint();
 }

}
```

Problem 2:[35%]

Sol6_2.java

```
public class Sol6_2
{
```

```
   public static void main(String[] args)
       {
           CreditCardAccount x1 = new CreditCardAccount(32483273, 0.00, 10000.00);
     System.out.println("\nNewly created credit card account:\n" + x1);
           System.out.println();
           CreditCardAccount x2 = new CreditCardAccount(93455454, 250.75, 5000.00);
           System.out.println("Newly created credit card account:\n" + x2 + "\n");

           ThreadGroup g = new ThreadGroup("Thread Group X");

           CreditCardTranscationsThread t1 = new CreditCardTranscationsThread(x1, g, "Thread t1");
           CreditCardTranscationsThread t2 = new CreditCardTranscationsThread(x2, g, "Thread t2");

           CreditCardTransactionsRunnable r1 = new CreditCardTransactionsRunnable(x1);
           CreditCardTransactionsRunnable r2 = new CreditCardTransactionsRunnable(x2);

           Thread t3 = new Thread(r1);
           Thread t4 = new Thread(r2);

     System.out.println("ThreadGroup: " + g);
     System.out.println("Thread: " + t1);
     System.out.println("Thread: " + t2);
     System.out.println();

     System.out.println("ThreadGroup: " + t3.getThreadGroup());
     System.out.println("Thread: " + t3);
     System.out.println("Thread: " + t4);
     System.out.println();

     t1.start();
           t2.start();
      t3.start();
           t4.start();
         }
}
```

CreditCardAccount.java

```
import java.text.*;
import java.io.* ;

class CreditCardAccount
```

```
{
   private int creditID;
   private double currentBalance;
   private double allowableLimit;
   static private DecimalFormat df = new DecimalFormat("##,##0.00");
   FileOutputStream ofp;
byte tmpByte[];
String tmpStr;

CreditCardAccount(int creditID, double currentBalance,
                  double allowableLimit)
   {
        this.creditID = creditID;
        this.currentBalance = currentBalance;
        this.allowableLimit = allowableLimit;
   File file = new File("c" + creditID);

   try
   {
     ofp = new FileOutputStream(file);
   }
       catch( FileNotFoundException e)
        {
          System.out.println(" File " + file + " could not be opened.");
        }
   }


double getCurrentBalance()
   {
         return currentBalance;
   }

double getAllowableLimit()
   {
         return allowableLimit;
   }


   synchronized void charge(double amount)
     {

     if(confirmation(amount))
```

```
   {
          currentBalance += amount;
          tmpStr = "Account has been charged an amount of $"
                 +  df.format(amount) + "\n";
   }
   else
   {
     tmpStr = "The amount $" + df.format(amount) +
       " was not authorized to be charged \n";
   }

    tmpStr += "   Account: " + this + "\n";
       tmpStr += "   By the Thread: " + Thread.currentThread() + "\n";
 tmpStr += "   of the ThreadGroup: " + Thread.currentThread().getThreadGroup() + "\n";

  try
    {
    tmpByte = tmpStr.getBytes();
    ofp.write(tmpByte);
      }
     catch(java.io.IOException e)
    {
     System.out.println("Unable to write to file: " );
         }
    }


boolean confirmation(double amount)
{
  try{
            Thread.sleep((int)(100*Math.random()));
         }
       catch(InterruptedException e)
         {
           System.out.println("Catch InterruptedException in run(): "
                                + "\n" + e.getMessage());
         }

  if(currentBalance+amount < allowableLimit)
   return true;
  else
   return false;
}
```

```
   synchronized void payment(double amount)
    {
        currentBalance -= amount;
        tmpStr = "A payment of " + df.format(amount) + " has been made\n";

   tmpStr += "   Account: " + this + "\n" ;
        tmpStr += "   By the Thread: " + Thread.currentThread() + "\n";
   tmpStr += "   of the ThreadGroup: " + Thread.currentThread().getThreadGroup() + "\n";

   try
    {
    tmpByte = tmpStr.getBytes();
    ofp.write(tmpByte);
      }
     catch(java.io.IOException e)
    {
     System.out.println("Unable to write to file: " );
         }
    }


   synchronized public String toString()
     {
        return ("   Credit card number = " + creditID +
     "   Current Balance = " + df.format(currentBalance)+
     " Limit = " +  df.format(allowableLimit));
     }

}
```

CreditCardTranscationsThread.java

```
class CreditCardTranscationsThread extends Thread
{
   private CreditCardAccount account;

   CreditCardTranscationsThread(CreditCardAccount account,
            ThreadGroup threadGroup, String threadName)
```

```
    {
      super(threadGroup, threadName);
      this.account = account;
    }

  public void run()
    {
      for (int i=0; i<10; i++)
      {
     try{
                Thread.sleep((int)(100*Math.random()));
              }
            catch(InterruptedException e)
              {
                System.out.println("Catch InterruptedException in run(): "
                                + "\n" + e.getMessage());
              }
            account.charge(Math.random()*account.getAllowableLimit());
            account.payment(Math.random()*account.getCurrentBalance());
  }
    }
}
```

CreditCardTransactionsRunnable.java

```
class CreditCardTransactionsRunnable implements Runnable
{
  private CreditCardAccount account;

  CreditCardTransactionsRunnable(CreditCardAccount account)
    {
      this.account = account;
    }

  public void run()
    {
      for (int i=0; i<10; i++)
      {
     try{
                Thread.sleep((int)(100*Math.random()));
              }
```

```
            catch(InterruptedException e)
              {
                System.out.println("Catch InterruptedException in run(): "
                                   + "\n" + e.getMessage());
              }
             account.charge(Math.random()*account.getAllowableLimit());
             account.payment(Math.random()*account.getCurrentBalance());
   }
    }
}
```
