СПбНИУ ИТМО
Кафедра ВТ

Лабораторная работа №4
по дисциплине
«Программирование интернет-приложений»
Вариант 2027

Выполнил
Широков О.И
гр.2120

Санкт-Петербург
г.2013

1. Текст задания.

Доработать программу из лабораторной работы №3 следующим образом. Реализовать приложение на базе Swing API, которое отображает на экране заданную область и заданные компоненты пользовательского интерфейса, с помощью которых вводятся данные о координатах точек и параметре R.

При щелчке мышкой по графику должна отображаться точка, цвет которой зависит от попадания или непопадания в область, при этом компоненты графического интерфейса должны отображать значения координат точки. При задании значений координат точки и R на графике должна также отображаться точка соответствующего цвета.

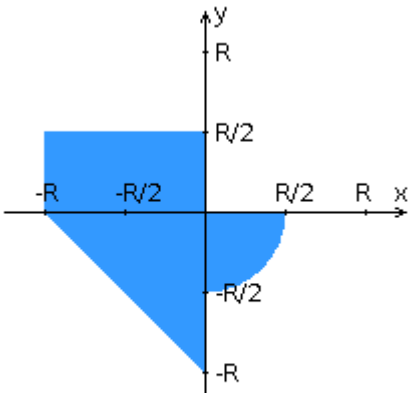
Согласно полученному варианту необходимо реализовать анимацию с использованием Java-потоков.

Приложение должно использовать следующие элементы:

- Для задания координаты X использовать JComboBox.
- Для задания координаты У - JCheckBox.
- Для задания R - JSlider.
- Для отображения координат установленной точки - JLabel.
- Элементы необходимо группировать с использованием менеджера компоновки GridLayout.
- В рамках групп необходимо использовать FlowLayout .
- При изменении радиуса должна осуществляться перерисовка фигуры с сохранением масштаба.
- При отрисовке области в качестве цвета фона использовать светло-зеленый цвет.
- Для заливки области использовать темно-зеленый цвет.

Приложение должно включать анимацию следующего вида:
точки на области должны плавно появляться в течение 4 секунд после установки
Условие запуска анимации: установка точки вне области.

Многопоточность должна быть реализована с помощью реализации интерфейса Runnable.



2. Исходный код

```
package RegLocation;


import java.awt.*;
import java.util.Vector;

public class SilhouetteView
{
    Vector<Mark> SAL;
    SizesConverter mySizeConverter;
    float MarkMaxR = 10;

    public SilhouetteView(Vector<Mark> SAL,SizesConverter convert)
    {
        this.SAL = SAL;
        mySizeConverter = convert;
    }

    public void paint(Graphics g)
    {

        g.setColor(Color.WHITE);
        g.fillRect(0,0,(int)mySizeConverter.getWidth(),
(int)mySizeConverter.getHeight());
        paintSilhouette(g);
        paintAxes(g);
        paintPoints(g);
    }

    void paintSilhouette(Graphics g)
    {
        int R       = mySizeConverter.toPx(SilhouetteComponent.R);
        int HalfR   = mySizeConverter.toPx(SilhouetteComponent.R/2);
        int XCenter = mySizeConverter.getCenterX();
        int YCenter = mySizeConverter.getCenterY();

        g.setColor(Color.BLUE);
        g.fillRect(XCenter-R,YCenter-HalfR,R, HalfR);
        g.fillArc(XCenter-HalfR,YCenter-HalfR,R,R,0,-90);
        g.fillPolygon(getTrianglePolygon(R));
    }

    void paintAxes(Graphics g)
    {
        int R       = mySizeConverter.toPx(SilhouetteComponent.R);
        int HalfR   = mySizeConverter.toPx(SilhouetteComponent.R/2);
        int XCenter = mySizeConverter.getCenterX();
        int YCenter = mySizeConverter.getCenterY();

        g.setColor(Color.BLACK);
        g.drawLine(0, YCenter, 2*XCenter,YCenter);
        g.drawLine(XCenter, 0, XCenter, 2*YCenter);

        g.drawLine(XCenter - R, YCenter + 5, XCenter - R, YCenter - 5);
```

```
        g.drawLine(XCenter + R, YCenter - 5, XCenter + R, YCenter + 5);
        g.drawLine(XCenter + 5, YCenter + R, XCenter - 5, YCenter + R);
        g.drawLine(XCenter + 5, YCenter - R, XCenter - 5, YCenter - R);

        g.drawLine(XCenter - HalfR, YCenter + 5, XCenter - HalfR, YCenter - 5);
        g.drawLine(XCenter + HalfR, YCenter - 5, XCenter + HalfR, YCenter + 5);
        g.drawLine(XCenter + 5, YCenter + HalfR, XCenter - 5, YCenter + HalfR);
        g.drawLine(XCenter + 5, YCenter - HalfR, XCenter - 5, YCenter - HalfR);
    }

    public void paintPoints(Graphics g)
    {
        for(Mark item : SAL)
        {
            int item_x,item_y;
            float factor = item.incFactor();
            item_x =  mySizeConverter.getCenterX() +
mySizeConverter.toPx(item.getCoord()[0]);
            item_y =  mySizeConverter.getCenterY() -
mySizeConverter.toPx(item.getCoord()[1]);
            int radius = (int)(MarkMaxR*factor);

            g.setColor(Color.RED);
            g.fillOval(item_x - radius/2,item_y - radius/2, radius,radius);
        }
    }

    Polygon getTrianglePolygon(int R)
    {
        int XCenter = mySizeConverter.getCenterX();
        int YCenter = mySizeConverter.getCenterY();

        int num_points = 3;
        int[] x_points = new int[num_points];
        int[] y_points = new int[num_points];

        x_points[0] = XCenter - R;
        y_points[0] = YCenter;

        x_points[1] = XCenter;
        y_points[1] = YCenter;

        x_points[2] = XCenter;
        y_points[2] = YCenter + R;

        return new Polygon(x_points,y_points,num_points);
    }
}

package RegLocation;

import javax.swing.*;
import java.awt.*;
import java.awt.event.ComponentAdapter;
import java.awt.event.ComponentEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;
import java.util.ArrayList;
import java.util.Vector;

public class SilhouetteComponent extends JComponent
{
```

```
    final static long delay = 100; //ms
    static float R;
    Silhouette silhouette;
    SilhouetteView silhouetteView;
    Vector<MarksListener> MarksListeners;
    SizesConverter converter;
    Vector<Mark> MAL;

    public SilhouetteComponent(float R)
    {
        this.R = R;
        MarksListeners = new Vector<MarksListener>();
        MAL = new Vector<Mark>();
        silhouette = new Silhouette();
        converter = new SizesConverter(getSize());
        silhouetteView = new SilhouetteView(MAL,converter);

        Thread Animator = new Thread(new Runnable()
        {
            @Override
            public void run()
            {
                try
                {
                    while(true)
                    {
                        silhouetteView.paintPoints(getGraphics());
                        Thread.currentThread().sleep(delay);
                    }
                }
                catch (InterruptedException ex)
                {
                    Thread.currentThread().interrupt();
                }
            }
        });

        Animator.start();

        addMouseListener(new MouseAdapter()
        {
            @Override
            public void mouseClicked(MouseEvent e)
            {
                int x = e.getX();
                int y = e.getY();

                markAdd(converter.toFloat(x -
getWidth()/2),converter.toFloat(getHeight()/2 - y));
            }
        });

        addComponentListener(new ComponentAdapter()
        {
            @Override
            public void componentResized(ComponentEvent e)
            {
                Rectangle b = e.getComponent().getBounds();
                e.getComponent().setBounds(b.x,b.y,b.width,b.width);
                converter.setSCS(e.getComponent().getSize());
                paintComponent(getGraphics());
                //Update component?
```

```
            }
        });

    }

    public void setR(float R)
    {
        SilhouetteComponent.R = R;
        silhouetteView.paint(getGraphics());
    }

    public void addMarksListener(MarksListener listener)
    {
        MarksListeners.add(listener);
    }

    public void MarksActionPerformed(Mark mk)
    {
        for(MarksListener item : MarksListeners)
            item.actionPerformed(mk);
    }

    public void markAdd(float x,float y)
    {
        TOL type = TOL.FOREVER;

        if(silhouette.InRegion(x,y) != 0)
            type = TOL.EXPAND;
        Mark newMark = new Mark(x,y,type);
        MAL.add(newMark);
        MarksActionPerformed(newMark);
    }

    @Override
    public void paintComponent(Graphics g)
    {
        super.paintComponent(g);
        silhouetteView.paint(g);
    }

    @Override
    public void update(Graphics g)
    {
        super.update(g);
        paintComponent(g);
    }


}
package RegLocation;

import java.awt.*;
public class SizesConverter
{
    Dimension SCS; //SilhouetteComponentSize
    static final float SilhouetteResolution = 4; //How many R in Plot
    public SizesConverter(Dimension size)
    {
        this.SCS = size;
    }
```

```
    public void setSCS(Dimension size)
    {
        this.SCS = size;
    }

    public  int toPx(float distance)
    {
        return toPxX(distance);
    }

    public int toPxY(float distance)
    {
       return (int)((distance*SCS.getHeight())/SilhouetteResolution);
    }

    public int toPxX(float distance)
    {
        return (int)((distance*SCS.getWidth())/SilhouetteResolution);
    }

    public float toFloat(int pxDistance)
    {
        return SilhouetteResolution*pxDistance/((float)SCS.getHeight());
    }

    public int getCenterX()
    {
        return (int)SCS.getWidth()/2;
    }

    public int getCenterY()
    {
        return (int)SCS.getHeight()/2;
    }

    public float getWidth() { return (float)SCS.getWidth(); }
    public float getHeight() { return (float)SCS.getHeight(); }

}
```

4.Выводы

В процессе выполнения ЛР был изучен пакет Swing, модель обработки событий, модель многопоточности в Java.