ИТМО Кафедра Вычислительной техники

Отчет по лабораторной работе №4
Программирование интернет-приложений
Вариант 1817

Выполнил: студент группы P3217
Плюхин Дмитрий
Преподаватель: Гаврилов А. В.

2016 год

1. Задание к лабораторной работе

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 - JSpinner.
- Для отображения координат установленной точки - JTextField.
- Элементы необходимо группировать с использованием менеджера компоновки GridLayout.
- В рамках групп необходимо использовать FlowLayout .
- При изменении радиуса должна осуществляется перерисовка точек с пересчетом масштаба.
- При отрисовке области в качестве цвета фона использовать светло-зеленый цвет.
- Для заливки области использовать синий цвет.

Приложение должно включать анимацию следующего вида:
при установке точки цвет области должен плавно измениться на светло-желтый и вернуться в первоначальное значение
Условие запуска анимации: установка точки в область.

Многопоточность должна быть реализована с помощью расширения класса Thread.

2. Исходный код

```
//Файл Lab4.java
package jswing;

import javax.swing.*;
import javax.swing.event.ChangeEvent;
import javax.swing.event.ChangeListener;
import java.awt.*;
import java.awt.event.*;
import java.util.ArrayList;
import java.util.LinkedHashSet;
import java.util.Set;

public class Lab4 extends JFrame{
    private static final int DEFAULT_WINDOW_WIDTH = 1024;
    private static final int DEFAULT_WINDOW_HEIGHT = 480;
    private static final double DEFAULT_R = 10d;
    private static final int DEFAULT_ELEMENT_WIDTH = 100;
    private static final int DEFAULT_ELEMENT_HEIGHT = 20;

    private JComboBox<Double> xComboBox;    //For entering x
    private ArrayList<JCheckBox> yCheckBoxes;   //For entering y
    private JSpinner rSpinner;  //For entering r
    private JTextField pTextField;  //For showing ponto
    private GraphPanel theGraphPanel;   //For showing graph

    private Double R;
    private GeneralSilhouette gsh;
    private Set<Ponto> pontos = new LinkedHashSet<>();

    public static void main(String[] args) {
        new Lab4();
    }

    public Lab4()
```

```
    {
        // Initialization
        super();

        setSize(DEFAULT_WINDOW_WIDTH, DEFAULT_WINDOW_HEIGHT);
        setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        setLocationRelativeTo(null);
        setTitle("Fourth Lab");
        setResizable(false);

        JPanel theMainPanel = new JPanel();
        theMainPanel.setLayout(new GridLayout(0,3));

        JPanel thePanelX = newPanelFlowLeft();

        JPanel thePanelY = newPanelFlowLeft();

        JPanel thePanelR = newPanelFlowLeft();

        JPanel thePanelPoint = newPanelFlowLeft();

        //UI components for x
        JLabel xLabel = new JLabel("Please, select x :");
        thePanelX.add(xLabel);

        xComboBox = getComboBoxForX();
        thePanelX.add(xComboBox);
        theMainPanel.add(thePanelX);

        //UI components for y
        JLabel yLabel = new JLabel("Please, select y :");
        thePanelY.add(yLabel);

        yCheckBoxes = getCheckBoxesForY();

        for (JCheckBox checkBox : yCheckBoxes){
            thePanelY.add(checkBox);
        }
        theMainPanel.add(thePanelY);

        //UI components for R
        JLabel rLabel = new JLabel("Please, select R :");
        thePanelR.add(rLabel);

        rSpinner = getSpinnerForR();
        thePanelR.add(rSpinner);

        theMainPanel.add(thePanelR);

        //UI components for point coordinates
        JLabel pLabel = new JLabel("Selected point :");
        thePanelPoint.add(pLabel);

        pTextField = getTextFieldForP();
        thePanelPoint.add(pTextField);

        theMainPanel.add(thePanelPoint);

        //UI components for graph
        theGraphPanel = new GraphPanel();
        theGraphPanel.setPreferredSize(new
Dimension(GraphPanel.SIZE_OF_GRAPH,GraphPanel.SIZE_OF_GRAPH));
        theGraphPanel.addMouseListener(new GraphPanelMouseListener());

        theMainPanel.add(theGraphPanel);
        add(theMainPanel);

        setVisible(true);
    }

    private JPanel newPanelFlowLeft(){
        JPanel theNewPanel = new JPanel();
        theNewPanel.setLayout(new FlowLayout(FlowLayout.LEFT));
        return theNewPanel;
    }
```

```
    private javax.swing.JComboBox<Double> getComboBoxForX() {
        Double[] xs = {-4d, -3d, -2d, 0d, 1d, 2d, 3d, 4d};
        JComboBox<Double> jCB = new javax.swing.JComboBox<>(xs);
        jCB.setFont(new Font("Calibri",Font.PLAIN,12));
        jCB.setPreferredSize(new Dimension(DEFAULT_ELEMENT_WIDTH,DEFAULT_ELEMENT_HEIGHT));
        jCB.addActionListener(new ComboBoxListener());
        return jCB;
    }

    private ArrayList<JCheckBox> getCheckBoxesForY() {
        ArrayList<JCheckBox> jCBs = new ArrayList<>();
        JCheckBox jCB;
        Double[] ys = {-4d, -3d, -2d, 0d, 1d, 2d, 3d, 4d};
        for (Double value : ys){
            jCB = new JCheckBox(value.toString());
            jCBs.add(jCB);
            jCB.addActionListener(new CheckBoxListener());
        }
        return jCBs;
    }

    private JSpinner getSpinnerForR() {
        R = DEFAULT_R;
        JSpinner rSpinner = new JSpinner();
        rSpinner.setValue(R);
        gsh = new GeneralSilhouette(R);
        rSpinner.setPreferredSize(new Dimension(DEFAULT_ELEMENT_WIDTH,DEFAULT_ELEMENT_HEIGHT));
        rSpinner.addChangeListener(new SpinnerChangeListener());
        return rSpinner;
    }

    private JTextField getTextFieldForP() {
        JTextField pTextField = new JTextField();
        pTextField.setEditable(false);
        pTextField.setText("None");
        pTextField.setPreferredSize(new
Dimension(DEFAULT_ELEMENT_WIDTH*3,DEFAULT_ELEMENT_HEIGHT));
        return pTextField;
    }

    public static double getRealX(double x, double R){
        return (x-GraphPanel.OFFSET_TO_CENTER)*R/GraphPanel.GRAPHICAL_R;
    }

    public static double getRealY(double y, double R){
        return (-y+GraphPanel.OFFSET_TO_CENTER)*R/GraphPanel.GRAPHICAL_R;
    }

    private class GraphPanelMouseListener extends MouseAdapter {
        @Override
        public void mouseClicked(MouseEvent e) {
            Ponto newPonto = new Ponto(getRealX(e.getX(),R),getRealY(e.getY(),R));
            pontos.add(newPonto);
            ((GraphPanel)e.getSource()).showPontoAnimated(newPonto,pontos,gsh);
            pTextField.setText(newPonto.toString());
        }
    }

    private class SpinnerChangeListener implements ChangeListener{
        @Override
        public void stateChanged(ChangeEvent e) {
            theGraphPanel.paint(theGraphPanel.getGraphics());
            R = ((Integer)((JSpinner)e.getSource()).getModel().getValue()).doubleValue();
            gsh = new GeneralSilhouette(R);
            for (Ponto ponto : pontos){
                theGraphPanel.showPonto(ponto,gsh);
            }
        }
    }

    private class CheckBoxListener implements ActionListener{

        @Override
        public void actionPerformed(ActionEvent e) {
```

```
            double x = Double.parseDouble(xComboBox.getModel().getSelectedItem().toString());
            double y = Double.parseDouble(((JCheckBox)e.getSource()).getText());
            Ponto newPonto = new Ponto(x,y);

            if (((JCheckBox)e.getSource()).isSelected() && !findPonto(pontos,newPonto)){
                    pontos.add(newPonto);
                    pTextField.setText(newPonto.toString());
                    theGraphPanel.showPontoAnimated(newPonto,pontos,gsh);
            }
        }
    }

    private class ComboBoxListener implements ActionListener{
        @Override
        public void actionPerformed(ActionEvent e) {
            boolean added = false;

            double x =
Double.parseDouble(((JComboBox)e.getSource()).getModel().getSelectedItem().toString());
            double y = 0;
            Ponto newPonto = new Ponto(x,y);

            for(JCheckBox ycheckBox : yCheckBoxes) {
                y = Double.parseDouble(ycheckBox.getText());
                newPonto = new Ponto(x,y);
                if (ycheckBox.isSelected() && !findPonto(pontos,newPonto)) {
                    pontos.add(newPonto);
                    added = true;
                    pTextField.setText(newPonto.toString());
                }
            }
            if (added){
                theGraphPanel.showPontoAnimated(newPonto,pontos,gsh);
            }
        }
    }

    boolean findPonto(Set<Ponto> pontos, Ponto p){
        for(Ponto ponto : pontos){
            if (ponto.equals(p)){
                return true;
            }
        }
        return false;
    }
}
```

//Файл GraphPanel.java

```
package jswing;

import javax.swing.*;
import java.awt.*;
import java.util.Set;

public class GraphPanel extends JPanel {
    public static final int GRAPHICAL_R = 60;
    public static final int OFFSET_TO_CENTER;
    public static final int SIZE_OF_POINTER = 5;
    public static final int OFFSET_TO_LABEL = 10;
    public static final int SIZE_OF_GRAD = 2;
    public static final int SIZE_OF_GRAPH = 200;
    public static final int SIZE_OF_POINT = 2;

    public static final int BLUE = 0x0000FF;
    public static final Color INNER_POINT_COLOR = Color.green;
    public static final Color OUTER_POINT_COLOR = Color.red;

    static {
        OFFSET_TO_CENTER = SIZE_OF_GRAPH/2;
    }

    private Color areaColor = new Color(BLUE);

    public void setAreaColor(int r, int g, int b){
        areaColor = new Color(correctValue(r,0,255), correctValue(g,0,255),
correctValue(b,0,255));
```

```
    }

    public void paint(Graphics g){
        //Background
        g.setColor(new Color(0x48CC5E));
        g.fillRect(0,0,SIZE_OF_GRAPH,SIZE_OF_GRAPH);

        //Rectangle
        g.setColor(areaColor);
        g.drawRect(OFFSET_TO_CENTER,OFFSET_TO_CENTER,-GRAPHICAL_R,GRAPHICAL_R);

        //Triangle
        int[] txs = {OFFSET_TO_CENTER,OFFSET_TO_CENTER+GRAPHICAL_R/2,OFFSET_TO_CENTER};
        int[] tys = {OFFSET_TO_CENTER,OFFSET_TO_CENTER,OFFSET_TO_CENTER+GRAPHICAL_R/2};
        Polygon triangle = new Polygon(txs,tys,3);
        g.fillPolygon(triangle);

        //Circle
        g.fillArc(OFFSET_TO_CENTER-GRAPHICAL_R,OFFSET_TO_CENTER-
GRAPHICAL_R,GRAPHICAL_R*2,GRAPHICAL_R*2,0,90);

        //Coordinates
        g.setColor(Color.black);
        g.drawLine(OFFSET_TO_CENTER,GRAPHICAL_R*3,OFFSET_TO_CENTER,SIZE_OF_GRAPH-GRAPHICAL_R*3);
        g.drawLine(OFFSET_TO_CENTER,SIZE_OF_GRAPH-
GRAPHICAL_R*3,OFFSET_TO_CENTER+SIZE_OF_POINTER,SIZE_OF_GRAPH-GRAPHICAL_R*3+SIZE_OF_POINTER);
        g.drawLine(OFFSET_TO_CENTER,SIZE_OF_GRAPH-GRAPHICAL_R*3,OFFSET_TO_CENTER-
SIZE_OF_POINTER,SIZE_OF_GRAPH-GRAPHICAL_R*3+SIZE_OF_POINTER);
        g.drawString("y",OFFSET_TO_CENTER+OFFSET_TO_LABEL,SIZE_OF_GRAPH-GRAPHICAL_R*3);

        g.drawLine(GRAPHICAL_R*3,OFFSET_TO_CENTER,SIZE_OF_GRAPH-GRAPHICAL_R*3,OFFSET_TO_CENTER);
        g.drawLine(GRAPHICAL_R*3,OFFSET_TO_CENTER,GRAPHICAL_R*3-
SIZE_OF_POINTER,OFFSET_TO_CENTER-SIZE_OF_POINTER);
        g.drawLine(GRAPHICAL_R*3,OFFSET_TO_CENTER,GRAPHICAL_R*3-
SIZE_OF_POINTER,OFFSET_TO_CENTER+SIZE_OF_POINTER);
        g.drawString("x",GRAPHICAL_R*3-OFFSET_TO_LABEL,OFFSET_TO_CENTER-OFFSET_TO_LABEL);

        //Tags
        g.drawLine(OFFSET_TO_CENTER+GRAPHICAL_R,OFFSET_TO_CENTER-
SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPHICAL_R,OFFSET_TO_CENTER+SIZE_OF_GRAD);
        g.drawString("R",OFFSET_TO_CENTER+GRAPHICAL_R-OFFSET_TO_LABEL/2,OFFSET_TO_CENTER-
OFFSET_TO_LABEL/2);

        g.drawLine(OFFSET_TO_CENTER+GRAPHICAL_R/2,OFFSET_TO_CENTER-
SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPHICAL_R/2,OFFSET_TO_CENTER+SIZE_OF_GRAD);
        g.drawString("R/2",OFFSET_TO_CENTER+GRAPHICAL_R-OFFSET_TO_LABEL*4,OFFSET_TO_CENTER-
OFFSET_TO_LABEL/2);

        g.drawLine(OFFSET_TO_CENTER-GRAPHICAL_R,OFFSET_TO_CENTER-SIZE_OF_GRAD,OFFSET_TO_CENTER-
GRAPHICAL_R,OFFSET_TO_CENTER+SIZE_OF_GRAD);
        g.drawString("-R",OFFSET_TO_CENTER-GRAPHICAL_R-OFFSET_TO_LABEL,OFFSET_TO_CENTER-
OFFSET_TO_LABEL/2);

        g.drawLine(OFFSET_TO_CENTER-GRAPHICAL_R/2,OFFSET_TO_CENTER-
SIZE_OF_GRAD,OFFSET_TO_CENTER-GRAPHICAL_R/2,OFFSET_TO_CENTER+SIZE_OF_GRAD);
        g.drawString("-R/2",GRAPHICAL_R,OFFSET_TO_CENTER-OFFSET_TO_LABEL/2);

        //

        g.drawLine(OFFSET_TO_CENTER-
SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPHICAL_R,OFFSET_TO_CENTER+SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPHIC
AL_R);
        g.drawString("-
R",OFFSET_TO_CENTER+OFFSET_TO_LABEL/2,OFFSET_TO_CENTER+GRAPHICAL_R+OFFSET_TO_LABEL/2);

        g.drawLine(OFFSET_TO_CENTER-
SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPHICAL_R/2,OFFSET_TO_CENTER+SIZE_OF_GRAD,OFFSET_TO_CENTER+GRAPH
ICAL_R/2);
        g.drawString("-
R/2",OFFSET_TO_CENTER+OFFSET_TO_LABEL/2,OFFSET_TO_CENTER+GRAPHICAL_R/2+OFFSET_TO_LABEL/2);

        g.drawLine(OFFSET_TO_CENTER-SIZE_OF_GRAD,OFFSET_TO_CENTER-
GRAPHICAL_R/2,OFFSET_TO_CENTER+SIZE_OF_GRAD,OFFSET_TO_CENTER-GRAPHICAL_R/2);
        g.drawString("R/2",OFFSET_TO_CENTER+OFFSET_TO_LABEL/2,OFFSET_TO_CENTER-
GRAPHICAL_R/2+OFFSET_TO_LABEL/2);
```

```
        g.drawLine(OFFSET_TO_CENTER-SIZE_OF_GRAD,OFFSET_TO_CENTER-
GRAPHICAL_R,OFFSET_TO_CENTER+SIZE_OF_GRAD,OFFSET_TO_CENTER-GRAPHICAL_R);
        g.drawString("R",OFFSET_TO_CENTER+OFFSET_TO_LABEL/2,OFFSET_TO_CENTER-
GRAPHICAL_R+OFFSET_TO_LABEL/2);

        //Border
        g.drawLine(0,0,SIZE_OF_GRAPH,0);
        g.drawLine(0,0,0,SIZE_OF_GRAPH);
        g.drawLine(SIZE_OF_GRAPH,0,SIZE_OF_GRAPH,SIZE_OF_GRAPH);
        g.drawLine(0,SIZE_OF_GRAPH,SIZE_OF_GRAPH,SIZE_OF_GRAPH);

    }

    public void showPonto(Ponto p, GeneralSilhouette gsh){
        if (isPontoOnGraph(p,gsh.getR())){
            addPontoToGraph(p,getGraphics(),gsh);
        }
    }

    public void showPontoAnimated(Ponto p, Set<Ponto> pontos, GeneralSilhouette gsh){
        if (isPontoOnGraph(p,gsh.getR())){
            addPontoToGraph(p,getGraphics(),gsh);
            // animation
            new AnimationThread(this,pontos,gsh.getR(),gsh).start();
        }
    }

    private void addPontoToGraph(Ponto p, Graphics g, GeneralSilhouette gsh){
        if(gsh.checkPonto(p)){
            g.setColor(INNER_POINT_COLOR);
        } else {
            g.setColor(OUTER_POINT_COLOR);
        }

        g.fillOval((int)p.getGraphX(gsh.getR())-SIZE_OF_POINT,(int)p.getGraphY(gsh.getR())-
SIZE_OF_POINT,SIZE_OF_POINT*2,SIZE_OF_POINT*2);
    }

    private boolean isPontoOnGraph(Ponto p, double R){
        return (p.getGraphX(R) >= 0) &&
                (p.getGraphX(R) <= SIZE_OF_GRAPH) &&
                (p.getGraphY(R)>=0) &&
                (p.getGraphY(R)<= SIZE_OF_GRAPH);
    }

    private int correctValue(int value, int lowbound, int highbound){
        if (value < lowbound){
            return lowbound;
        }
        if (value > highbound){
            return highbound;
        }
        return value;
    }
}
```

//Файл AnimationThread.java

```
package jswing;

import javax.swing.*;
import java.util.ArrayList;
import java.util.Set;

public class AnimationThread extends Thread{

    private int step = 51;
    private int delay = 20;

    private GraphPanel animatedGraphPanel;

    private Set<Ponto> pontos;

    private double R;

    private GeneralSilhouette gsh;
```

```
    public AnimationThread(GraphPanel graphPanel, Set<Ponto> pontos, double R, GeneralSilhouette
gsh){
        animatedGraphPanel = graphPanel;
        this.pontos = pontos;
        this.R = R;
        this.gsh = gsh;
    }

    public void run(){
        int r = 0;
        int g = 0;
        int b = 255;

        animatedGraphPanel.setAreaColor(r,g,b);
        while (r != 255){
            draw(r+=step,g+=step,b-=step);
        }
        while (r != 0){
            draw(r-=step,g-=step,b+=step);
        }
    }

    private void draw(int r, int g, int b){
        animatedGraphPanel.setAreaColor(r,g,b);

        animatedGraphPanel.paint(animatedGraphPanel.getGraphics());

        for (Ponto ponto : pontos){
            animatedGraphPanel.showPonto(ponto,gsh);
        }

        try{
            Thread.sleep(delay);
        } catch (InterruptedException e){
            JOptionPane.showMessageDialog(null, "Error", "Animation thread interrupted",
JOptionPane.ERROR_MESSAGE);
        }
    }
}
```

3. Вывод

В результате лабораторной работы я ознакомился с созданием приложений на языке Java, обладающих визуальным интерфейсом. Были сделаны выводы о том, что при помощи Swing API можно создавать приложения, которые выглядят лучше, чем программы, написанные с использованием AWT, но в то же время хуже, чем написанные при помощи JavaFX. Был сделан вывод о том, что Java поддерживает распространенную модель обработки событий, схожую с подобными моделями в других высокоуровневых языках программирования. Был сделан вывод и о том, что управлять анимацией, используя Swing, совсем неудобно и для этого лучше подходит JavaFX. По моему мнению, данная лабораторная работа очень важна, поскольку позволяет получить представление о методах создания пользовательского интерфейса с использованием одного из наиболее распространенных языков программирования.