**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.

1. **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));

if (!findPonto(pontos,newPonto)) {

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+GRAPHICAL\_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+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/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);

}

}

}

1. **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.

