Университет ИТМО, кафедра ВТ

Лабораторная работа №4 по “Программированию интернет-приложений”

Работу выполнил

студент группы P3200

Рогов Я. С.

Преподаватель

Николаев В.В.

Санкт-Петербург, 2016

Задание: 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 - JTextArea.
- Элементы необходимо группировать с использованием менеджера компоновки BorderLayout.
- 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-желтый цвет.
- Для заливки области использовать коричневый цвет.

Приложение должно включать анимацию следующего вида:
после установки размер точки должен циклически изменяться от R/12 до R/22 значения и обратно
Условие запуска анимации: вход в область одной из точек при изменении радиуса.

Многопоточность должна быть реализована с помощью расширения класса Thread.

Program4.java
```
import javax.swing.*;

import java.awt.BorderLayout;
import java.awt.Dimension;
import java.awt.FlowLayout;
import java.awt.Container;
import java.awt.Shape;
import java.awt.Color;

import java.awt.geom.Path2D;
import java.awt.geom.Arc2D;
import java.awt.geom.Line2D;
import java.awt.geom.Rectangle2D;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;
import java.awt.event.MouseAdapter;
import java.awt.event.MouseEvent;

import java.lang.IllegalArgumentException;
import java.lang.Thread;

import java.util.ArrayList;
import java.util.stream.DoubleStream;

import FuncLib.*;
```

```
import java.awt.Rectangle;
import java.awt.Graphics2D;


public class Program4{

	public static void main(String[] args){
		MainGUIThread guiThread = new MainGUIThread();
		guiThread.start();
	}

}

class MainGUIThread extends Thread{
	public void run(){
		ProgramGUI gui = new ProgramGUI();
		gui.setVisible(true);
	}
}

class ProgramGUI extends JFrame{

	private JLabel jLabel;

	private final Dimension size = new Dimension(900, 700);
	private final Dimension minSize = new Dimension(700, 500);
	private final String title = "Lab4_v720";

	private JTextField dotCoordinates;

	private JComboBox x_comboBox;
	private final static Double[] x_values = DoubleStream.of(new double[]{
		-4.0f, -3.0f, -2.0f, 0.0f, 1.0f, 2.0f, 3.0f, 4.0f
	}).boxed().toArray(Double[]::new);

	private MyJCheckBoxGroup y_checkBoxGroup;
	private final static Double[] y_values = DoubleStream.of(new double[]{
		-4.0f, 2.0f, -3.0f, 4.0f, 5.0f, -1.0f, 0.0f
	}).boxed().toArray(Double[]::new);

	private JButton addDotButton;
	private ArrayList<Dot> dots;

	private MyJSliderDouble r_Slider;
	final private float r_SliderMin = 1.0f,
					    r_SliderMax = 10.0f,
					    r_SliderDefault = 3.0f,
					    r_SliderDelta = 0.01f,
					    r_SliderExtent = 2.0f;

	private Graph graph;
	final private double graphBounds = 1.1 * r_SliderMax;

	DrawableContourModel model;

	public ProgramGUI(){ initGUI(); }

	public void initGUI(){

		//initialize window
		setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
		this.setTitle(title);
		this.setSize(size);
		setMinimumSize(minSize);
		this.setLayout(new BorderLayout());

		//Field with Dot Coordinates
		dotCoordinates = new JTextField("(x; y)");
		dotCoordinates.setFont(dotCoordinates.getFont().deriveFont(20.0f));
		dotCoordinates.setEditable(false);
		this.add(dotCoordinates, BorderLayout.PAGE_START);

		//initalize controlls
		JPanel controls = getControls();
		this.add(controls, BorderLayout.PAGE_END);

		//initialize DrawableContour
		model = initModel720(r_SliderDefault);
		DrawableContourView view = new DrawableContourView(2*graphBounds);
```

```
        DrawableContourController controller = new DrawableContourController(model);

        //initialize Graph
        graph = new Graph(view, dots);
        graph.setBounds(
                        -graphBounds, graphBounds, 1.0d,
                        -graphBounds, graphBounds, 1.0d);
        graph.setBGColor(new Color(0x63, 0x38, 0x07));
        graph.setFillColor(new Color(0xff, 0xff, 0x60));
        graph.addMouseListener(new MouseAdapter(){
                public void mouseClicked(MouseEvent e){
                        addDot( graph.translateAndScale(e.getX(), true),
                                        graph.translateAndScale(e.getY(), false),
                                        0.5);
                }
        });

        model.addObserver(graph);
        model.notifyObservers();
        this.add(graph, BorderLayout.CENTER);

        r_Slider.addChangeListener(controller);

}

private void addDot(double x, double y, double speed){
        Dot dot = new Dot(x, y, speed);
        model.addObserver(dot);
        dots.add(dot);
        dot.start();

        dotCoordinates.setText(String.format("(%3.1f; %3.1f)", x, y));
}

private JPanel getControls(){

        // Dot-Set Controls
        x_comboBox = new JComboBox<Double>(x_values);
        y_checkBoxGroup = new MyJCheckBoxGroup<Double>(y_values);


        addDotButton = new JButton("Add Dot");
        dots = new ArrayList<Dot>(10);
        addDotButton.addActionListener(new ActionListener(){

                public void actionPerformed(ActionEvent e){
                        Double x = (Double) x_comboBox.getSelectedItem();
                        Double y = (Double) y_checkBoxGroup.getSelectedItem();
                        addDot(x.doubleValue(), y.doubleValue(), 0.5);
                }
        });

        // R Controls
        r_Slider = new MyJSliderDouble(
                r_SliderMin, r_SliderMax, r_SliderDefault, r_SliderDelta);
        r_Slider.DcreateStandardLabels(r_SliderExtent);
        r_Slider.setPaintLabels(true);

        JPanel controls = new JPanel(new FlowLayout());
        controls.add(x_comboBox);
        controls.add(y_checkBoxGroup);
        controls.add(addDotButton);
        controls.add(r_Slider);
        return controls;
}

private DrawableContourModel initModel720(float defaultR){
        DrawableContourModel model = new DrawableContourModel(
                defaultR,

                (float R, Vertice v) ->
                        v.x>0?
                                v.y>=0 && v.y<=R/2 - v.x:
                                v.y>0?
                                        v.y*v.y+v.x*v.x<=R*R/4:
                                        v.x>=-R && v.y>=-R/2,

                (float R) -> {Path2D.Double result = new Path2D.Double();
                        result.append(new Rectangle2D.Double(-R, 0, R, R/2), true);
                        result.append(new Line2D.Double(-R/2, 0, R/2, 0), true);
```

```
                                result.append(new Arc2D.Double(-R/2, -R/2, R, R, 90, 90, Arc2D.OPEN),
true);
                                return result;},
                        1000.0f
                );

                return model;
        }
}
```

FuncLib/BoundsLambda.java

```
package FuncLib;

public interface BoundsLambda{
        boolean check(float R, Vertice v);}
```

FuncLib/DrawableContourModel.java

```
package FuncLib;

import java.util.ArrayList;
import java.util.Arrays;
import java.awt.Shape;
import java.awt.geom.Path2D;


public class DrawableContourModel extends java.util.Observable{

        protected float R;
        protected BoundsLambda bounds;
        // to compensate scale loss in view
        protected float drawScale;

        private Path2D path;
        private GenericPath2D genericPath;

        public DrawableContourModel(
                        float R, BoundsLambda bounds,
                        GenericPath2D genericPath, float drawScale){
                this.R = R;
                this.bounds = bounds;
                this.drawScale = drawScale;
                this.genericPath = genericPath;

                setChanged();
        }

        public Path2D getPath(){ return path; };

        public float getDrawScale(){ return drawScale; }

        protected void updatePath(){
                path = genericPath.getPath(R * drawScale); }

        public boolean checkVertice(Vertice v){ return bounds.check(R, v); }

        public float getR(){ return R; }

        public void setR(float R){
                this.R = R;
                setChanged();
                notifyObservers();
        }

        @Override
        protected void setChanged(){
                super.setChanged();
                updatePath();
        }

}
```

FuncLib/DrawableContourView.java

```
package FuncLib;

import java.awt.Color;
import java.awt.Graphics2D;
import java.awt.Graphics;

import java.awt.Rectangle;
import java.awt.geom.Path2D;
```

```
import FuncLib.DrawableContourModel;
import FuncLib.Vertice;

public class DrawableContourView
                        extends javax.swing.JPanel
                        implements java.util.Observer{

        //private DrawableContourModel model;
        private Color color = Color.BLACK;
        private double scaleDiv = 1.0;
        // for compensating of scaling
        private float modelScale = 1.0f;

        Path2D pathToDraw;

        public DrawableContourView(double scaleDiv){
                this.color = color;
                this.scaleDiv = scaleDiv;
                setOpaque(true);
        }

        public double getScaleDiv(){ return scaleDiv; }

        @Override
        public void update(java.util.Observable o, Object arg){

                DrawableContourModel model = ((DrawableContourModel) o);
                modelScale=model.getDrawScale();
                pathToDraw = model.getPath();

                repaint();
        }

        public void setColor(Color color){ this.color = color; }

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

                        //get new Graphics object to modify
                        Graphics2D g2 = (Graphics2D) g.create();

                        //set center of the plane to center of component
                        Rectangle bounds = g2.getClipBounds();

                        g2.translate(bounds.width/2, bounds.height/2);
                        g2.setColor(color);
                        g2.scale(      bounds.width/scaleDiv/modelScale,
                                                bounds.height/scaleDiv/modelScale);
                        g2.draw(pathToDraw);
                        g2.fill(pathToDraw);
                }
}
```

FuncLib/GenericPath2D.java

```
package FuncLib;

import java.awt.geom.Path2D;

public interface GenericPath2D{
        public Path2D getPath(float R);
}
```

FuncLib/Vertice.java

```
package FuncLib;

public class Vertice{
                public double x;
                public double y;

                public Vertice(){ super(); }

                public Vertice(double x, double y){
                        this.x = x;
                        this.y = y;
                }

                public Vertice(double[] vert){
                        x = vert[0];
```

```
                y = vert[1];
        }

        @Override
        public String toString(){
                return String.format("(%.3f; %.3f)", x, y); }
    }
```

Graph.java

```
import FuncLib.DrawableContourView;
import java.util.ArrayList;

import java.awt.Color;
import java.awt.Graphics;
import java.awt.Graphics2D;
import java.awt.Rectangle;
import java.awt.FontMetrics;
import java.awt.Font;

public class Graph
        extends javax.swing.JPanel
        implements java.util.Observer{

    private int strokeLength = 5;

    private DrawableContourView view;
    private ArrayList<Dot> dots;

    private ArrayList<String> xStrings;
    private ArrayList<String> yStrings;

    private double xMin, xMax, dx,
                                yMin, yMax, dy;

    private Color bgColor = Color.WHITE;
    private Color fillColor = Color.BLACK;

    //graphics for dots
    Graphics2D graphics2D;
    Rectangle bounds;

    public Graph(DrawableContourView view, ArrayList<Dot> dots){
        this.view = view;
        this.dots = dots;
    }

    public void setBounds(double xMin, double xMax, double dx,
                                            double yMin, double yMax, double dy){

        this.xMin = xMin;
        this.xMax = xMax;
        this.dx = dx;
        this.yMin = yMin;
        this.yMax = yMax;
        this.dy = dy;

        xStrings = new ArrayList<String>((int)((xMax-xMin)/dx)+1);
        yStrings = new ArrayList<String>((int)((yMax-yMin)/dy)+1);

        for (;; yMax-=dy, xMin+=dx){
            if( xMin <= xMax)
                xStrings.add(String.format("%3.1f", xMin));
            if( yMax >= yMin)
                yStrings.add(String.format("%3.1f", yMax));
            else if (xMin > xMax)
                break;
        }
    }

    public void setBGColor(Color color){ bgColor = color; }

    public void setFillColor(Color color){
        fillColor = color;
        view.setColor(color);}

    public void update(java.util.Observable o, Object arg){
        view.update(o, arg);
        repaint();
    }
```

```
public double translateAndScale(double d, boolean isXAxis){
	return (d - (isXAxis?bounds.getWidth():bounds.getHeight())/2
		)/view.getScaleDiv();
}

public void paintComponent(Graphics g){

	Rectangle r = g.getClipBounds();
	super.paintComponent(g);

	//bg
	g.setColor(bgColor);
	g.fillRect(r.x, r.y, r.width, r.height);
	//function contour
	g.setColor(fillColor);
	view.paintComponent(g);

	//axis
	g.setColor(Color.BLACK);
	paintAxis(g);

	//dots
	drawDots(g);
}

public void drawDots(Graphics g){
	Graphics2D g2 = (Graphics2D) (g.create());

	bounds = g2.getClipBounds();

	g2.translate(bounds.width/2, bounds.height/2);
	g2.scale(	bounds.width/view.getScaleDiv(),
			bounds.height/view.getScaleDiv());

	for(int i=0; i<dots.size(); i++){
		System.out.println("Draw dot #" + i);
		System.out.println(dots.get(i).getStats());
		dots.get(i).paintDot(g2);
	}
}

public void initPublicGraphics2D(){
	if(graphics2D==null
		|| graphics2D.getClipBounds() == null
		){
			System.out.println("Initalizing graph's graphics2D");

			graphics2D = (Graphics2D) (getGraphics().create());
			if (graphics2D==null) System.out.println("Null graphics at init!");

			Rectangle r = getGraphics().getClipBounds();
			if(r==null) System.out.println("NULL rectangle at init!");

			graphics2D.setClip(r);

			graphics2D.translate(r.width/2, r.height/2);

			double s = view.getScaleDiv();
			graphics2D.scale(s,s);
		}
}

public void paintAxis(Graphics g){

	Rectangle r = g.getClipBounds();

	int x0 = r.width/2+r.x, y0 = r.height/2+r.y;

	//axes
	g.drawLine(r.x, r.y+r.height/2, r.x+r.width, r.y+r.height/2);
	g.drawLine(r.x+r.width/2, r.y, r.x+r.width/2, r.y+r.height);

	//strokes
	FontMetrics f = g.getFontMetrics();
	double  xScale = ((double) r.width)  /(xMax - xMin),
			yScale = ((double) r.height)/(yMax - yMin);

	int xBound = r.x + r.width;
	int yBound = r.y + r.height;
```

```
                double xStroke = r.x, yStroke = r.y;

                for (int ix=0, iy=0;
                        ;
                        ix++, iy++, xStroke += xScale*dx, yStroke += yScale*dy){

                        if(xStroke<=xBound){
                                String s = xStrings.get(ix);
                                g.drawLine(    (int) xStroke,        y0 - strokeLength/2,
                                                        (int)xStroke,  y0 + strokeLength/2);
                                g.drawString(  s,
                                                                (int)xStroke - f.stringWidth(s)/2,
                                                                y0 + strokeLength/2 + f.getHeight());
                        }
                        if(yStroke<=yBound){
                                String s = yStrings.get(iy);
                                g.drawLine( x0 - strokeLength/2, (int) yStroke,
                                                        x0 + strokeLength/2, (int) yStroke);
                                g.drawString(  s,
                                                                x0 + strokeLength,
                                                                (int) yStroke + f.getHeight()/2);
                        }
                        else if (xStroke>xBound)
                                break;
                }
        }
}
```

MyJCheckBoxGroup.java

```
import javax.swing.JPanel;
import javax.swing.JLabel;
import javax.swing.JCheckBox;

import java.awt.event.ActionListener;
import java.awt.event.ActionEvent;

import java.awt.FlowLayout;

import java.util.Arrays;
import java.util.ArrayList;

public class MyJCheckBoxGroup<T extends Object> extends JPanel{
                private ArrayList<JCheckBox> checkBoxes;
                private int selection = 0;
                private ArrayList<T> retvalues;

                public MyJCheckBoxGroup(T[] args){
                        checkBoxes = new ArrayList<>(args.length); //replace to args.length???
                        setLayout(new FlowLayout());
                        retvalues = new ArrayList<T>(Arrays.asList(args));
                        for(int i=0; i<args.length; i++){
                                JCheckBox c = new JCheckBox(args[i].toString());
                                c.addActionListener(new ActionListener (){
                                        @Override
                                        public void actionPerformed(ActionEvent e){
                                                int newselection = checkBoxes.indexOf(e.getSource());;

        checkBoxes.get(selection).setSelected(newselection==selection);
                                                selection = newselection;
                                        }
                                });
                                checkBoxes.add(c);
                                this.add(c);
                        }
                        checkBoxes.get(selection).setSelected(true);
                }

                public T getSelectedItem(){
                        return retvalues.get(selection);
                }
}
```

MyJSliderDouble.java

```
import javax.swing.JSlider;
import javax.swing.JLabel;

import java.util.Dictionary;
import java.util.Hashtable;
import java.util.Enumeration;
```

```
public class MyJSliderDouble extends JSlider{
	private double multiplier;
	private double dmin;
	private double dmax;

	public MyJSliderDouble(double min, double max, double value, double delta){
		super((int)(min/delta),(int)(max/delta));
		if(Math.abs(min/delta)>Integer.MAX_VALUE ||
			Math.abs(max/delta)>Integer.MAX_VALUE)
			throw new IllegalArgumentException(
				"Too high precision for this range");
		multiplier=delta;
		setValue((int)(value/delta));
		dmin=min;
		dmax=max;
	}

	public double DgetValue(){return multiplier * super.getValue();}
	public double DgetMinimum(){return dmin;}
	public double DgetMaximum(){return dmax;}


	public void DcreateStandardLabels(double d){
		Hashtable<Double,JLabel> labels = new Hashtable<>();
		for(double i = dmin; i<=dmax; i+=d){
			labels.put(new Double(i), new JLabel(Double.toString(i)));
		}
		DsetLabelTable(labels);
	}


	public void DsetLabelTable(Dictionary dict){
		Hashtable<Integer,JLabel> labels = new Hashtable<>();
		for(Enumeration keys = dict.keys(); keys.hasMoreElements();){
			Double d = (Double) keys.nextElement();
			int i = (int)(d / multiplier);
			labels.put(new Integer(i), (JLabel)dict.get(d));
		}
		super.setLabelTable(labels);
	}

}
```

Dot.java

```
import FuncLib.*;

import java.awt.Graphics2D;

public class Dot
		extends Thread,
		java.util.Observer {

	private boolean isAnimated;
	private double radius=1.0;
	private Vertice vertice;
	private double speed;
	//for Graphics2D
	private Graphics2D graphics2D;

	private boolean isDownscaleAnim;
	private double r;

	public Dot(double x, double y, double speed){
		vertice = new Vertice(x,y);
		this.speed = speed;
		isDownscaleAnim = true;
	}

	@Override
	public void update(java.util.Observable o, Object arg){
		DrawableContourModel model = (DrawableContourModel) o;
		radius = model.getR();
		isAnimated = model.checkVertice(vertice);
		r = radius/12;
	}

	public String getStats(){
		return "Current Radius: " + r + "\n"+
				"Current coordinates: " + vertice.x + ";" + vertice.y + "\n" +
```

```
                        "Current R: " + radius;
        }

        public void setGraphics2D(Graphics2D g){ graphics2D = g; }

        public void updateRadius(){
                if(isAnimated){
                        if (isDownscaleAnim){
                                r-=speed*radius/22;
                                if(r<radius/22)
                                        isDownscaleAnim = false;
                        }
                        else{
                                r+=speed*radius/22;
                                if(r>radius/12)
                                        isDownscaleAnim = true;
                        }
                }
        }

        public void paintDot(Graphics2D g){
                graphics2D = g;
                g.fillOval(    (int)(vertice.x - r/2), (int)(vertice.y - (int)(r/2)),
                                        2*(int)r, 2*(int)r);
        }

        public void run(){
                while(true){
                        updateRadius();
                        paintDot(graphics2D);
                }
        }

}
```

DrawableContourController.java

```
import javax.swing.event.ChangeListener;
import javax.swing.event.ChangeEvent;

import FuncLib.*;

public class DrawableContourController implements ChangeListener{

        DrawableContourModel model;

        public DrawableContourController(DrawableContourModel model){
                this.model = model;
        }

        public void stateChanged(ChangeEvent event){
                        double R = ((MyJSliderDouble)event.getSource()).DgetValue();
                        model.setR((float) R);
        }
}
```

Вывод: в ходе выполнения данной лабораторной работы я ознакомился с Swing API и многопоточностью, а также с некоторыми паттернами создания графических приложений, например, MVC.