

Федеральное агентство связи
Ордена Трудового Красного Знамени
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский технический университет связи и информатики»
Кафедра Информатики



Отчет по лабораторной работе №5
по предмету «КТП»:

Выполнил: студент группы БВТ1802

Самаков Владислав Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

1 Цель работы

Цель работы: изучить алгоритм расчета фрактала, а также познакомиться с java.swing.

2 Задание

Добавить два новых типа фракталов и функцию сохранения изображения.

3 Текст программы

FractalExplorer.java

```
import javax.swing.*;
import java.awt.*;
import java.awt.geom.Rectangle2D;
import java.awt.event.*;
import java.io.IOException;
import javax.swing.filechooser.*;
import javax.imageio.ImageIO;

public class FractalExplorer {
    private int displaySize;
    private FractalGenerator generator;
    private Rectangle2D.Double range;

    private JFrame frame;
    private JImageDisplay display;
    private JButton resetButton;
    private JComboBox switchButton;
    private JButton saveButton;

    public static void main(String args[]) {
        FractalExplorer explorer = new FractalExplorer(800);
        explorer.createAndShowGUI();
        explorer.drawFractal();
    }

    private class ActionListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {

            if (actionEvent.getSource() == resetButton) {
                generator.getInitialRange(range);
                drawFractal();
            }
            else if (actionEvent.getSource() == switchButton) {
                generator = (FractalGenerator) switchButton.getSelectedItem();
                generator.getInitialRange(range);
                drawFractal();
            }
            else if (actionEvent.getSource() == saveButton) {
                JFileChooser fileChooser = new JFileChooser();
                FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");
```

```

        fileChooser.setFileFilter(filter);
        fileChooser.setAcceptAllFileFilterUsed(false);
        if (fileChooser.showSaveDialog(null) == JFileChooser.APPROVE_OPTION)
    {
        try {
            ImageIO.write(display.img, "png",
fileChooser.getSelectedFile());
        } catch (IOException e) {
            JOptionPane.showMessageDialog(frame, e.getMessage(),
"Something went wrong",
                                JOptionPane.ERROR_MESSAGE);
        }
    }
}

private class MouseListener extends MouseAdapter {
    @Override
    public void mouseClicked(MouseEvent e) {
        int x = e.getX();
        int y = e.getY();

        double xCoord = generator.getCoord(range.x, range.x + range.width,
displaySize,x);
        double yCoord = generator.getCoord(range.y, range.y + range.height,
displaySize,y);
        generator.recenterAndZoomRange(range, xCoord, yCoord, 0.5);
        drawFractal();
    }
}

public FractalExplorer(int ScreenSize) {
    displaySize = ScreenSize;
    range = new Rectangle2D.Double();
    generator = new Mandelbrot();
    generator.getInitialRange(range);
}

private void drawFractal() {
    for (int x = 0; x < displaySize; x++)
    {
        for (int y = 0; y < displaySize; y++)
        {
            double xCoord = FractalGenerator.getCoord
                (range.x, range.x + range.width, displaySize, x);
            double yCoord = FractalGenerator.getCoord
                (range.y, range.y + range.height, displaySize, y);
            int IterNum = generator.numIterations(xCoord, yCoord);
            if (IterNum == -1) display.drawPixel(x, y, 0);
            else {
                float hue = 0.7f + (float) IterNum / 200f;
                int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
                display.drawPixel(x, y, rgbColor);
            }
        }
    }
    display.repaint();
}
}

```

```

public void createAndShowGUI() {

    // setting up a panel with switch and label
    JPanel panel = new JPanel();
    switchButton = new JComboBox();

    switchButton.addItem(new Mandelbrot());
    switchButton.addItem(new Tricorn());
    switchButton.addItem(new BurningShip());
    switchButton.addActionListener(new ActionListener());

    JLabel label = new JLabel("Fractal type:");
    panel.add(label);
    panel.add(switchButton);

    display = new JImageDisplay(displaySize, displaySize);
    display.addMouseListener(new MouseListener());

    resetButton = new JButton("Reset");
    resetButton.addActionListener(new ActionListener());
    saveButton = new JButton("Save Image");
    saveButton.addActionListener(new ActionListener());
    JPanel panel2 = new JPanel();
    panel2.add(resetButton);
    panel2.add(saveButton);

    frame = new JFrame();
    frame.getContentPane().add(panel, BorderLayout.NORTH);
    frame.getContentPane().add(display, BorderLayout.CENTER);
    frame.getContentPane().add(panel2, BorderLayout.SOUTH);
    frame.setDefaultCloseOperation(JFrame.DISPOSE_ON_CLOSE);

    frame.pack();
    frame.setVisible(true);
    frame.setResizable(true);
}
}

```

Tricorn.java

```

import java.awt.geom.Rectangle2D;

public class Tricorn extends FractalGenerator {
    public static final int MAX_ITERATIONS = 2000;

    public void getInitialRange(Rectangle2D.Double range) {
        range.setRect(-2, -2, 4, 4);
    }

    public int numIterations(double x, double y) {
        Complex z = new Complex(0, 0);
        Complex c = new Complex(x, y);

        for (int IterNum = 0; IterNum < MAX_ITERATIONS; IterNum++) {
            z = z.sopr().mul(z.sopr()).sum(c);
            if (z.abs() > 4) return IterNum;
        }
        return -1;
    }
}

```

```

    @Override
    public String toString() {
        return "Tricorn";
    }
}

```

BurningShip.java

```

import java.awt.geom.Rectangle2D;

public class BurningShip extends FractalGenerator{
    public static final int MAX_ITERATIONS = 2000;

    public void getInitialRange(Rectangle2D.Double range) {
        range.setRect(-2, -2.5, 4,4);
    }

    @Override
    public String toString() {
        return "Burning Ship";
    }

    public int numIterations(double x, double y) {
        Complex z = new Complex(0, 0);
        Complex c = new Complex(x, y);

        double realTemp;
        for (int IterNum = 0; IterNum < MAX_ITERATIONS; IterNum++) {
            realTemp = z.real*z.real - z.imag*z.imag + x;
            z.imag = Math.abs(2*z.real*z.imag + y);
            z.real = Math.abs(realTemp);
            if (z.abs() > 4) return IterNum;
        }
        return -1;
    }
}

```

Complex.java

```

public class Complex {
    public double real, imag;
    public Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }
    public double abs() { return real * real + imag * imag; }

    public Complex sum(Complex c) {
        return new Complex(this.real + c.real, this.imag + c.imag);
    }

    public Complex mul(Complex c) {
        double real = this.real * c.real - this.imag * c.imag;
        double imag = this.real * c.imag + this.imag * c.real;
        return new Complex(real,imag);
    }
}

```

```
}  
  
public Complex sopr() { return new Complex(this.real, -this.imag); }  
}
```

4 Работа программы





