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 №4

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

Дополнить исходный текст программы таким образом, чтобы она выводила окно с фракталом и интерфейсом работы с ним.

3 Текст программы

FractalExplorer.java

```java
import javax.swing.*;
import java.awt.*;
import java.awt.geom.Rectangle2D;
import java.awt.event.*;

public class FractalExplorer {
    private int displaySize;
    private JImageDisplay display;
    private JButton button;
    private JFrame frame;
    private FractalGenerator generator;
    private Rectangle2D.Double range;


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



    public FractalExplorer(int ScreenSize) {
        displaySize = ScreenSize;

        range = new Rectangle2D.Double();
        generator = new Mandelbrot();
        generator.getInitialRange(range);
    }

    public void createAndShowGUI() {
        frame = new JFrame();

        button = new JButton("reset");
        button.addActionListener(new actionListener());
        frame.getContentPane().add(button, BorderLayout.SOUTH);

        display = new JImageDisplay(displaySize, displaySize);
        display.addMouseListener(new MouseListener());
        frame.getContentPane().add(display, BorderLayout.CENTER);

        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
        frame.pack();
        frame.setVisible(true);
        frame.setResizable(true);
    }

    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 numIter = generator.numIterations(xCoord, yCoord);
                if (numIter == -1) display.drawPixel(x, y, 0);
                else {
                    float hue = 0.7f + (float) numIter / 200f;
                    int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
                    display.drawPixel(x, y, rgbColor);
                }
            }
        }
        display.repaint();
    }

    private class actionListener implements ActionListener {
        @Override
        public void actionPerformed(ActionEvent actionEvent) {
            generator.getInitialRange(range);
            drawFractal();
        }
    }

    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();
        }
    }

}
```

FractalGenerator.java

```java
import java.awt.geom.Rectangle2D;


/**
 * This class provides the common interface and operations for fractal
 * generators that can be viewed in the Fractal Explorer.
 */
public abstract class FractalGenerator {

    /**
     * This static helper function takes an integer coordinate and converts it
     * into a double-precision value corresponding to a specific range.  It is
     * used to convert pixel coordinates into double-precision values for
     * computing fractals, etc.
     *
     * @param rangeMin the minimum value of the floating-point range
     * @param rangeMax the maximum value of the floating-point range
     *
     * @param size the size of the dimension that the pixel coordinate is from.
     *        For example, this might be the image width, or the image height.
     *
     * @param coord the coordinate to compute the double-precision value for.
     *        The coordinate should fall in the range [0, size].
     */
    public static double getCoord(double rangeMin, double rangeMax,
                                  int size, int coord) {

        assert size > 0;
        assert coord >= 0 && coord < size;

        double range = rangeMax - rangeMin;
        return rangeMin + (range * (double) coord / (double) size);
    }


    /**
     * Sets the specified rectangle to contain the initial range suitable for
     * the fractal being generated.
     */
    public abstract void getInitialRange(Rectangle2D.Double range);


    /**
     * Updates the current range to be centered at the specified coordinates,
     * and to be zoomed in or out by the specified scaling factor.
     */
    public void recenterAndZoomRange(Rectangle2D.Double range,
                                     double centerX, double centerY, double scale) {

        double newWidth = range.width * scale;
        double newHeight = range.height * scale;

        range.x = centerX - newWidth / 2;
        range.y = centerY - newHeight / 2;
        range.width = newWidth;
        range.height = newHeight;
    }


    /**
     * Given a coordinate <em>x</em> + <em>iy</em> in the complex plane,
     * computes and returns the number of iterations before the fractal
     * function escapes the bounding area for that point.  A point that
     * doesn't escape before the iteration limit is reached is indicated
     * with a result of -1.
     */
    public abstract int numIterations(double x, double y);
}
```

Mandelbrot.java

```java
import java.awt.geom.Rectangle2D;

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

    public void getInitialRange(Rectangle2D.Double rect) {
        rect.setRect(-2, -1.5, 3,3);
    }

    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.times(z).sum(c);
            if (z.abs() > 4) return IterNum;
        }
        return -1;
    }
}
```

JImageDisplay.java

```java
import java.awt.*;
import java.awt.image.BufferedImage;
public class JImageDisplay extends javax.swing.JComponent {
    private BufferedImage img;

    public JImageDisplay(int width, int height) {
        img = new java.awt.image.BufferedImage(width, height,
BufferedImage.TYPE_INT_RGB);
        super.setPreferredSize(new Dimension(width, height));
    }

    @Override
    protected void paintComponent(Graphics g) {
        super.paintComponent(g);
        g.drawImage(img, 0, 0, img.getWidth(), img.getHeight(), null);
    }

    public void clearImage() {
        for (int i = 0; i < img.getWidth(); i++) {
            for (int j = 0; j < img.getHeight(); j++) {
                img.setRGB(i, j, 0);
            }
        }
    }

    public void drawPixel(int x, int y, int rgbColor) {
        img.setRGB(x, y, rgbColor);
    }
}
```

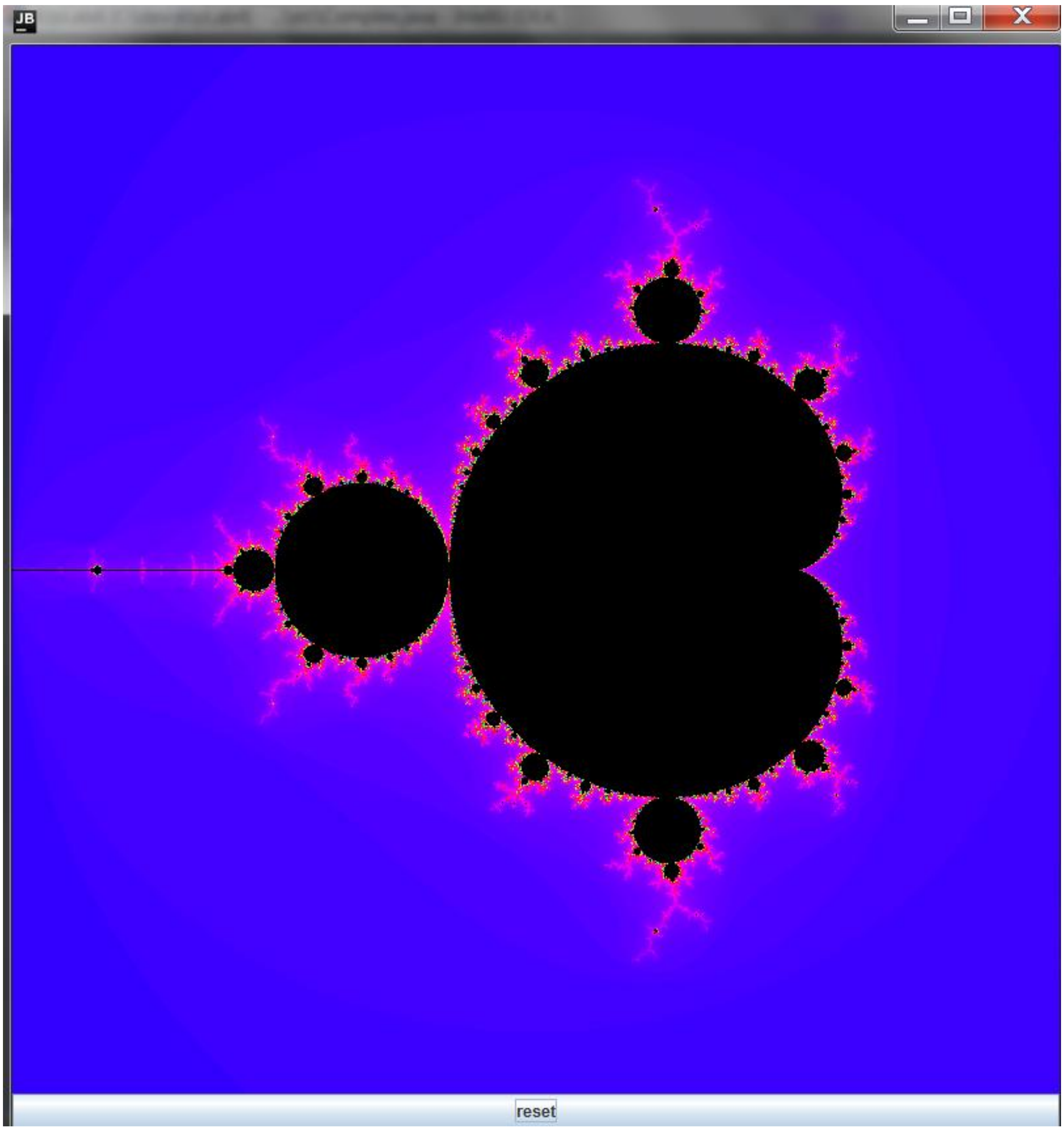
Complex.java

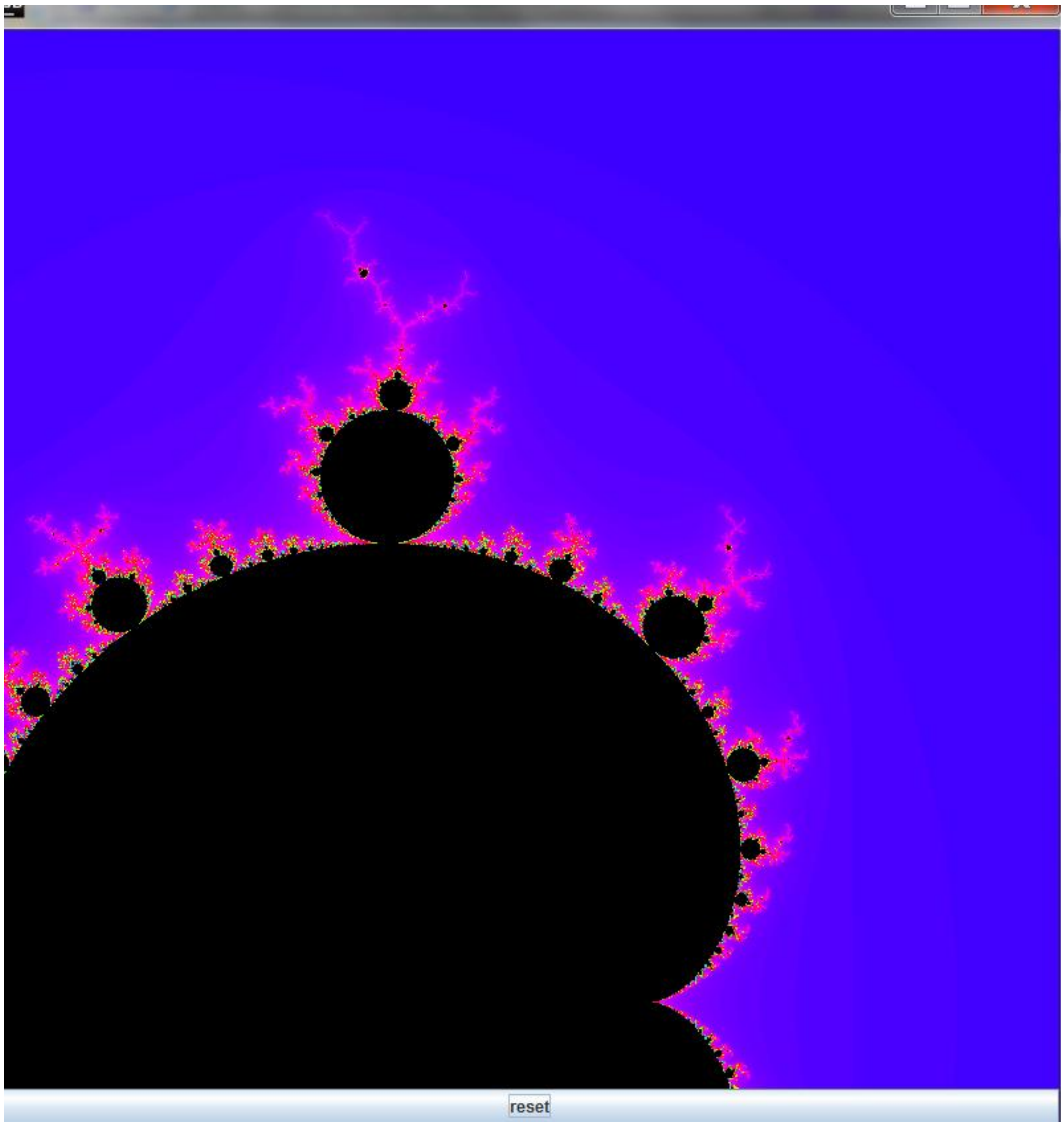
```java
public class Complex {
    private double real, imag;
    Complex(double real, double imag) {
        this.real = real;
        this.imag = imag;
    }
    public double abs() { return real * real + imag * imag; }

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

    public Complex times(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);
    }
}
```

4 Работа программы


reset


reset