**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

Кафедра «Математическая кибернетика и информационные технологии»

**Отчет по лабораторной работе №4**

по дисциплине «Информационные технологии и программирование»

на тему:

Рисование фракталов

Выполнил: студент группы БФИ2101

Иванов Глеб

Проверил преподаватель:

Мосева Марина Сергеевна

Москва, 2022

**Оглавление**

[**Поставленная задача** 3](#_Toc122868185)

[**Скриншоты выполненного задания** 4](#_Toc122868186)

[**Код** 5](#_Toc122868187)

[**Вывод** 10](#_Toc122868188)

[**Список источников** 11](#_Toc122868189)

**Поставленная задача**

1. Создать пользовательский интерфейс для отображения фрактала.

2. Реализовать метод для приближения отдельного участка картинки.

**Скриншоты выполненного задания**

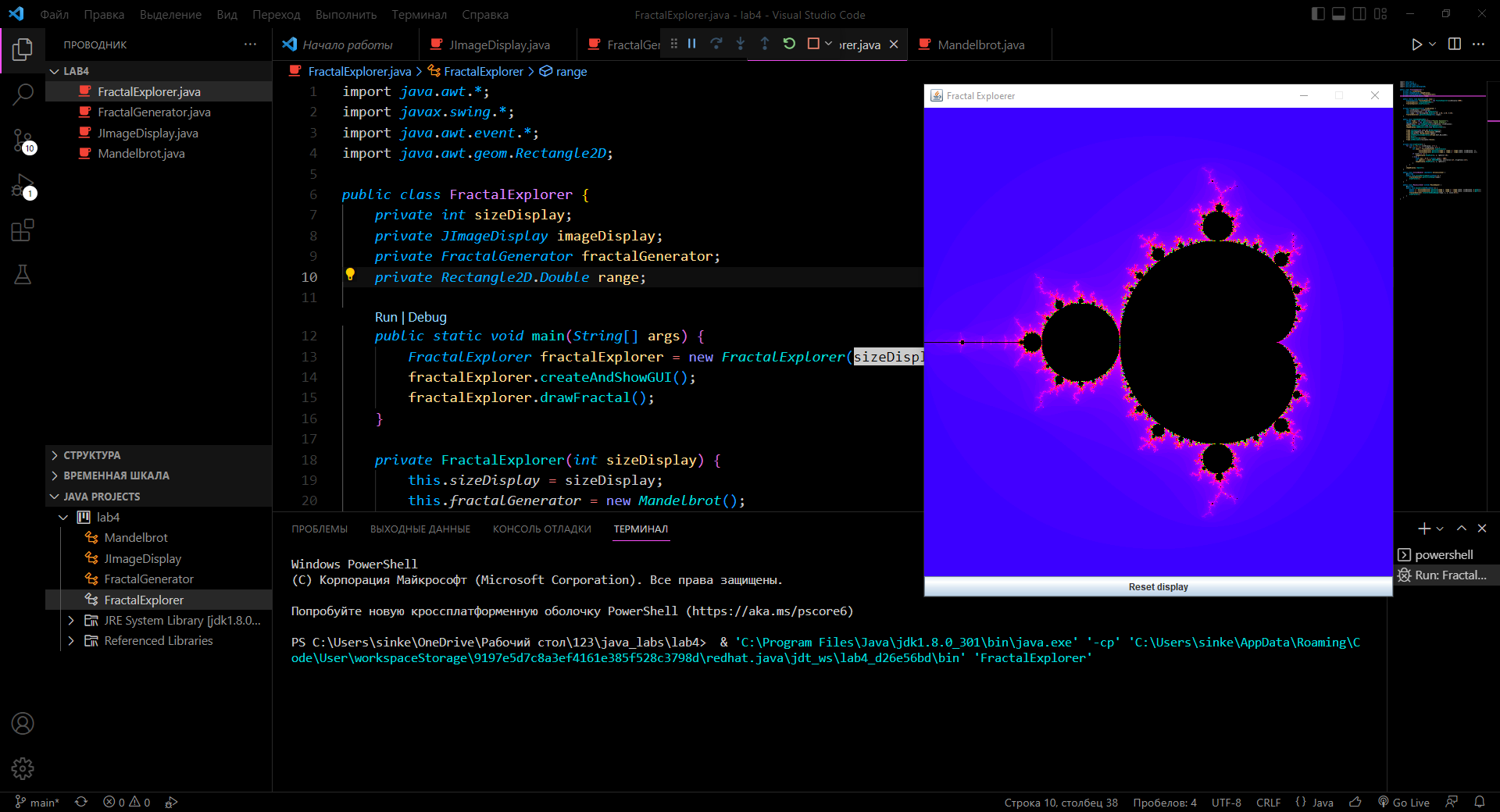


Рисунок 1 – Отображение пользовательского интерфейса

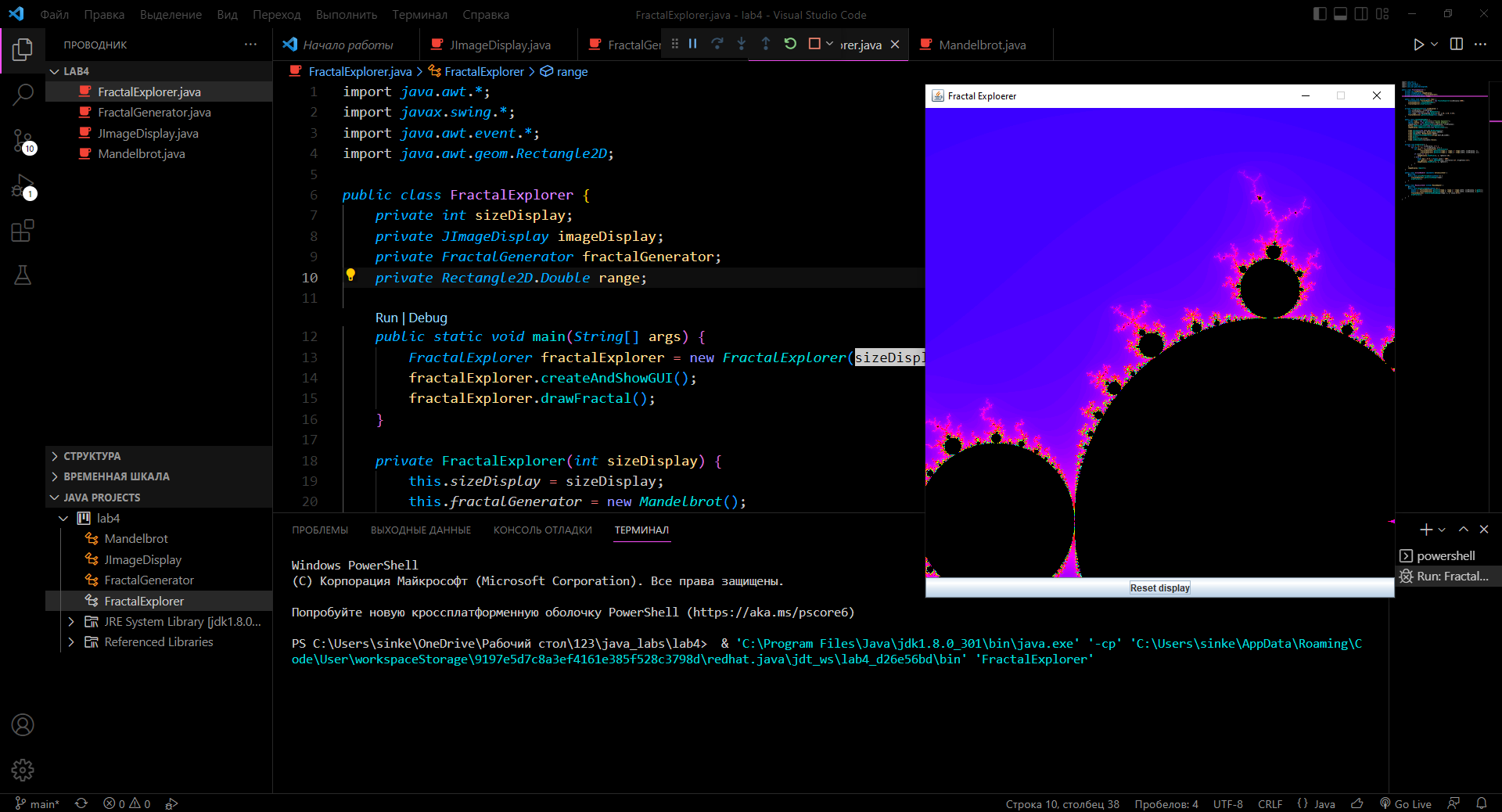


Рисунок 2 – Демонстрация метода для приближения отдельного участка изображения.

**Код**

import *java*.*awt*.\*;

import *javax*.*swing*.\*;

import *java*.*awt*.*event*.\*;

import *java*.*awt*.*geom*.*Rectangle2D*;

*public* *class* FractalExplorer {

*private* *int* sizeDisplay;

*private* *JImageDisplay* imageDisplay;

*private* *FractalGenerator* fractalGenerator;

*private* *Rectangle2D*.*Double* range;

*public* *static* *void* main(*String*[] args) {

*FractalExplorer* fractalExplorer = new *FractalExplorer*(600);

        fractalExplorer.createAndShowGUI();

        fractalExplorer.drawFractal();

    }

*private* FractalExplorer(*int* sizeDisplay) {

        this.*sizeDisplay* = sizeDisplay;

        this.*fractalGenerator* = new *Mandelbrot*();

        this.*range* = new Rectangle2D.Double(0, 0, 0, 0);

        fractalGenerator.getInitialRange(this.*range*);

    }

*public* *void* createAndShowGUI() {

*JFrame* frame = new *JFrame*("Fractal Exploerer");

*JButton* button = new *JButton*("Reset display");

        imageDisplay = new *JImageDisplay*(*sizeDisplay*, *sizeDisplay*);

        button.addActionListener(new *ActionHandler*());

        imageDisplay.addMouseListener(new *MouseListener*());

        frame.setLayout(new java.*awt*.BorderLayout());

        frame.add(imageDisplay, BorderLayout.*CENTER*);

        frame.add(button, BorderLayout.*SOUTH*);

        frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);

        frame.pack();

        frame.setVisible(true);

        frame.setResizable(false);

    }

*private* *void* drawFractal() {

        for (*int* x = 0; x < sizeDisplay; x++) {

            for (*int* y = 0; y < sizeDisplay; y++) {

*int* count = fractalGenerator.numIterations(

                        FractalGenerator.getCoord(range.*x*, range.*x* + range.*width*, sizeDisplay, x),

                        FractalGenerator.getCoord(range.*y*, range.*y* + range.*width*, sizeDisplay, y));

                if (count == -1) {

                    imageDisplay.drawPixel(x, y, 0);

                } else {

*float* hue = 0.7f + (*float*) count / 200f;

*int* rgbColor = Color.HSBtoRGB(hue, 1f, 1f);

                    imageDisplay.drawPixel(x, y, rgbColor);

                }

            }

        }

        imageDisplay.repaint();

    }

*public* *class* ActionHandler *implements* ActionListener {

        @*Override*

*public* *void* actionPerformed(*ActionEvent* e) {

            fractalGenerator.getInitialRange(range);

*drawFractal*();

        }

    }

*public* *class* MouseListener *extends* MouseAdapter {

        @*Override*

*public* *void* mouseClicked(*MouseEvent* e) {

*double* x = FractalGenerator.getCoord(range.*x*, range.*x* + range.*width*, sizeDisplay, e.getX());

*double* y = FractalGenerator.getCoord(range.*y*, range.*y* + range.*width*, sizeDisplay, e.getY());

            fractalGenerator.recenterAndZoomRange(range, x, y, 0.5);

*drawFractal*();

        }

    }

}

Код файла FractalExplorer.java

import *java*.*awt*.*geom*.*Rectangle2D*;

*public* *abstract* *class* FractalGenerator {

*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);

    }

*public* *abstract* *void* getInitialRange(*Rectangle2D*.*Double* range);

*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;

    }

*public* *abstract* *int* numIterations(*double* x, *double* y);

}

Код файла FractalGenerator.java

import *javax*.*swing*.\*;

import *java*.*awt*.\*;

import *java*.*awt*.*image*.*BufferedImage*;

*public* *class* JImageDisplay *extends* JComponent {

*private* *BufferedImage* image;

*public* JImageDisplay(*int* width, *int* height) {

        image = new *BufferedImage*(*width*, *height*,BufferedImage.*TYPE\_INT\_RGB*);

*Dimension* dimension = new *Dimension*(*width*, *height*);

        super.setPreferredSize(dimension);

    }

*public* *void* paintComponent(*Graphics* g) {

        super.paintComponent(g);

        g.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), null);

    }

*public* *void* ImageClear() {

        for (*int* i = 0; i < image.getWidth(); i++) {

            for (*int* j = 0; j < image.getHeight(); j++) {

*drawPixel*(*i*, *j*,0);

            }

        }

    }

*public* *void* drawPixel(*int* x, *int* y, *int* rgbColor) {

        image.setRGB(x, y, rgbColor);

    }

}

Код файла JImageDisplay.java

import *java*.*awt*.*geom*.*Rectangle2D*;

*public* *class* Mandelbrot *extends* FractalGenerator {

*public* *void* getInitialRange(*Rectangle2D*.*Double* range) {

        range.*x* = -2;

        range.*y* = -1.5;

        range.*height* = 3;

        range.*width* = 3;

    }

*public* *static* *final* *int* MAX\_ITERATIONS = 2000;

    @*Override*

*public* *int* numIterations(*double* x, *double* y) {

*double* i = y;

*double* x1 = x;

*int* count = 0;

        while (count < MAX\_ITERATIONS) {

            count++;

*double* a = x1 \* x1 - i \* i + x;

*double* b = 2 \* x1 \* i + y;

            x1 = a;

            i = b;

            if (x1 \* x1 + i \* i > 4)

                break;

        }

        if (count == MAX\_ITERATIONS)

            return -1;

        else

            return count;

    }

}

Код файла Mandelbrot.java

**Вывод**

В ходе работы данной лабораторной я работал с библиотекой java.swing. Смог реализовать собственный интерфейс. Узнал о базовых элементах управления.

**Список источников**

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа, 2006.

2. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.