**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 «Кроссплатформенные технологии программирования»

на тему:

«Рисование фракталов»

Выполнил: студент группы БВТ 1802

Ганюшкина Елизавета Сергеевна

Руководитель:

Полянцева Ксения Андреевна

Москва 2020

**Цель работы**

В следующей лабораторной работе надо создать небольшое JAVA-приложение, которое сможет рисовать фракталы. Это будет сделано с помощью фреймворка Swing и Java API, который позволяет создавать графические пользовательские интерфейсы.

**Задания**

Создать класс FractalExplorer, который позволит исследовать различные области фрактала, путем его создания, отображения через графический интерфейс Swing и обработки событий, вызванных взаимодействием приложения с пользователем.

* Класс FractalExplorer должен отслеживать несколько важных полей для состояния программы:
  + Целое число «размер экрана», которое является шириной и высотой отображения в пикселях.
  + Ссылка JImageDisplay, для обновления отображения в разных методах в процессе вычисления фрактала.
  + Объект FractalGenerator. Будет использоваться ссылка на базовый класс для отображения других видов фракталов в будущем.
  + Объект Rectangle2D.Double, указывающий диапазона комплексной плоскости, которая выводится на экран. Все вышеприведенные поля будут иметь тип доступа private.
* У класса должен быть конструктор, который принимает значение размера отображения в качестве аргумента, затем сохраняет это значение в соответствующем поле, а также инициализирует объекты диапазона и фрактального генератора. Данный конструктор не должен устанавливать какие-либо компоненты Swing.
* Создать метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и кнопку для сброса отображения. Используйте java.awt.BorderLayout для содержимого окна; добавить объект отображения изображения в позицию BorderLayout.CENTER и кнопку в позицию BorderLayout.SOUTH.
* Реализовать вспомогательный метод с типом доступа private для вывода на экран фрактала.
* Создать внутренний класс для обработки событий java.awt.event.ActionListener от кнопки сброса. Обработчик должен сбросить диапазон к начальному, определенному генератором, а затем перерисовать фрактал.
* Создать другой внутренний класс для обработки событий java.awt.event.MouseListener с дисплея.
* Необходимо создать статический метод main() для FractalExplorer так, чтобы можно было его запустить.

**Коды программы**

*FractalExplorer.java*

import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import javax.swing.filechooser.\*;  
import java.awt.image.\*;  
  
public class FractalExplorer{  
private JButton resetButton;  
private int rowsRemaining;  
private int displaySize;  
private JImageDisplay display;  
private FractalGenerator fractal;  
private Rectangle2D.Double range;

public FractalExplorer(int size) {  
displaySize = size;  
fractal = new Mandelbrot();  
range = new Rectangle2D.Double();  
fractal.getInitialRange(range);  
display = new JImageDisplay(displaySize, displaySize);}

public void createAndShowGUI(){  
display.setLayout(new BorderLayout());  
JFrame myFrame = new JFrame("Fractal Explorer");  
myFrame.add(display, BorderLayout.CENTER);  
MouseHandler click = new MouseHandler();  
display.addMouseListener(click);  
myFrame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
resetButton = new JButton("Reset");  
ButtonHandler resetHandler = new ButtonHandler();  
resetButton.addActionListener(resetHandler);  
JPanel myBottomPanel = new JPanel();  
myBottomPanel.add(resetButton);  
myFrame.add(myBottomPanel, BorderLayout.SOUTH);  
myFrame.pack();  
myFrame.setVisible(true);  
myFrame.setResizable(false);}  
private void drawFractal(){  
enableUI(false);  
rowsRemaining = displaySize;  
for (int x=0; x<displaySize; x++){  
FractalWorker drawRow = new FractalWorker(x);  
drawRow.execute();}}  
private void enableUI(boolean val){  
resetButton.setEnabled(val);}  
private class ButtonHandler implements ActionListener{  
public void actionPerformed(ActionEvent e){  
String command = e.getActionCommand();  
if (e.getSource() instanceof JComboBox) {  
JComboBox mySource = (JComboBox) e.getSource();  
fractal = (FractalGenerator) mySource.getSelectedItem();  
fractal.getInitialRange(range);  
drawFractal();}  
else if (command.equals("Reset")) {  
fractal.getInitialRange(range);  
drawFractal();}}}  
private class MouseHandler extends MouseAdapter{  
public void mouseClicked(MouseEvent e){  
if (rowsRemaining != 0) {  
return;}  
int x = e.getX();  
double xCoord = fractal.getCoord(range.x,  
range.x + range.width, displaySize, x);  
int y = e.getY();  
double yCoord = fractal.getCoord(range.y,  
range.y + range.height, displaySize, y);  
fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);

drawFractal();}}  
private class FractalWorker extends SwingWorker<Object, Object>{  
int yCoordinate;  
int[] computedRGBValues;  
private FractalWorker(int row) {  
yCoordinate = row;}  
// метод, выполняющий фоновые операции  
protected Object doInBackground() {  
computedRGBValues = new int[displaySize];  
for (int i = 0; i < computedRGBValues.length; i++) {  
double xCoord = fractal.getCoord(range.x,  
range.x + range.width, displaySize, i);  
double yCoord = fractal.getCoord(range.y,  
range.y + range.height, displaySize, yCoordinate);  
int iteration = fractal.numIterations(xCoord, yCoord);  
if (iteration == -1){  
computedRGBValues[i] = 0;}  
else {  
float hue = 0.7f + (float) iteration / 200f;

int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);  
computedRGBValues[i] = rgbColor;}}  
return null;}  
// вызывается при завершении фоновой задачи  
protected void done() {  
for (int i = 0; i < computedRGBValues.length; i++) {  
display.drawPixel(i, yCoordinate, computedRGBValues[i]);}

display.repaint(0, 0, yCoordinate, displaySize, 1);  
rowsRemaining--;  
if (rowsRemaining == 0) {  
enableUI(true);}}}  
public static void main(String[] args){  
FractalExplorer displayExplorer = new FractalExplorer(600);  
displayExplorer.createAndShowGUI();  
displayExplorer.drawFractal();}}

*FractalGenerator.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);}

*JImageDisplay.java*

import javax.swing.\*;

import java.awt.image.\*;

import java.awt.\*;

class JImageDisplay extends JComponent{

private BufferedImage displayImage;

public BufferedImage getImage() {

return displayImage;}

public JImageDisplay(int width, int height) {

displayImage = new BufferedImage(width, height,

BufferedImage.TYPE\_INT\_RGB);

Dimension imageDimension = new Dimension(width, height);

super.setPreferredSize(imageDimension);}

public void paintComponent(Graphics g){

super.paintComponent(g);

g.drawImage(displayImage, 0, 0, displayImage.getWidth(),

displayImage.getHeight(), null);}

public void clearImage(){

int[] blankArray = new int[getWidth() \* getHeight()];

displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);}

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

displayImage.setRGB(x, y, rgbColor);}}

*Mandelbrot.java*

import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator{

public static final int MAX\_ITERATIONS = 2000;

public void getInitialRange(Rectangle2D.Double range){

range.x = -2;

range.y = -1.5;

range.width = 3;

range.height = 3;}

public int numIterations(double x, double y){

int iteration = 0;

double zreal = 0;

double zimaginary = 0;

while (iteration < MAX\_ITERATIONS &&

zreal \* zreal + zimaginary \* zimaginary < 4){

double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;

double zimaginaryUpdated = 2 \* zreal \* zimaginary + y;

zreal = zrealUpdated;

zimaginary = zimaginaryUpdated;

iteration += 1;}

if (iteration == MAX\_ITERATIONS){

return -1;}

return iteration;}

public String toString() {

return "Mandelbrot";}}

**Результат работы кода**

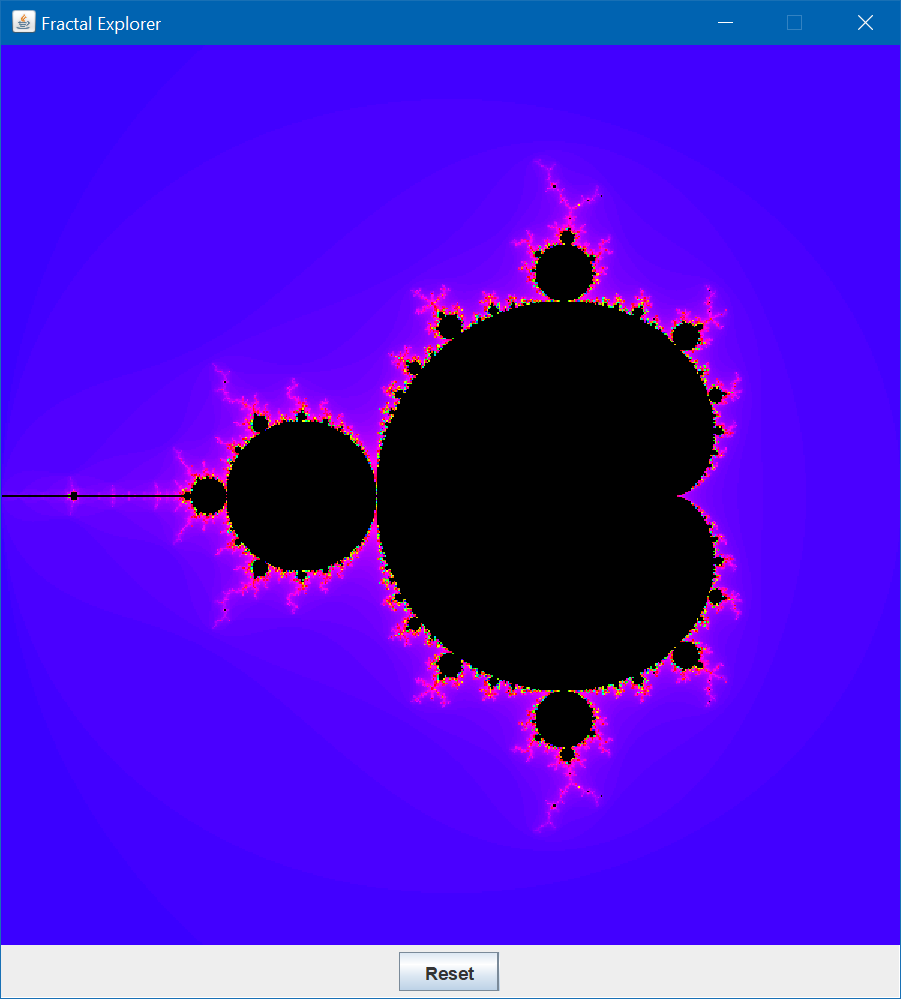


Рисунок 1 - Результат работы программы

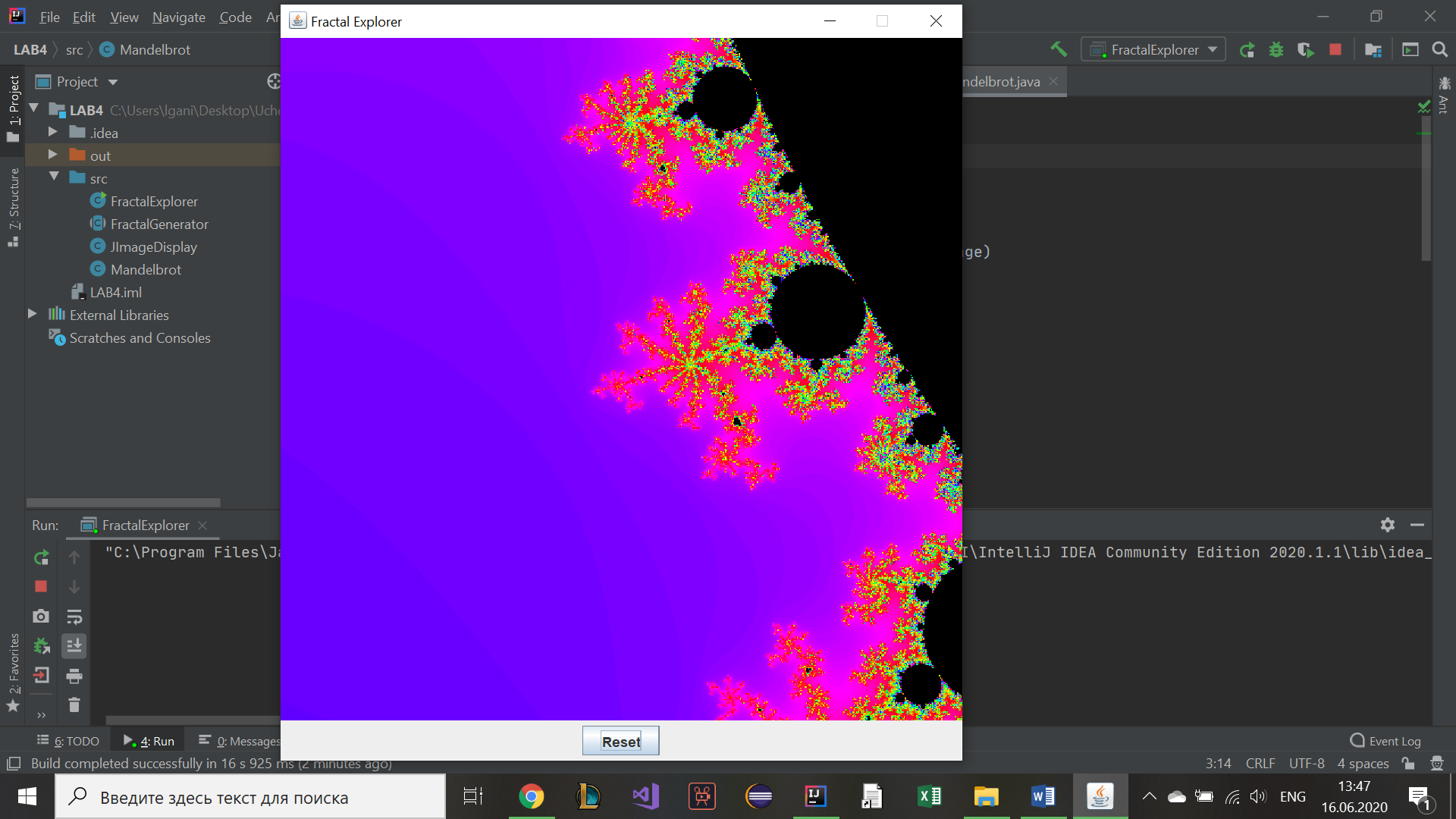


Рисунок 2 - Результат работы программы

**Вывод**

В данной лабораторной работе было создано небольшое приложение по рисованию фракталов. Оно было сделано с помощью фреймворка Swing и Java API. Был создан класс FractalExplorer, который позволил исследовать различные области фрактала, путем его создания, отображения через графический интерфейс Swing. Класс FractalExplorer отслеживал несколько важных полей по состоячнию программы. Создан метод createAndShowGUI ().