Министерство цифрового развития и массовых коммуникаций

Российской Федерации Ордена Трудового Красного Знамени

федеральное государственное образовательное бюджетное учреждение высшего профессионального образования

«Московский технический университет связи и информатики»

**Кафедра электроники**

**Отчет по лабораторной работе №4**

**По дисциплине «Кроссплатформенные технологии программирования» по теме:** **«Рисование фракталов»**

Выполнил: студент

группы БСТ2001

Ибодуллоев У.Х.

Проверила:

Ст. преп. Полянцева К. А.

Москва

1. Цель работы:

Создать приложение, которое сможет рисовать фракталы.

1. Задание:
2. Создать пользовательский интерфейс JImageDisplay.java
3. Вычислить фрактал Мандельброта FractalGenerator.java, Mandelbrot.java (наследник FractalGenerator.java)
4. Создать класс FractalExplorer.java, который позволит исследовать различные области фрактала, путем его создания, отображения через графический интерфейс Swing и обработки событий, вызванных взаимодействием приложения с пользователем
5. Код программы:

Ссылка на GitHub: <https://github.com/1Double/MTUCI/tree/main/Term_4/KTP/La>b4

1. Листинг файла JImageDisplay.java

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
  
public class JImageDisplay extends JComponent {  
 private BufferedImage image;  
  
 // объявление изображения и его параметров  
 public JImageDisplay(int width, int height) {  
 image = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension dimension = new Dimension(width, height);  
 super.setPreferredSize(dimension);  
 }  
  
 // метод отрисовки изображения  
 public void paintComponent(Graphics graphics) {  
 super.paintComponent(graphics);  
 graphics.drawImage(image, 0, 0, image.getWidth(), image.getHeight(), null);  
 }  
  
 // очистка изображения (устанавливает все пиксели  
 // в черный цвет)  
 public void clearImage() {  
 for (int i = 0; i < image.getWidth(); i++) {  
 for (int j = 0; j < image.getHeight(); j++) {  
 drawPixel(i, j, 0);  
 }  
 }  
 }  
  
 // задание цвета конкретному пикселю  
 public void drawPixel(int x, int y, int rgbColor) {  
 image.setRGB(x, y, rgbColor);  
 }  
  
 public BufferedImage getImage() {  
 return image;  
 }  
}

1. Листинг файла FractalGenerator.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);  
}

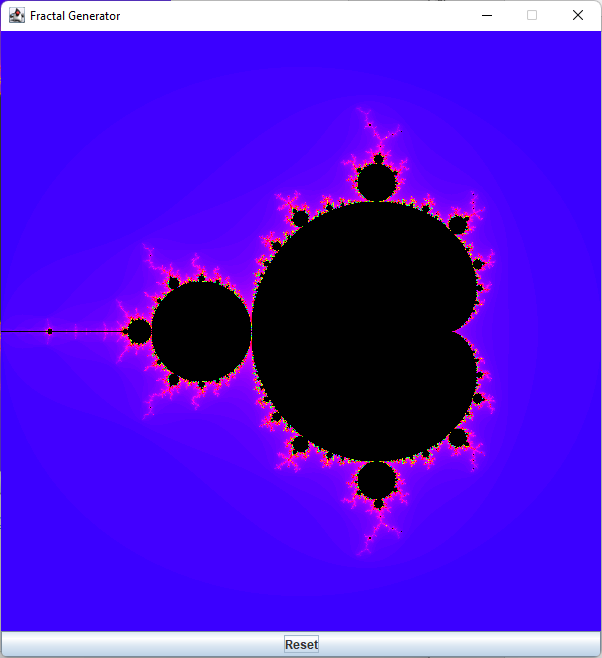
1. Листинг файла Mandelbrot.java

import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator {  
  
 public static final int *MAX\_ITERATIONS* = 2000;  
  
 @Override  
 public void getInitialRange(Rectangle2D.Double range) {  
 range.x = -2;  
 range.y = -1.5;  
 range.height = 3;  
 range.width = 3;  
 }  
  
 @Override  
 public int numIterations(double x, double y) {  
 double r = x;  
 double i = y;  
 int counter = 0;  
 while (counter < *MAX\_ITERATIONS*) {  
 counter++;  
 double k = r \* r - i\*i+x;  
 double m = 2 \* r \* i + y;  
 r = k;  
 i = m;  
 if (r\*r+i\*i > 4)  
 break;  
 }  
 if (counter == *MAX\_ITERATIONS*)  
 return -1;  
 return counter;  
 }  
}

1. Листинг файла FractalExplorer.java

import java.awt.\*;  
import javax.swing.\*;  
import javax.swing.border.Border;  
import java.awt.event.\*;  
import java.awt.geom.Rectangle2D;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JImageDisplay imageDisplay;  
 private FractalGenerator fractalGenerator;  
 private Rectangle2D.Double range;  
  
 private FractalExplorer (int displaySize) {  
 this.displaySize = displaySize;  
 this.fractalGenerator = new Mandelbrot();  
 this.range = new Rectangle2D.Double(0,0,0,0);  
 fractalGenerator.getInitialRange(this.range);  
 }  
  
 public static void main(String[] args) {  
 FractalExplorer fractalExplorer = new FractalExplorer(600);  
 fractalExplorer.setGUI();  
 fractalExplorer.drawFractal();  
 }  
  
 // задание интерфейса  
 public void setGUI() {  
 JFrame frame = new JFrame("Fractal Generator");  
 JButton button = new JButton("Reset");  
  
 imageDisplay = new JImageDisplay(displaySize, displaySize);  
 imageDisplay.addMouseListener(new MouseListener());  
  
 button.addActionListener(new ActionHandler());  
  
 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);  
 }  
  
 // отрисовка фрактала в JImageDisplay  
 private void drawFractal() {  
 for (int x = 0; x < displaySize; x++) {  
 for (int y = 0; y < displaySize; y++) {  
 int counter = fractalGenerator.numIterations(FractalGenerator.*getCoord*(range.x, range.x + range.width, displaySize, x),  
 fractalGenerator.*getCoord*(range.y, range.y + range.width, displaySize, y));  
 if (counter == -1) {  
 imageDisplay.drawPixel(x, y, 0);  
 }  
 else {  
 float hue = 0.7f + (float) counter / 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, displaySize, e.getX());  
 double y = FractalGenerator.*getCoord*(range.y, range.y + range.width, displaySize, e.getY());  
 fractalGenerator.recenterAndZoomRange(range, x, y, 0.5);  
 drawFractal();  
 }  
 }  
}

1. Выполнение:



1. Выводы

В ходе работы я научился создавать фракталы и выводить их в графический интерфейс программы.