Министерство цифрового развития, связи и массовых коммуникаций Российской Федерации

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский технический университет связи и информатики» (МТУСИ)

Кафедра «Математической кибернетики и информационных технологий»

Лабораторная работа №8

По дисциплине: «Java - программирование»

По теме: «Фракталы»

Выполнил студент

Группы БСУ1901:

Иванов Р. А.

Проверил:

Оглавление

[1Задание 3](#_Toc88229235)

[2 Выполнение лабораторной работы 3](#_Toc88229236)

# Задание

Создать JAVA-приложение, которое сможет рисовать фракталы. Обеспечить реализацию метода equals ().

# 2 Выполнение лабораторной работы

На рисунке ниже представлен результат выполнения программы



Ниже представлен измененный код файла FractalExplorer.

package fracktal;  
import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import javax.swing.filechooser.\*;  
import java.awt.image.\*;  
import java.io.File;  
  
public class FractalExplorer  
{  
 private JButton saveButton;  
 private JButton resetButton;  
 private JComboBox myComboBox;  
 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*);  
  
 resetButton = new JButton("Reset");  
  
 ButtonHandler resetHandler = new ButtonHandler();  
 resetButton.addActionListener(resetHandler);  
  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
  
 myFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 myComboBox = new JComboBox();  
  
 FractalGenerator mandelbrotFractal = new Mandelbrot();  
 myComboBox.addItem(mandelbrotFractal);  
 FractalGenerator tricornFractal = new Tricorn();  
 myComboBox.addItem(tricornFractal);  
 FractalGenerator burningShipFractal = new BurningShip();  
 myComboBox.addItem(burningShipFractal);  
  
 ButtonHandler fractalChooser = new ButtonHandler();  
 myComboBox.addActionListener(fractalChooser);  
  
 JPanel myPanel = new JPanel();  
 JLabel myLabel = new JLabel("Fractal:");  
 myPanel.add(myLabel);  
 myPanel.add(myComboBox);  
 myFrame.add(myPanel, BorderLayout.*NORTH*);  
  
 saveButton = new JButton("Save");  
 JPanel myBottomPanel = new JPanel();  
 myBottomPanel.add(saveButton);  
 myBottomPanel.add(resetButton);  
 myFrame.add(myBottomPanel, BorderLayout.*SOUTH*);  
  
 ButtonHandler saveHandler = new ButtonHandler();  
 saveButton.addActionListener(saveHandler);  
  
 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) {  
 myComboBox.setEnabled(val);  
 resetButton.setEnabled(val);  
 saveButton.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();  
 }  
  
 else if (command.equals("Save")) {  
  
 JFileChooser myFileChooser = new JFileChooser();  
  
 // Save only PNG images.  
 FileFilter extensionFilter =  
 new FileNameExtensionFilter("PNG Images", "png");  
 myFileChooser.setFileFilter(extensionFilter);  
  
  
 myFileChooser.setAcceptAllFileFilterUsed(false);  
  
  
 int userSelection = myFileChooser.showSaveDialog(display);  
  
  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
  
 java.io.File file = myFileChooser.getSelectedFile();  
 String file\_name = file.toString();  
  
 try {  
 BufferedImage displayImage = display.getImage();  
 javax.imageio.ImageIO.*write*(displayImage, "png", file);  
 }  
  
 catch (Exception exception) {  
 JOptionPane.*showMessageDialog*(display,  
 exception.getMessage(), "Cannot Save Image",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else return;  
 }  
 }  
 }  
  
  
 private class MouseHandler extends MouseAdapter  
 {  
  
 @Override  
 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);  
  
 // Redraw the fractal after the area being displayed has changed.  
 drawFractal();  
 }  
 }  
 private class FractalWorker extends SwingWorker<Object, Object>  
 {  
 */\*\*  
 \* A field for the integer y-coordinate of the row  
 \* that will be computed.  
 \*/* int yCoordinate;  
  
int[] computedRGBValues;  
  
private FractalWorker(int row) {  
 yCoordinate = row;  
 }  
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;  
  
 }  
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();  
 }  
  
}