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 №6-7

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 код файла JImageDisplay.java:

package fracktal;  
import javax.swing.JComponent;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
  
public class JImageDisplay extends JComponent {  
 private BufferedImage myImage;  
 public BufferedImage getImage(){  
 return myImage;  
 }  
 public JImageDisplay(int width , int height){  
 myImage = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
 Dimension myImageDimension = new Dimension(width, height);  
 setPreferredSize(myImageDimension);  
 }  
 @Override  
 public void paintComponent (Graphics g){  
 super.paintComponent(g);  
 g.drawImage (myImage, 0, 0, myImage.getWidth(), myImage.getHeight(), null);  
 }  
 public void clearImage(){  
 int[] allPixels = new int[getWidth() \* getHeight()];  
 myImage.setRGB(0, 0, getWidth(), getHeight(), allPixels, 0, 1);  
 }  
 public void drawPixel(int x, int y, int color){  
 myImage.setRGB(x, y, color);  
 }  
}

Ниже представлен код файла Mandelbrot.java:

package fracktal;  
import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator{  
 public static final int *MAX\_ITERATIONS* = 2500;  
 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 zR = 0;  
 double zIm = 0;  
 while (iteration < *MAX\_ITERATIONS* &&  
 zR \* zR + zIm \* zIm < 4)  
 {  
 double zrealUpdated = zR \* zR - zIm \* zIm + x;  
 double zimaginaryUpdated = 2 \* zR \* zIm + y;  
 zR = zrealUpdated;  
 zIm = zimaginaryUpdated;  
 iteration += 1;  
 }  
 if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
 }  
 public String toString() {  
 return "Mandelbrot";  
 }  
  
}

Ниже представлен код файла Tricorn.java:

package fracktal;  
  
import java.awt.geom.Rectangle2D;  
  
  
public class Tricorn extends FractalGenerator  
{  
  
 public static final int *MAX\_ITERATIONS* = 2000;  
  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
  
  
 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 "Tricorn";  
 }  
  
}

Ниже представлен код файла BurningShip.java:

package fracktal;  
import java.awt.geom.Rectangle2D;  
  
  
public class BurningShip extends FractalGenerator  
{  
 public static final int *MAX\_ITERATIONS* = 2000;  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2.5;  
 range.width = 4;  
 range.height = 4;  
 }  
 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 \* Math.*abs*(zreal) \* Math.*abs*(zimaginary) + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
 if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
 }  
 public String toString() {  
 return "Burning Ship";  
 }  
  
}

Ниже представлен код файла FractalExplorer.java:

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 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*);  
  
 JButton resetButton = new JButton("Reset");  
  
 ButtonHandler resetHandler = new ButtonHandler();  
 resetButton.addActionListener(resetHandler);  
  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
  
 myFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 JComboBox 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*);  
  
 JButton 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()  
 {  
 for (int x=0; x<displaySize; x++){  
 for (int y=0; y<displaySize; y++){  
 double xCoord = fractal.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
 double yCoord = fractal.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1){  
 display.drawPixel(x, y, 0);  
 }  
 else {  
 float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 display.drawPixel(x, y, rgbColor);  
 }  
  
 }  
 }  
  
 display.repaint();  
 }  
 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 chooser = new JFileChooser();  
 FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");  
 chooser.setFileFilter(filter);  
 chooser.setAcceptAllFileFilterUsed(false);  
  
 int userSelection = chooser.showSaveDialog(display);  
  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) {  
  
 java.io.File file = chooser.getSelectedFile();  
 String file\_name = file.toString();  
  
 try {  
 BufferedImage displayImage = display.getImage();  
 javax.imageio.ImageIO.*write*(displayImage, "png", new File(file\_name+".png"));  
 JOptionPane.*showMessageDialog*(display,"Save");  
 }  
  
 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)  
 {  
 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();  
 }  
 }  
 public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}