МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ

КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Знамени федеральное государственное

Бюджетное образовательное учреждение высшего образования

МОСКОВСКИЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ СВЯЗИ И

ИНФОРМАТИКИ

Кафедра математической кибернетики и информационных технологий

Лабораторная работа № 4 по теме:

«Рисование фракталов»

Выполнил: студент группы БИБ1902 \_\_\_\_\_\_\_Арустамян А.Б.

Проверил: \_\_\_\_\_\_\_\_Херсонский А. В.

Москва 2021

**Цель работы:**

В данной лабораторной работе нужно создать небольшое JAVA-приложение, которое сможет рисовать фракталы.

**Код программы:**

**FractalExplorer.java**

package com.company;  
  
import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
  
  
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 frame = new JFrame("Фракталы");  
  
  
 frame.add(display, BorderLayout.*CENTER*);  
  
  
 JButton resetButton = new JButton("Сбросить все");  
  
  
 ResetHandler handler = new ResetHandler();  
 resetButton.addActionListener(handler);  
  
  
 frame.add(resetButton, BorderLayout.*SOUTH*);  
  
  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
  
  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
  
  
  
 frame.pack();  
 frame.setVisible(true);  
 frame.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 ResetHandler implements ActionListener{  
  
  
 public void actionPerformed(ActionEvent e){  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 }  
  
  
 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();  
 }  
}

**FractalGenerator.java**

package com.company;  
  
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);  
}

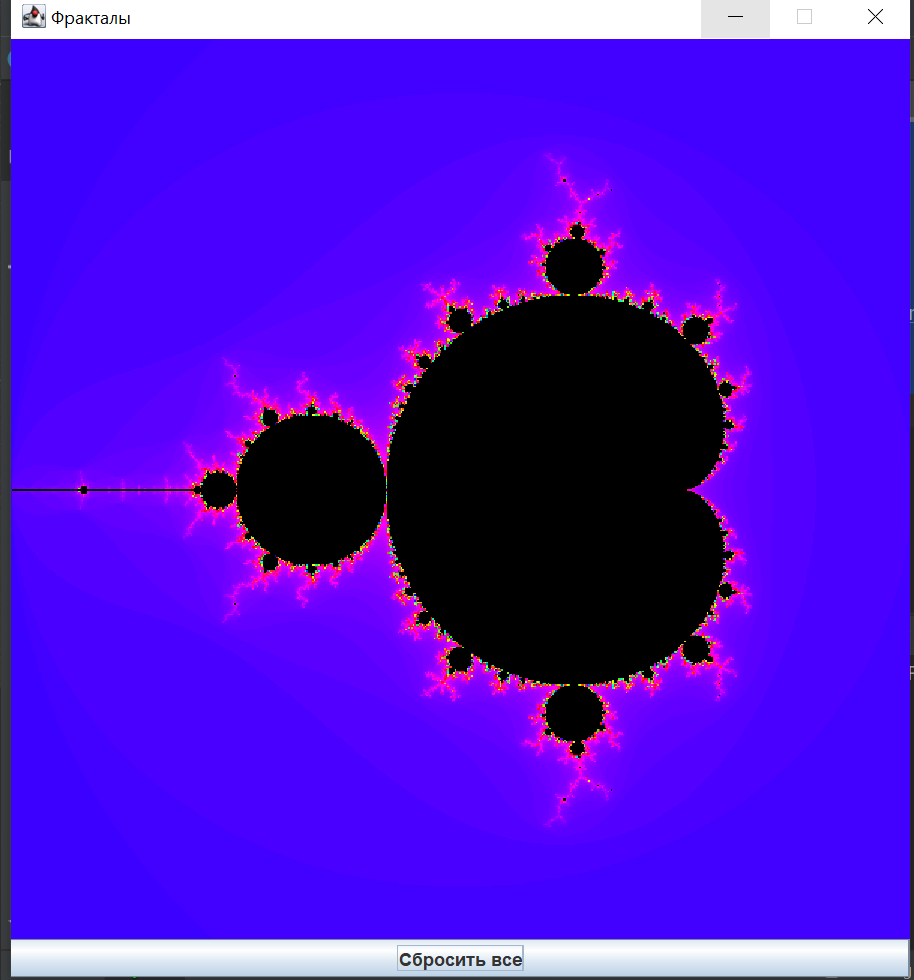
**JImageDisplay.java**

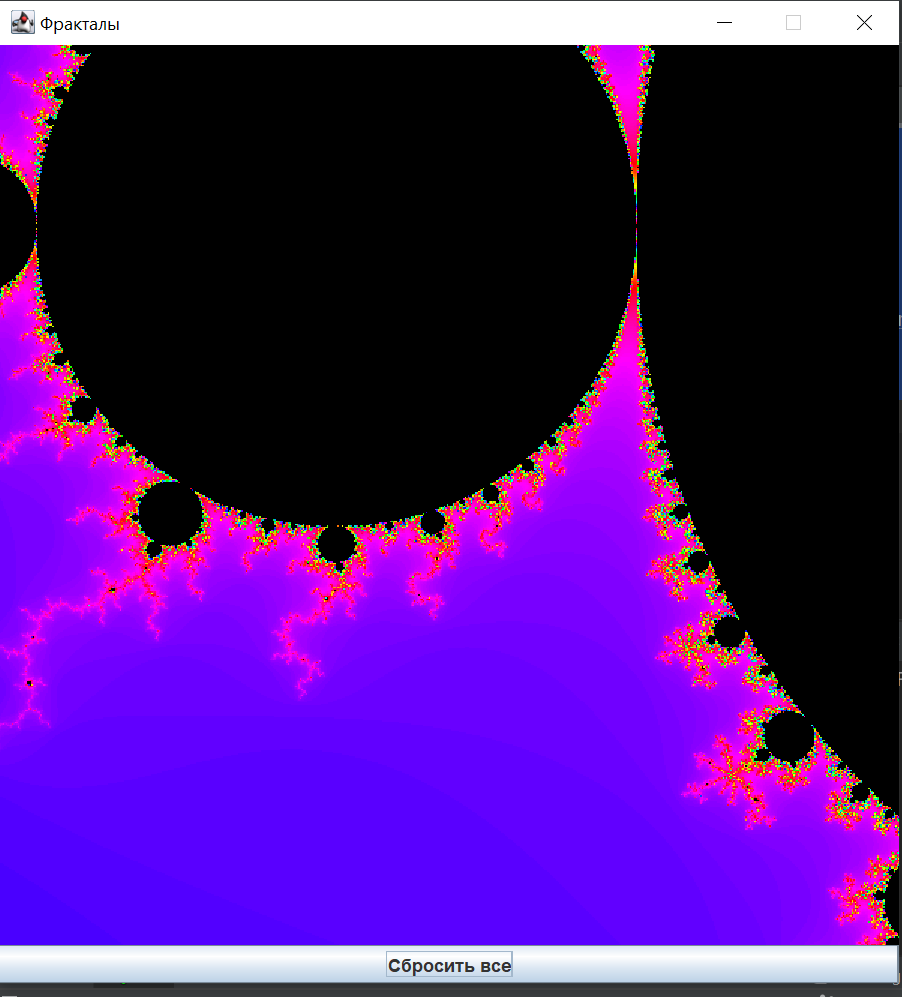
package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.\*;  
  
  
public class JImageDisplay extends JComponent {  
 private BufferedImage image;  
  
 JImageDisplay(int width, int height){  
 image= new BufferedImage( width, height, BufferedImage.*TYPE\_INT\_RGB*);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

**Mandelbrot.java**

package com.company;  
  
import java.awt.geom.\*;  
  
  
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;  
 }  
}

**Выполнение программы:**





**Вывод:**По итогу данной лабораторной работы я научился создавать простейший графический интерфейс и механизм для поддержки функции рисования фракталов, зависящий от классовой иерархии.