ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра Информатики

Лабораторная работа по технологиям программирования №4

«Фрактал Эксплорер»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Тагунов Виталий

Вариант №16

Москва, 2018

**Цель работы:** написать программу, рисующую фрактал Мандельброта.

**Анализ предметной области:** в этой работе я использовал пакет разработчика Java Development Kit. Для написания кода была использована программа Блокнот, для того, чтобы можно было без проблем использовать виртуальную среду разработки Java и его компилятор.

**Программа:**

**JImageDisplay.java:**

import javax.swing.JComponent;

import java.awt.image.BufferedImage;

import java.awt.Dimension;

import java.awt.Graphics;

public class JImageDisplay extends JComponent{

private BufferedImage image;

/\*\*

\* @param width - Picture width

\* @param height - Picture height

\*/

JImageDisplay(int width, int height){

image = new BufferedImage(width, height, BufferedImage.TYPE\_INT\_RGB);

super.setPreferredSize(new Dimension(width, height));

}

/\*\*

\* @param g - Displaying an image on the screen

\*/

protected void paintComponent(Graphics g)

{

super.paintComponent(g);

g.drawImage(image, 0, 0, im.getWidth(), im.getHeight(), null);

}

// Image cleanup

public void clearImage(){

int[] rgbArray = new int[image.getWidth()];

im.setRGB(0, 0, im.getWidth()-1, im.getHeight()-1, rgbArray, 0, 0);

}

// Setting the color to pixel

/\*\*

\* @param x - The abscissa of the point

\* @param y - The ordinate of the point

\* @param rgbColor - The color of the point

\*/

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

image.setRGB(x, y, rgbColor);

}

}

В классе **JImageDisplay** наследуется другой класс JComponent и используются функции: содержит функции отрисовки компонента, очистки изображения и закрашивания пикселя.

Отрисовки объекта **- paintComponent();**

Обновления изображения – **clearImage();**

Отрисовка пикселя – **drawPixel();**

**Mandelbrot.java:**

import java.awt.geom.Rectangle2D;

public class Mandelbrot extends FractalGenerator{

// Maximum iterations:

public static final int MAX\_ITERATIONS = 2000;

/\*\*

\* @param range - The range borders of the fractal

\*/

public void getInitialRange (Rectangle2D.Double range){

range.x=-2;

range.y=-1.5;

range.width=range.height=3;

}

//Use formulas for fractal Mandelbrot

/\*\*

\* @param x - The abscissa of the point

\* @param y - The ordinate of the point

\*/

public int numIterations(double x, double y){

double re=0;

double im=0;

for (int i=0; i<MAX\_ITERATIONS; i++){

double nextRe=re\*re-im\*im+x;

double nextIm=2\*re\*im+y;

**//Так как мы достигли MAX\_ITERATIONS возвращаем количество итераций**

if ((im\*im+re\*re) > 4)

return i;

**//Изменяем мнимую и действительные части**

re=nextRe;

im=nextIm;

}

return -1;

}

}

В этом классе происходит наследование класса FractalGenerator, а также рассчитывается количество итераций для вычисления фрактала Мандельброта.

**FractalExplorer.java:**

import javax.swing.\*;

import java.awt.\*;

import java.awt.event.\*;

import java.awt.geom.Rectangle2D;

public class FractalExplorer{

private JImageDisplay Jdisplay;

private int s;

private FractalGenerator Gen;

private Rectangle2D.Double range = new Rectangle2D.Double(0,0,0,0);

public static void main(String[] args)

{

FractalExplorer expl = new FractalExplorer(800);

expl.createAndShowGUI();

expl.drawFractal();

}

/\*\*

\* @param size - The height and width of the window

\*/

public FractalExplorer(int size){

s = size;

Jdisplay = new JImageDisplay(s, s);

Gen = new Mandelbrot();

Gen.getInitialRange(range);

}

public void createAndShowGUI(){

**// создаем окно**

JFrame frame = new JFrame("Fractal Explorer");

Jdisplay.addMouseListener(new mouse\_listener());**//добавляем фрактала**

frame.add(Jdisplay,BorderLayout.CENTER);

JButton button = new JButton("Reset");

button.addActionListener(new act\_listener());**//создаем кнопку**

frame.add(button, BorderLayout.SOUTH);

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);**//кнопка выхода**

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

public void drawFractal(){ **//функция отрисовки фрактала**

for(int i=0;i<s;i++)

for (int j=0;j<s;j++){

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, s, i);

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, s, j);

**// вычисление количества итераций**

int iterations = Gen.numIterations(xCoord, yCoord);

**//установка цвета, если он есть**

if(iterations!=-1){

float hue = 0.7f + (float) iterations / 200f;

int color = Color.HSBtoRGB(hue, 1f, 1f);

Jdisplay.drawPixel(i, j, color);

}

**//если нет**

else Jdisplay.drawPixel(i, j, 0);

}

Jdisplay.repaint();**//отрисовка картинки**

}

private class act\_listener implements ActionListener {

public void actionPerformed(ActionEvent e)

{

Gen.getInitialRange(range);

Jdisplay.clearImage();

drawFractal();

}

}

private class mouse\_listener extends MouseAdapter {

public void mouseClicked(MouseEvent e) {

super.mouseClicked(e);**//нахождение координат курсора**

int mouseX=e.getX();

int mouseY=e.getY();

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, dSize, mouseX);

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, dSize, mouseY);

Gen.recenterAndZoomRange(range, xCoord, yCoord, 0.5);

Jdisplay.clearImage();

drawFractal();

}

}

}

В этом классе происходит взаимодействие между остальными классами и собственно отрисовка самого фрактала.

**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.

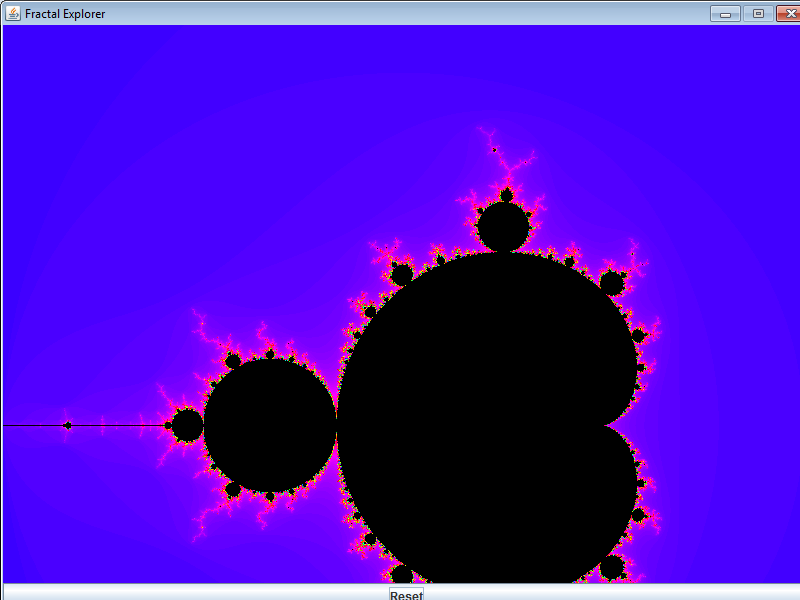
\*/

public abstract int numIterations(double x, double y);

}

В этом классе происходит генерация фрактала.

**Результат работы:**

****

**Вывод:** задача по графическому представлению фрактала Мандельброта выполнена.