ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа по технологиям программирования №4

«Фрактал Эксплорер»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Абросимов Данил

Вариант №1

Москва, 2018

Оглавление

[1. Цель и задачи. 3](#_Toc509173254)

[2. Анализ предметной области и выбор инструментария. 3](#_Toc509173255)

[3. Код программы и объяснение функций. 3](#_Toc509173256)

[3.1. Класс Mandelbrot. 3](#_Toc509173257)

[3.2. Класс JImageDisplay. 3](#_Toc509173258)

[3.3. Класс FractalExplorer. 3](#_Toc509173259)

[4. Заключение. 3](#_Toc509173260)

1. Цель и задачи.
2. Анализ предметной области и выбор инструментария.

В текущей работе я использовал бесплатно распространяемый пакет разработчика Java Development Kit. Однако, этот пакет не предусматривает интегрированную среду разработки, поэтому для написания непосредственно кода мною была выбрана программа Notepad++, так как она обеспечивает достаточно гибкий интерфейс и предоставляет возможность подсветки синтаксиса языка.

1. Код программы и объяснение функций.
   1. Класс Mandelbrot.

import java.awt.geom.Rectangle2D;

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 count = 0;

double real = 0;

double image = 0;

double z2 = 0;

while (count < MAX\_ITERATIONS && z2 < 4) {

count++;

double nextRe = Math.pow(real, 2) - Math.pow(image, 2) + x;

double nextIm = 2 \* real \* image + y;

z2 = Math.pow(nextRe, 2) + Math.pow(nextIm, 2);

real = nextRe;

image = nextIm;

}

return count < MAX\_ITERATIONS ? count : -1;

}

}

getInitialRange()

numIterations()

* 1. Класс JImageDisplay.

public class JImageDisplay extends JComponent {

private BufferedImage mImage;

public JImageDisplay(int width, int height) {

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

Dimension size = new Dimension(width, height);

super.setPreferredSize(size);

}

public BufferedImage getImage() {

return mImage;

}

public void paintComponent(Graphics g) {

super.paintComponent(g);

g.drawImage(mImage, 0, 0, mImage.getWidth(), mImage.getHeight(), null);

}

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

mImage.setRGB(x, y, rgbColor);

}

public void clearImage() {

for (int i = 0; i < mImage.getWidth(); i++) {

for (int j = 0; j < mImage.getHeight(); j++) {

drawPixel(i, j, 0);

}

}

}

}

Конструктор JImageDisplay() получает ширину и высоту и инициализирует ими переменную mImage.

PaintComponent() – для того, чтобы отрисовать изображение.

clearImage() – устанавливает все пиксели изображения на черный цвет.

drawPixel – установка пикселю на конкретной координате цвета

* 1. Класс FractalExplorer.

private class FractalWorker extends SwingWorker<Object, Object> {

private int mY;

private int[] RGBValue;

public FractalWorker(int y) {

mY = y;

}

public Object doInBackground() {

RGBValue = new int[display\_size];

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, display\_size, mY);

for (int x = 0; x < display\_size; x++) {

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, display\_size, x);

int n\_iters;

int colors = 0;

float temp;

n\_iters = generate.numIterations(xCoord, yCoord);

if (n\_iters >= 0) {

temp = 0.7f + n\_iters / 200f;

colors = Color.HSBtoRGB(temp, 1f, 1f);

}

RGBValue[x] = colors;

}

return null;

}

public void done() {

for (int x = 0; x < display\_size; x++) {

mImage.drawPixel(x, mY, RGBValue[x]);

}

mImage.repaint(0, 0, mY, display\_size, 1);

}

}

public void drawFractal() {

reset\_button.setEnabled(true);

for (int y = 0; y < display\_size; y++) {

FractalWorker worker = new FractalWorker(y);

worker.execute();

}

mImage.repaint();

}

private class actListener implements ActionListener {

public void actionPerformed(ActionEvent e) {

String cmd = e.getActionCommand();

if (cmd.equals("reset")) {

range = new Rectangle2D.Double();

generate.getInitialRange(range);

drawFractal();

}

}

}

private class mouseListener extends MouseAdapter {

public void mouseClicked(MouseEvent e) {

double xCoord = FractalGenerator.getCoord(range.x, range.x + range.width, display\_size, e.getX());

double yCoord = FractalGenerator.getCoord(range.y, range.y + range.height, display\_size, e.getY());

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

drawFractal();

}

}

public FractalExplorer(int size) {

display\_size = size;

generate = new Mandelbrot();

range = new Rectangle2D.Double();

generate.getInitialRange(range);

}

public void createAndShowGUI() {

JFrame frame = new JFrame("Лабораторная работа №3");

actListener handler = new actListener();

mImage = new JImageDisplay(display\_size, display\_size);

frame.getContentPane().add(mImage, BorderLayout.CENTER);

JPanel buttonsPanel = new JPanel();

reset\_button = new JButton("Обновить");

reset\_button.setActionCommand("reset");

reset\_button.addActionListener(handler);

buttonsPanel.add(reset\_button);

frame.getContentPane().add(buttonsPanel, BorderLayout.SOUTH);

frame.getContentPane().addMouseListener(new mouseListener());

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

}

createAndShowGUI() обеспечивает интерфейс пользователя: создает картинку кнопку сброса, которая обрабатывается командой.

FractalExplorer()

mouseListener()

actListener()

drawFractal()

doInBackground()

done()

1. Заключение.