**Федеральное агентство связи**

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №5

по дисциплине «Кроссплатформенные технологии программирования»

Выполнил: студент

группы БСТ1801

Скоморохов Виктор

Вариант 21

Москва 2020

**1. Задание на разработку программы**

В данной работе генератор фракталов из 4 лабораторной работы будет расширен двумя новыми функциями.

**2. Ход выполнения работы**

2.1)FractalExplorer.java

import javax.swing.\*;

import java.awt.\*;

import java.awt.geom.Rectangle2D;

import java.awt.event.\*;

import java.util.ArrayList;

import java.io.File;

import javax.swing.filechooser.FileNameExtensionFilter;

import javax.imageio.ImageIO;

import java.awt.image.BufferedImage;

import java.io.IOException;

public class FractalExplorer {

private int width;

private int height;

private JFrame frame;

private JPanel northP;

private JComboBox chooseF;

private JLabel textF;

private JImageDisplay display;

private Rectangle2D.Double range;

private JPanel southP;

private JButton resetB;

private JButton saveB;

private ArrayList<FractalGenerator> fractals;

private File nowPath = null;

//Класс события кнопки Reset

private class resetButtonListener implements ActionListener {

public void actionPerformed(ActionEvent event) {

int i = chooseF.getSelectedIndex();

fractals.get(i).getInitialRange(range);

FractalExplorer.this.drawFractal(i);

}

}

//Класс события кнопки Save

private class saveButtonListener implements ActionListener {

public void actionPerformed(ActionEvent event) {

// Создание диалогового окна для получения пути сохранения файла

JFileChooser fchooser;

if (nowPath == null) fchooser = new JFileChooser();

else fchooser = new JFileChooser(nowPath);

fchooser.setDialogTitle("Choose path");

// Настройка фильтров

fchooser.addChoosableFileFilter(new FileNameExtensionFilter("PNG Images", "\*.png"));

fchooser.setAcceptAllFileFilterUsed(false);

int result = fchooser.showSaveDialog(frame);

// Получение полного пути

String ext = "";

String extension = fchooser.getFileFilter().getDescription();

if (extension.equals("PNG Images")) ext = "png";

nowPath = new File(fchooser.getSelectedFile().getPath() + "." + ext);

// Запись файла на диск

try

{

ImageIO.*write*(display.getImage(), ext, nowPath);

JOptionPane.*showMessageDialog*(FractalExplorer.this.frame, "Save is success!", "File save", JOptionPane.*INFORMATION\_MESSAGE*);

} catch (IOException e) {

e.printStackTrace();

JOptionPane.*showMessageDialog*(FractalExplorer.this.frame, "Save is failed!", "File save", JOptionPane.*WARNING\_MESSAGE*);

}

}

}

//Класс события нажатия мыши

private class mouseClickListener implements MouseListener {

public void mouseClicked(MouseEvent e) {

int i = chooseF.getSelectedIndex();

if (i >= fractals.size()) return;

int x = e.getX();

int y = e.getY();

double xCoord = FractalGenerator.*getCoord*(range.x, range.x +

range.width, display.getWidth(), x);

double yCoord = FractalGenerator.*getCoord*(range.y, range.y +

range.height, display.getHeight(), y);

//ЛКМ

if (e.getButton() == MouseEvent.*BUTTON1*) {

fractals.get(i).recenterAndZoomRange(range, xCoord, yCoord, 0.5);

}

//ПКМ

if (e.getButton() == MouseEvent.*BUTTON3*) {

fractals.get(i).recenterAndZoomRange(range, xCoord, yCoord, 1.5);

}

FractalExplorer.this.drawFractal(i);

}

public void mouseEntered(MouseEvent e) {}

public void mouseExited(MouseEvent e) {}

public void mousePressed(MouseEvent e) {}

public void mouseReleased(MouseEvent e) {}

}

//Класс события нажатия на окно выбора

private class comboBoxClickListener implements ActionListener {

public void actionPerformed(ActionEvent event) {

int i = chooseF.getSelectedIndex();

fractals.get(i).getInitialRange(range);

FractalExplorer.this.drawFractal(i);

}

}

//Конструкторы

public FractalExplorer() {

this(800);

}

public FractalExplorer(int size) {

this(size, size);

}

public FractalExplorer(int width, int height) {

this.width = width;

this.height = height;

this.range = new Rectangle2D.Double();

fractals = new ArrayList<FractalGenerator>();

fractals.add(new Mandelbrot());

fractals.add(new Tricorn());

fractals.add(new BurningShip());

}

//Создание графического интерфейса

public void createAndShowGUI() {

//Создание формы

this.frame = new JFrame("Fraktalz");

this.frame.setDefaultCloseOperation(WindowConstants.*EXIT\_ON\_CLOSE*);

this.frame.setSize(this.width, this.height);

this.frame.setResizable(false);

//Создание панелей

northP = new JPanel();

southP = new JPanel();

//Добавление кнопки сброса

this.resetB = new JButton("Reset display");

this.resetB.setPreferredSize(new Dimension(frame.getWidth() / 3, 30));

southP.add(this.resetB);

//Добавление кнопки сохранения

this.saveB = new JButton("Save image");

this.saveB.setPreferredSize(new Dimension(frame.getWidth() / 3, 30));

southP.add(this.saveB);

//Добавление текста

this.textF = new JLabel("Fraktals: ");

Font font = saveB.getFont();

textF.setFont(font);

northP.add(this.textF);

//Создание и заполнение списка элементами

this.chooseF = new JComboBox();

for (int i = 0; i < fractals.size(); i++) {

chooseF.addItem(fractals.get(i).toString());

}

chooseF.addItem("-Empty-");

chooseF.setSelectedIndex(fractals.size());

// Задание размера и добавление на панель

this.chooseF.setPreferredSize(new Dimension(frame.getWidth() / 4, 30));

northP.add(this.chooseF);

// Добавление панелей на форму

frame.getContentPane().add(BorderLayout.*NORTH*, this.northP);

frame.getContentPane().add(BorderLayout.*SOUTH*, this.southP);

int height = frame.getHeight() - 60;

int width = height;

frame.setSize(width, frame.getHeight());

// Создание панели рисования

this.display = new JImageDisplay(width, height);

frame.getContentPane().add(BorderLayout.*CENTER*, this.display);

display.addMouseListener(new mouseClickListener());

resetB.addActionListener(new resetButtonListener());

saveB.addActionListener(new saveButtonListener());

chooseF.addActionListener(new comboBoxClickListener());

frame.setVisible(true);

}

//Отрисовка фрактала

public void drawFractal(int i) {

this.clearImage();

for (int x = 0; x < this.width; x++) {

for (int y = 0; y < this.height; y++) {

double xCoord = FractalGenerator.*getCoord*(range.x, range.x

+ range.width, display.getWidth(), x);

double yCoord = FractalGenerator.*getCoord*(range.y, range.y

+ range.height, display.getHeight(), y);

int numOfIter = fractals.get(i).numIterations(xCoord, yCoord);

int rgbColor;

if (numOfIter == -1) {

rgbColor = Color.*HSBtoRGB*(0, 0, 0);

}

else {

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

rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);

}

display.drawPixel(x, y, new Color(rgbColor));

}

}

}

//Управление панелью для рисования

public void clearImage() {

this.display.clearImage();

}

//Точка входа

public static void main(String[] args) {

FractalExplorer explorer = new FractalExplorer(800);

explorer.createAndShowGUI();

}

}

2.3)BurningShip.java

import java.awt.geom.Rectangle2D;

public class BurningShip extends FractalGenerator {

public static final int *MAX\_ITERATIONS* = 2000;

public BurningShip() {

}

*/\*\**

*\* Метод, позволяющий генератору фракталов определить наиболее «интересную»*

*\* область комплексной плоскости для конкретного фрактала*

*\*/*

@Override

public void getInitialRange(Rectangle2D.Double range) {

range.x = -2;

range.y = -2.5;

range.width = 4;

range.height = 4;

}

//Реализует итеративную функцию для фрактала BurningShip

@Override

public int numIterations(double x, double y) {

int i = 0;

double re = 0;

double im = 0;

while ((i < *MAX\_ITERATIONS*) && ((re \* re + im \* im) < 4)) {

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

double ip = Math.*abs*(2 \* re \* im + y);

re = Math.*abs*(rp);

im = ip;

i += 1;

}

if (i == *MAX\_ITERATIONS*)

return -1;

else

return i;

}

@Override

public String toString() {

return "BurningShip";

}

}

2.4)Tricorn.java

import java.awt.geom.Rectangle2D;

public class Tricorn extends FractalGenerator {

public static final int *MAX\_ITERATIONS* = 2000;

public Tricorn() {

}

*/\*\**

*\* Метод, позволяющий генератору фракталов определить наиболее «интересную»*

*\* область комплексной плоскости для конкретного фрактала*

*\*/*

@Override

public void getInitialRange(Rectangle2D.Double range) {

range.x = -2;

range.y = -2;

range.width = 4;

range.height = 4;

}

//Реализует итеративную функцию для фрактала Tricorn

@Override

public int numIterations(double x, double y) {

int i = 0;

double re = 0;

double im = 0;

while ((i < *MAX\_ITERATIONS*) && ((re \* re + im \* im) < 4)) {

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

double ip = -2 \* re \* im + y;

re = rp;

im = ip;

i += 1;

}

if (i == *MAX\_ITERATIONS*)

return -1;

else

return i;

}

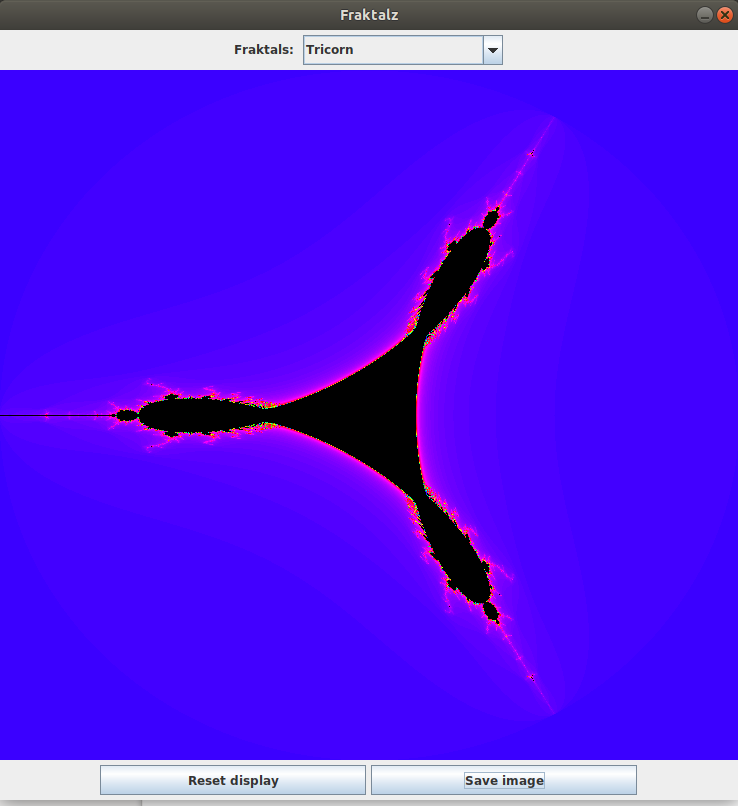
@Override

public String toString() {

return "Tricorn";

}

}

**3. Результат работы программы**

