**Федеральное агентство связи**

**Государственное бюджетное образовательное учреждение высшего**

**образование**

**Ордена Трудового Красного Знамени**

**«Московский технический университет связи и информатики»**

**Кафедра «МКиИТ»**

**дисциплина «Объектно-ориентированное программирование»**

Отчет по Лабораторной работе №5 :

Подготовила студентка

группы БВТ1901: Нкурикийе Х

Проверил: Мосева М.С.

Москва 2020

import javax.imageio.ImageIO;

import javax.swing.\*;

import javax.swing.filechooser.FileFilter;

import javax.swing.filechooser.FileNameExtensionFilter;

import java.awt.\*;

import java.awt.event.ActionEvent;

import java.awt.event.ActionListener;

import java.awt.event.MouseAdapter;

import java.awt.event.MouseEvent;

import java.awt.geom.Rectangle2D;

import java.io.File;

import java.io.IOException;

public class FractalExplorer {

private int size;

private JImageDisplay imageDisplay;

private FractalGenerator fractalGenerator;

private Rectangle2D.Double rectRange;

private JFrame frame;

public FractalExplorer(int size) {

this.size = size;

this.fractalGenerator = new Mandelbrot();

this.rectRange = new Rectangle2D.Double();

fractalGenerator.getInitialRange(rectRange);

}

public void createAndShowGUI() {

frame = new JFrame("Fractals generator");

frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);

JPanel panel = new JPanel();

JLabel label = new JLabel("Fractal:");

JComboBox<FractalGenerator> comboBox = new JComboBox<>();

comboBox.addItem(new Mandelbrot());

comboBox.addItem(new Tricorn());

comboBox.addItem(new BuringShip());

comboBox.addActionListener(new ComboBoxActionListener());

panel.add(label);

panel.add(comboBox);

imageDisplay = new JImageDisplay(size, size);

imageDisplay.addMouseListener(new MouseListener());

JPanel buttonPanel = new JPanel();

JButton clearButton = new JButton("Clear");

clearButton.addActionListener(new ButtonActionListener());

JButton saveButton = new JButton("Save");

saveButton.addActionListener(new ButtonActionListener());

buttonPanel.add(saveButton);

buttonPanel.add(clearButton);

frame.add(panel, BorderLayout.NORTH);

frame.add(imageDisplay, BorderLayout.CENTER);

frame.add(buttonPanel, BorderLayout.SOUTH);

frame.pack();

frame.setVisible(true);

frame.setResizable(false);

}

private void drawFractal() {

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

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

double x = FractalGenerator.getCoord(rectRange.x, rectRange.x + rectRange.width, size, i);

double y = FractalGenerator.getCoord(rectRange.y, rectRange.y + rectRange.width, size, j);

int numIters = fractalGenerator.numIterations(x, y);

// System.out.println(numIters);

if (numIters == -1) {

imageDisplay.drawPixel(i, j, Color.BLACK.getRGB());

} else {

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

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

imageDisplay.drawPixel(i, j, rgbColor);

}

}

}

imageDisplay.repaint();

}

public static void main(String[] args) {

FractalExplorer explorer = new FractalExplorer(800);

explorer.createAndShowGUI();

explorer.drawFractal();

}

class ButtonActionListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

if (e.getActionCommand().equals("Clear")) {

System.out.println(e.getActionCommand());

imageDisplay.clearImage();

imageDisplay.repaint();

drawFractal();

} else if (e.getActionCommand().equals("Save")) {

JFileChooser fileChooser = new JFileChooser();

FileFilter filter = new FileNameExtensionFilter("PNG Images", "png");

fileChooser.setFileFilter(filter);

fileChooser.setAcceptAllFileFilterUsed(false);

int result = fileChooser.showSaveDialog(frame);

if (result == JFileChooser.APPROVE\_OPTION) {

File file = fileChooser.getSelectedFile();

try {

ImageIO.write(imageDisplay.bufferedImage, "png", file);

} catch (IOException ex) {

JOptionPane.showMessageDialog(frame, ex.getMessage(), "Cannot save message", JOptionPane.ERROR\_MESSAGE);

}

}

}

}

}

class MouseListener extends MouseAdapter {

@Override

public void mouseClicked(MouseEvent e) {

super.mouseClicked(e);

double x = FractalGenerator.getCoord(rectRange.x, rectRange.x + rectRange.width, size, e.getX());

double y = FractalGenerator.getCoord(rectRange.y, rectRange.y + rectRange.width, size, e.getY());

fractalGenerator.recenterAndZoomRange(rectRange, x, y, 0.5);

drawFractal();

}

}

class ComboBoxActionListener implements ActionListener {

@Override

public void actionPerformed(ActionEvent e) {

fractalGenerator = (FractalGenerator) ((JComboBox<FractalGenerator>) e.getSource()).getSelectedItem();

fractalGenerator.getInitialRange(rectRange);

imageDisplay.clearImage();

imageDisplay.repaint();

drawFractal();

}

}

}





