**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

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

Кафедра Математической кибернетики и информационных технологий

**Отчет по лабораторной работе**

по предмету «Кроссплатформенные технологии программирования»

на тему:

«Многопоточный генератор фракталов»

Выполнил: студент группы

Митрохин Ярослав Игоревич

Руководитель:

Мосева Марина Сергеевна

Москва 2020

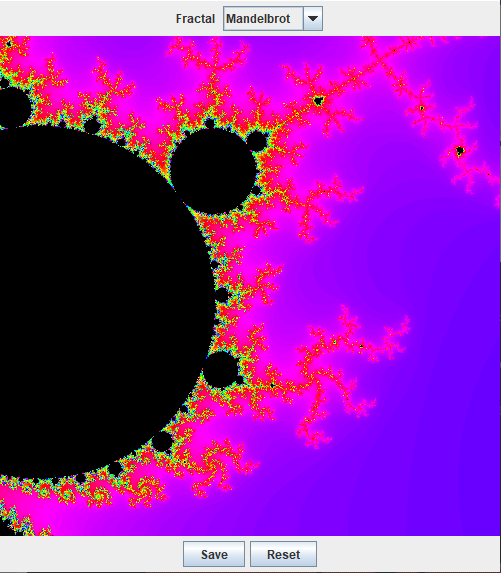
*Цель работы:* реализовать возможность рисования фрактала с несколькими фоновыми потоками. Добавление методов doInBackground(), done() и enableUI().

*Ход работы:*

*Реализация класса FractalExplorer:*

import java.awt.\*;  
import javax.swing.\*;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.BorderLayout;  
import java.awt.event.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.image.BufferedImage;  
import java.io.FileFilter;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JimageDisplay upDisplay;//Ссылка,для отображения в разных методах  
 private FractalGenerator fractalF;//Ссылка на базовый класс  
 private Rectangle2D.Double range;//Диапазон комплексной плоскости  
 private JComboBox comboBox;  
 private JButton saveBut;  
 private JButton resBut;  
 private int rowsR;  
 public FractalExplorer(int size)  
 {  
 displaySize = size;  
 range = new Rectangle2D.Double();  
 fractalF = new Mandelbrot();  
 fractalF.getInitialRange(range);//Инициализация объекты диапазона и фрактального генератора  
 upDisplay = new JimageDisplay(displaySize,displaySize);  
 }  
 public void createAndShowGUI()  
 {  
 JFrame frame = new JFrame("Fractal");  
 frame.add(upDisplay,BorderLayout.*CENTER*);  
 resBut = new JButton("Reset");  
 ResetDis reset = new ResetDis();  
 resBut.addActionListener(reset);  
 //frame.add(button,BorderLayout.SOUTH);  
  
 MousePiPi mouse = new MousePiPi();  
 upDisplay.addMouseListener(mouse);  
  
 comboBox = new JComboBox();  
 //добавление реализации генератора фрактала  
 FractalGenerator mandelbrotFrac = new Mandelbrot();  
 comboBox.addItem(mandelbrotFrac);  
 FractalGenerator tricornFrac = new Tricorn();  
 comboBox.addItem(tricornFrac);  
 FractalGenerator burningshipFrac = new BurningShip();  
 comboBox.addItem(burningshipFrac);  
  
 ResetDis fracCh = new ResetDis();  
 comboBox.addActionListener(fracCh);  
  
 JPanel panel = new JPanel();  
 JLabel label = new JLabel("Fractal ");  
 panel.add(label);  
 panel.add(comboBox);  
 frame.add(panel,BorderLayout.*NORTH*);//обавляем label в интерфейс перед списком  
  
 saveBut = new JButton("Save");  
 JPanel BotPan = new JPanel();  
 BotPan.add(saveBut);  
 BotPan.add(resBut);  
 frame.add(BotPan,BorderLayout.*SOUTH*);  
  
 ResetDis saveH = new ResetDis();  
 saveBut.addActionListener(saveH);  
  
 frame.pack();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 //Правильно разместят содержимое окна  
 }  
 private void drawFractal()  
 {  
 enableUI(false);  
 rowsR = displaySize;  
 for (int k = 0; k<displaySize; k++)  
 {  
 FractalWorker drawrow = new FractalWorker(k);  
 drawrow.execute();//запуск фоного потока и запуск задачу в фоновом режиме  
 }  
 /\*  
 for (int i = 0; i<displaySize;i++){  
 for (int j = 0; j<displaySize;j++){  
 double xCoord = FractalGenerator.getCoord(range.x, range.x+range.width,displaySize, i);//Пиксельная и координата в пространстве  
 double yCoord = FractalGenerator.getCoord(range.y, range.y+range.height,displaySize, j);  
 int iter = fractalF.numIterations(xCoord,yCoord);  
 if (iter == -1)  
 {  
 upDisplay.drawPixel(i,j,0);  
 }  
 else  
 {  
 float hue = 0.7f + (float) iter/200f;  
 int rgbColor = Color.HSBtoRGB(hue,1f,1f);  
 upDisplay.drawPixel(i,j,rgbColor);  
 }  
 }  
 }  
 upDisplay.repaint();//Обновление JimageDisplay в соответствии с цветом для каждого пикселя  
 \*/  
 }  
  
 private void enableUI(boolean res)//вкл\выкл кнопок и тд  
 {  
 comboBox.setEnabled(res);  
 resBut.setEnabled(res);  
 saveBut.setEnabled(res);  
 }  
 private class ResetDis implements ActionListener  
 {  
 public void actionPerformed(ActionEvent e)  
 {  
 if (e.getSource() instanceof JComboBox)  
 {//извлечение выбранного элемента из виджета и установить его в качестве текущего генератора  
 JComboBox cB = (JComboBox) e.getSource();  
 fractalF = (FractalGenerator) cB.getSelectedItem();  
 fractalF.getInitialRange(range);  
 drawFractal();  
 } else if (e.getActionCommand().equals("Reset"))//Выбираем элемент из виджета  
 {  
 fractalF.getInitialRange(range);  
 drawFractal();  
 } else if (e.getActionCommand().equals("Save"))  
 {  
 //В какой файл сохранять  
 JFileChooser fileChooser = new JFileChooser();  
 //Выбор файла, только пнг  
 FileNameExtensionFilter eFilter = new FileNameExtensionFilter("PNG IMAGES","png");  
 fileChooser.setFileFilter(eFilter);  
   
 fileChooser.setAcceptAllFileFilterUsed(false);//Только пнг  
 //Открытие диалогового окна  
 int sel = fileChooser.showSaveDialog(upDisplay);  
  
 if(sel == JFileChooser.*APPROVE\_OPTION*)//нет-отменил сохранение  
 {  
 java.io.File file = fileChooser.getSelectedFile();  
 String filename = file.toString();  
 try {  
 BufferedImage bufferedImage = upDisplay.bufferedImage;  
 javax.imageio.ImageIO.*write*(bufferedImage,"png",file);//загрузка и сохранение изображения  
 }  
 catch(Exception exception) {//обработка ошибки  
 JOptionPane.*showMessageDialog*(upDisplay,exception.getMessage(),"Can't save file",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else return;  
 }  
 }  
 }  
 private class MousePiPi extends MouseAdapter  
 {  
 public void mouseClicked(MouseEvent e)  
 {  
 if (rowsR!=0) {return;}//реагирует на щелк только, когда больше нет строк  
 int x = e.getX();  
 double xCoord = fractalF.*getCoord*(range.x, range.x+range.width,displaySize, x);  
 int y = e.getY();  
 double yCoord = fractalF.*getCoord*(range.y, range.y+range.height,displaySize, y);  
 fractalF.recenterAndZoomRange(range, xCoord,yCoord,0.5);  
 drawFractal();  
 }  
 }  
  
  
 private class FractalWorker extends SwingWorker<Object,Object>//вычисление значений цвета для одной строки  
 {  
 int yCoord;//строки  
 int[] RGBVal;//хранение вычисленных значений ргб  
  
 private FractalWorker(int row)  
 {  
 yCoord = row;  
 }  
  
 protected Object doInBackground()//выполнение фоновых операций  
 {  
 RGBVal = new int[displaySize];  
  
 for (int i = 0; i < RGBVal.length;i++)//Сохранение каждое значение ргб в соответствующем элементе  
 {  
 double xCord = fractalF.*getCoord*(range.x,range.x+range.width,displaySize, i);  
 double yCord = fractalF.*getCoord*(range.y,range.y+range.height,displaySize, yCoord);  
  
 int iter2 = fractalF.numIterations(xCord,yCord);  
  
 if (iter2 == -1)  
 {  
 RGBVal[i] = 0;  
 }  
 else  
 {  
 float tref = 0.7f + (float) iter2/ 200f;  
 int rgbCol = Color.*HSBtoRGB*(tref,1f,1f);  
 RGBVal[i] = rgbCol;  
 }  
 }  
 return null;  
 }  
  
 protected void done()//фоновая задача завершена  
 {  
 for (int i = 0; i < RGBVal.length; i++)  
 {  
 upDisplay.drawPixel(i,yCoord,RGBVal[i]);  
 }  
 upDisplay.repaint(0,0,yCoord,displaySize,1);//перерисовываем часть изображения  
  
 rowsR--;  
 if (rowsR==0)  
 {  
 enableUI(true);  
 }}}  
 public static void main(String[] args)  
 {  
 FractalExplorer display1 = new FractalExplorer(500);  
 display1.createAndShowGUI();  
 display1.drawFractal();  
 }  
}

*Результаты:*

**

Рисование фракталов происходит гораздо быстрее. При процессе рисования фрактала кнопки сохранения, сброса и выбора фрактала становятся недействительными. А также рисование фрактала происходит построчно, сохраняя каждое значение PNG в соответствующем значении массива.

*Выводы:* реализовать возможность рисования фрактала с несколькими фоновыми потоками. Добавление методов doInBackground(), done() и enableUI().