ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени федеральное государственное

Бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра информатики

Лабораторная работа №6

По дисциплине: «Технология разработки программного обеспечения»

Тема: «Многопотоный фракталов»

Выполнил студент:

Группы: БФИ1902

Шацкий Е. И.

Проверила:

Мосева. М.С.

Москва, 2020 г.

**Содержание**

[1 Цель работы 3](#_Toc59458312)

[2 Задачи на лабораторную работу. 3](#_Toc59458313)

[3 Ход решения лабораторной работы. 3](#_Toc59458314)

[Листинг класса FractalExplorer 8](#_Toc59458315)

[Листинг класса FractalWorker 13](#_Toc59458316)

[Вывод: 14](#_Toc59458317)

[Список используемых источников 15](#_Toc59458318)

# **1 Цель работы**

В данной лабораторной работе необходимо реализовать возможность рисования фрактала с несколькими фоновыми потоками.

# **2 Задачи на лабораторную работу.**

Для выполнения лабораторной работы необходимо:

1) Необходимо изменить программу так, чтобы она использовала один или несколько фоновых потоков для вычисления фракталов

2) Необходимо соблюдать важное ограничение Swing, а именно, что в  
потоке обработке событий происходит взаимодействие только с компонентами Swing.

3) Создать подкласс SwingWorker с именем FractalWorker, который  
будет внутренним классом FractalExplorer.

4) Создайте функцию void enableUI(boolean val), которая будетвключать или отключать кнопки с выпадающим списком в пользовательскоминтерфейсе на основе указанного параметра.

5) Создайте функцию void enableUI(boolean val), которая будет  
включать или отключать кнопки с выпадающим списком в пользовательском  
интерфейсе на основе указанного параметра.

6) Наконец, измените реализацию mouse-listener для того, чтобы онасразу возвращалась в предыдущее состояние, если значение «rows remaining»не равно нулю.

# **3 Ход решения лабораторной работы.**

В начале стоит отметить то, что большая часть классов описана в предыдущей лабораторной работе.

В данной лабораторной работе в основном я буду работать в  
классе FractalExplorer. Часть кода будет новой, но некоторые части будут  
представлять из себя модифицированный код, который я уже написал ранее.  
 Для начала создам подкласс SwingWorker с именем FractalWorker, который будет внутренним классом FractalExplorer. Это необходимо для того, чтобы у него был доступ к нескольким внутренним членам FractalExplorer. Необходимо помнить, что класс SwingWorker является универсальным, поэтому нужно указать параметры - можно просто указать Object для двух параметров, потому что в данной реализации эти параметры не будут использоваться. В результате у получится следующая строчка кода:  
private class FractalWorker extends SwingWorker<Object, Object>  
 2) Класс FractalWorker будет отвечать за вычисление значений цвета для  
одной строки фрактала, поэтому ему потребуются два поля: целочисленная y координата вычисляемой строки, и массив чисел типа int для хранения  
вычисленных значений RGB для каждого пикселя в этой строке. Конструктор  
должен будет получать y-координату в качестве параметра и сохранять это. (На данном этапе не надо выделять память под целочисленный массив, так как он не потребуется, пока строка не будет вычислена.)  
 3) Метод doInBackground() вызывается в фоновом потоке и отвечает за  
выполнение длительной задачи. Поэтому в моей реализации нужно  
взять часть кода из вашей предыдущей функции «draw fractal» и поместить ее в этот метод. Вместо того, чтобы рисовать изображение в окне, цикл должен  
будет сохранить каждое значение RGB в соответствующем элементе  
целочисленного массива. Чтобы не нарушить ограничения потоков Swing можно изменять отображение из этого потока.  
 4) Вместо этого можно выделить память для массив целых чисел в начале реализации этого метода (массив должен быть достаточно большим для  
хранения целой строки значений цвета), а затем сохраняю цвет каждого  
пикселя в этом массиве. Единственные различия между настоящим и  
предыдущим кодом в том, что вам нужно будет вычислить фрактал для  
указанной строки, и что вы на данном этапе не обновляете отображение.  
Метод doInBackground() будет возвращать объект типа Object, так как  
это указано в объявлении SwingWorker <T, V>. Просто верну null.  
 5) Метод done() вызывается, когда фоновая задача завершена, и этот  
метод вызывается из потока обработки событий Swing. Это означает, что   
можно модифицировать компоненты Swing на ваш вкус. Поэтому в этом  
методе можно перебирать массив строк данных, рисуя пиксели, которые  
были вычислены в doInBackground ().  
После того, как строка будет вычислена, необходимо будет сообщить  
Swing, перерисовать часть изображения, которая была изменена. Поскольку я изменил только одну строку, перерисовывать изображение целиком будет  
затратно, поэтому можно использовать метод JComponent.repaint(), который  
позволит указать область для перерисовки. У данного метода есть  
неиспользуемый параметр типа long, можно просто указать 0 для этого  
аргумента. В качестве остальных параметров укажу вычисленную строку,  
значения начала фрагмента для перерисовки (0, y) и конечные значения  
фрагмента (displaySize, 1).  
 После того, как я завершил класс для фоновой задачи, следующим  
шагом нужно будет привязать его к процессу рисования фракталов. Так как  
часть кода из функции «draw fractal» уже задействована в разрабатываемом  
классе, на данном этапе можно изменить функцию «draw fractal», а именно, для каждой строки в отображении создается отдельный рабочий объект, а затем вызвается для него метод execute (). Это действие запустит фоновый поток и запустит задачу в фоновом режиме. Необходимо запомнить, что класс FractalWorker отвечает за генерацию данных строки и за рисование этой строки, поэтому функция «draw fractal» должна быть простой. После завершения данной функции, я заменяю отображение, чтобы оно  
стало более быстрым, а пользовательский интерфейс стал более отзывчивым.  
Также заменю одну проблему в пользовательском интерфейсе – при нажатии на экран или на кнопку во время перерисовки, программа обработает это событие, хотя оно должно быть проигнорировано до завершения операции.

Самый простой способ решить проблему игнорирования событий во  
время перерисовки - отслеживать количество оставшихся строк, которые  
должны быть завершены, и игнорировать или отключать взаимодействие с  
пользователем до тех пор, пока не будут нарисованы все строки. Для этого нужно добавить поле «rows remaining» в класс Fractal Explorer и использовать его, чтобы узнать, когда будет завершена перерисовка. Чтение и запись этого  
значения будет происходить в потоке обработки событий, чтобы не было  
параллельного доступа к этому элементу. Если взаимодействие с ресурсом  
будет происходить только из одного потока, то не возникнет ошибок  
параллелизма. Для этого:  
 Создам функцию void enableUI(boolean val), которая будет  
включать или отключать кнопки с выпадающим списком в пользовательском  
интерфейсе на основе указанного параметра. Для включения или отключения  
этих компонентов буду использовать метод Swing setEnabled(boolean).  
Необходимо убедиться , что метод обновляет состояние кнопки сохранения, кнопки сброса и выпадающего списка.  
 Функция «draw fractal» делает две вещи. Первая - она  
вызвает метод enableUI (false), чтобы отключить все элементы  
пользовательского интерфейса во время рисования. Вторая -   
устанавливает значение «rows remaining» равным общему количеству строк,  
которые нужно нарисовать. Эти действия сделаны перед  
выполнением каких-либо рабочих задач, иначе это может привести к  
некорректной работе алгоритма.  
 В методе done(), уменьшаю значение «rows remaining» на 1, как  
последний шаг данной операции. Затем, если после уменьшения значение «rows remaining» равно 0, вызовается метод enableUI (true).  
 Наконец, изменю реализацию mouse-listener для того, чтобы она  
сразу возвращалась в предыдущее состояние, если значение «rows remaining»  
не равно нулю. Другими словами, приложение будет реагировать на щелчки  
мышью, только в том случае, если больше нет строк, которые должны быть  
нарисованы.

На рисунке 1 представлен результат работы программы.

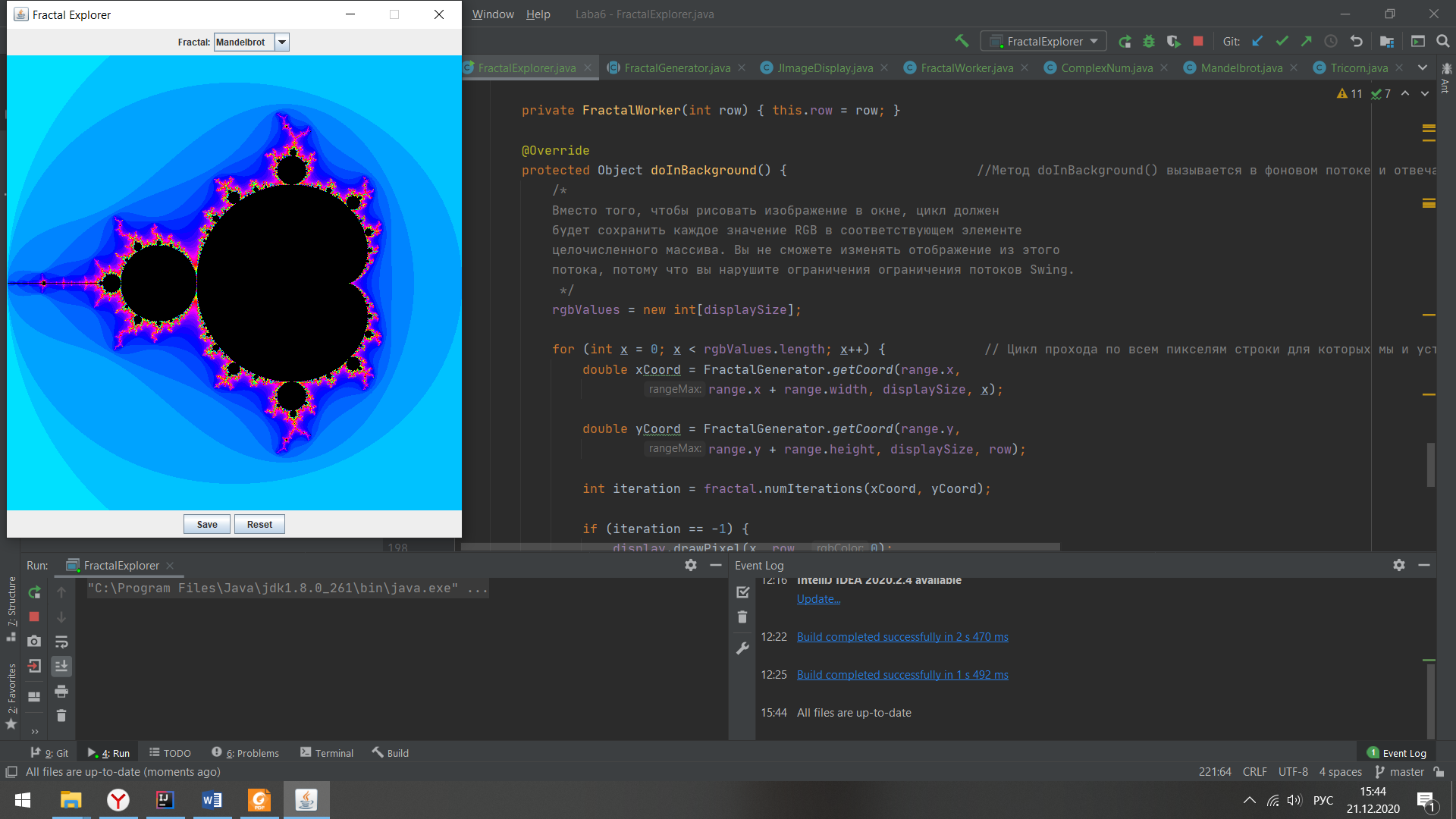


Рисунок 1 – Результат работы программы

На рисунке 2 – Показан результат приближения фрактала.

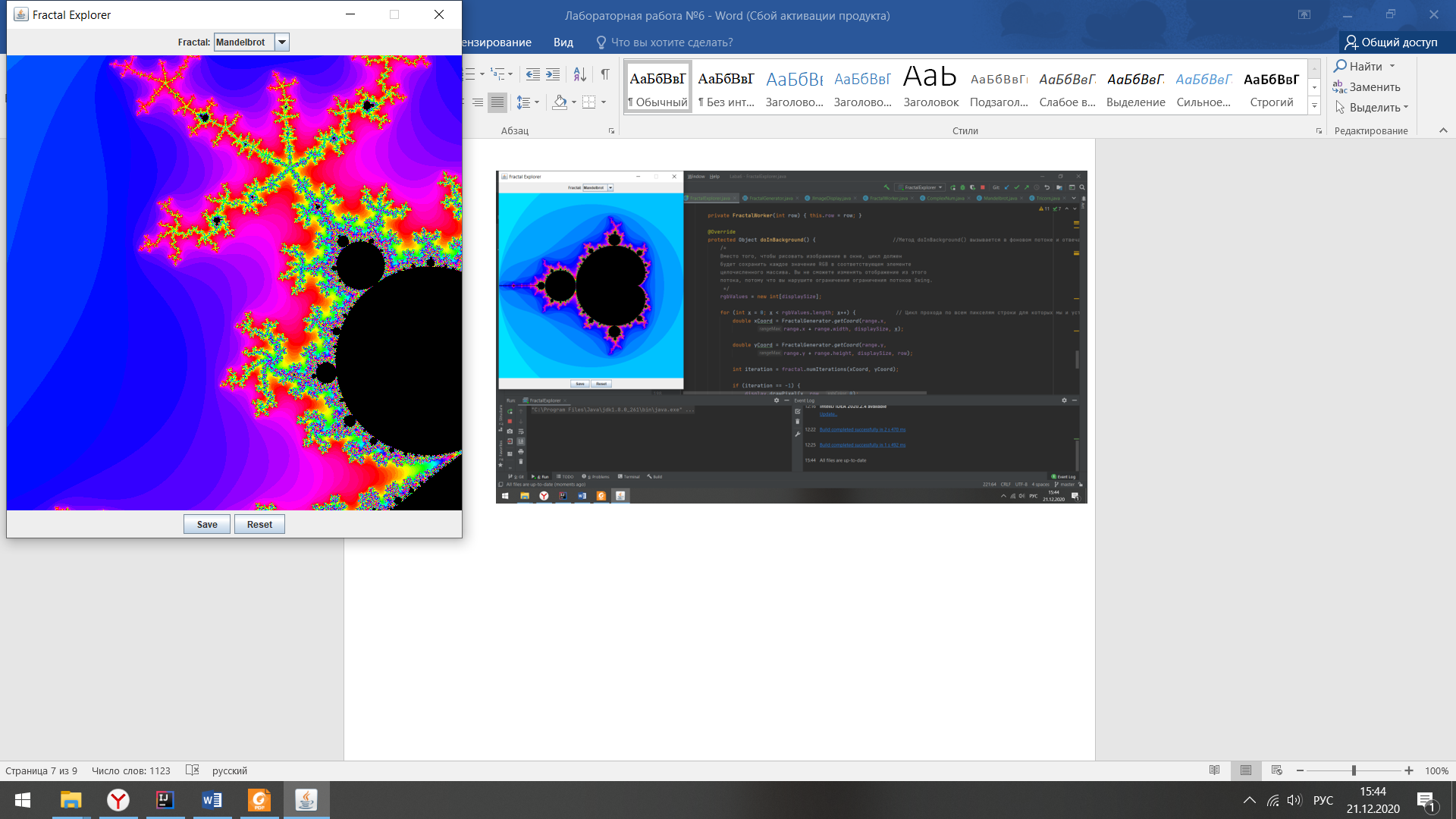


Рисунок 2 – результат приближения фрактала

На рисунке 3 продемонстрирован результат сохранения картинки фрактала в формате png

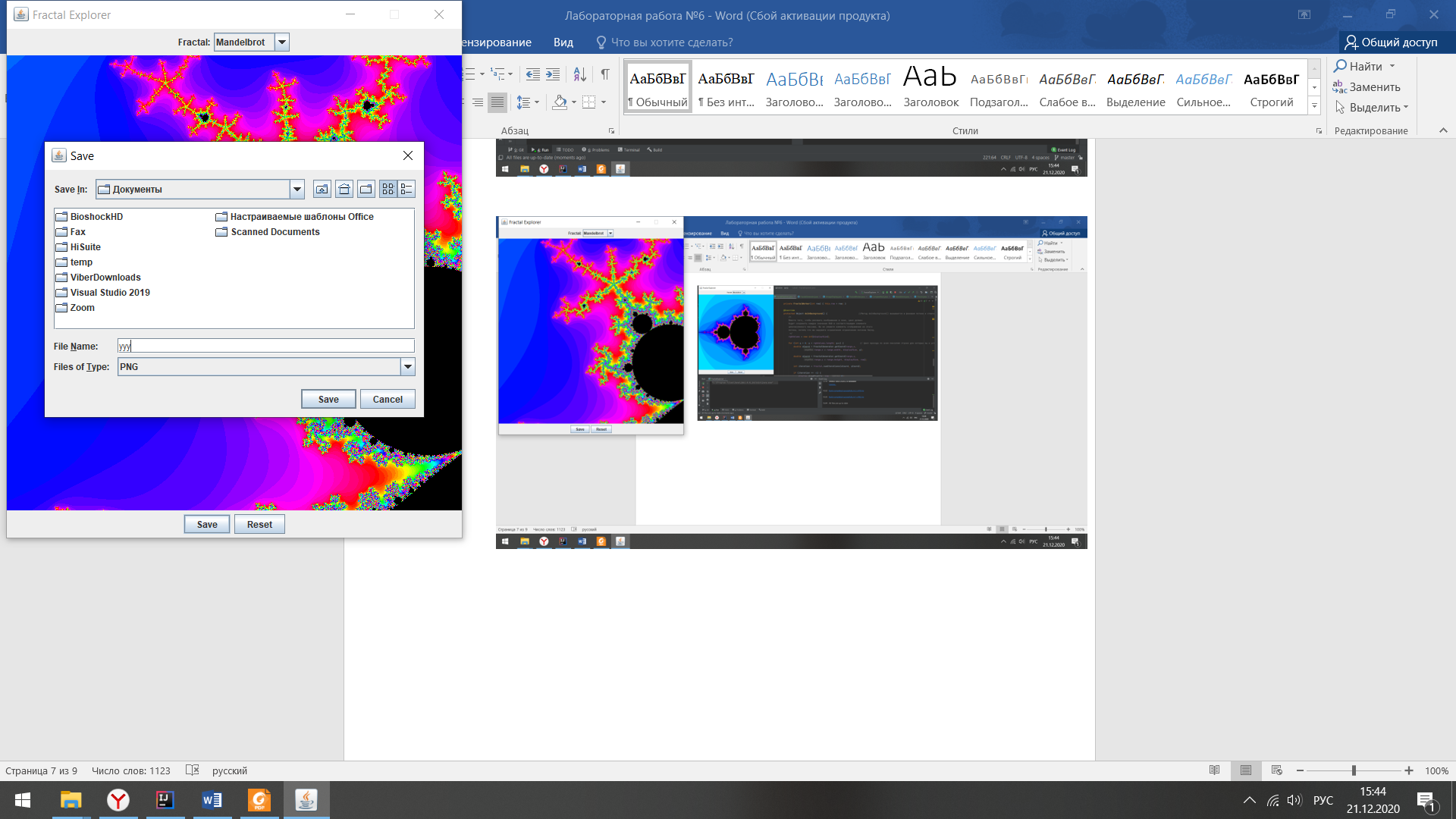


Рисунок 3 – Сохранение картинки фрактала

## **Листинг класса FractalExplorer**

import java.awt.\*;  
import javax.swing.\*;  
import javax.swing.filechooser.FileFilter;  
import javax.swing.filechooser.FileNameExtensionFilter;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
import java.awt.image.BufferedImage;  
import java.io.File;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JImageDisplay display;  
 private FractalGenerator fractal;  
 private Rectangle2D.Double range;  
 private int rowRemaining;  
  
 private JButton saveButton;  
 private JButton resetButton;  
 private JComboBox comboBox;  
  
 public FractalExplorer(int size) {  
 displaySize = size; // Установка размера экрана  
  
 fractal = new Mandelbrot();  
 range = new Rectangle2D.Double(); // Создание комплексной плоскости  
  
 fractal.getInitialRange(range); // Считывание диапозона  
 display = new JImageDisplay(displaySize, displaySize); // Создание изображения  
 }  
  
 public void createAndShowGUI() {  
 display.setLayout(new BorderLayout());  
  
 resetButton = new JButton("Reset"); // Добавляем кнопоку Reset  
 Resetter resetHandler = new Resetter();  
 resetButton.addActionListener(resetHandler);  
  
 saveButton = new JButton("Save"); // Добавляем кнопку Save  
 Saver saveHandler = new Saver();  
 saveButton.addActionListener(saveHandler);  
  
 Clicker click = new Clicker();  
 display.addMouseListener(click);  
  
 FractalGenerator mandelbrotFractal = new Mandelbrot(); // Cоздание обьектов классов  
 FractalGenerator tricornFractal = new Tricorn();  
 FractalGenerator burningShipFractal = new BurningShip();  
  
 comboBox = new JComboBox(); // Создание комбобокса  
  
 comboBox.addItem(mandelbrotFractal); // Добавение элементов комбобокса  
 comboBox.addItem(tricornFractal);  
 comboBox.addItem(burningShipFractal);  
  
 Chooser fractalChooser = new Chooser();  
 comboBox.addActionListener(fractalChooser);  
  
 JLabel label = new JLabel("Fractal:"); // Текст  
  
 JPanel panel = new JPanel(); //Добавление Label на панель  
 panel.add(label);  
 panel.add(comboBox);  
  
 JPanel myBottomPanel = new JPanel(); // Нижняя панель  
 myBottomPanel.add(saveButton);  
 myBottomPanel.add(resetButton);  
  
 JFrame myFrame = new JFrame("Fractal Explorer"); // Название окна  
  
 myFrame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 myFrame.add(myBottomPanel, BorderLayout.*SOUTH*); // Задаем расположение  
 myFrame.add(display, BorderLayout.*CENTER*);  
 myFrame.add(panel, BorderLayout.*NORTH*);  
  
 myFrame.pack();  
 myFrame.setVisible(true);  
 myFrame.setResizable(false);  
 }  
  
 private void drawFractal() { // Отрисовка фрактала на фоне  
 enableUI(false); // Отключение всех элементов пользовательского интерфейса во время рисования фрактала  
  
 rowRemaining = displaySize; // Установить значения «rows remaining» равным общему количеству строк, которые нужно нарисовать.  
  
 for (int y = 0; y < displaySize; y++){  
 FractalWorker drawRow = new FractalWorker(y); // Создание рабочего обьекта дял каждой строки  
 drawRow.execute(); // Это действие запустит фоновый поток и запустит задачу в фоновом режиме.  
 }  
 }  
 private void enableUI(boolean value) {  
 comboBox.setEnabled(value); // Включение или отключение кнопки с выпадающим списком в пользовательском интерфейсе на основе указанного параметра  
 resetButton.setEnabled(value);  
 saveButton.setEnabled(value);  
 }  
  
 private class Resetter implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 if (e.getActionCommand().equals("Reset")) {  
 fractal.getInitialRange(range); // Сброс диапазона и перерисовывка фрактала  
 drawFractal();  
 }  
 }  
 }  
  
 private class Chooser implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 Object source = e.getSource(); // Метод getsource возвращает источник события, это необходимо для того, чтобы понять, что произошло в комбобоксе  
 if (source instanceof JComboBox) {  
 JComboBox comboBox = (JComboBox) source; // Преобразование для того что бы дальше работать как с обьектом а не как с ссылкой  
  
 fractal = (FractalGenerator) comboBox.getSelectedItem(); // Передаем в фрактал генератор тот элемент который мы выбрали  
 assert fractal != null; // Убеждаемся что он что то содержит  
  
 fractal.getInitialRange(range); // Узнаем размер  
 drawFractal(); // Рисуем фрактал  
 }  
 }  
 }  
  
 private class Saver implements ActionListener {  
 public void actionPerformed(ActionEvent e) {  
 if (e.getActionCommand().equals("Save")) { //Save - команда действия эту команду мы считываем с помощью getActionCommand  
 JFileChooser fileChooser = new JFileChooser();  
  
 FileFilter extensionFilter = new FileNameExtensionFilter( //для того что бы отображались файлы только с указанными типами, в методический указаниях есть указание использовать тип png  
 "PNG",  
 "png"  
 );  
  
 fileChooser.setFileFilter(extensionFilter);  
  
 fileChooser.setAcceptAllFileFilterUsed(false);  
  
 int userSelection = fileChooser.showSaveDialog(display); //открывает окно Save File  
  
 if (userSelection == JFileChooser.*APPROVE\_OPTION*) { //если пользователь выбрал файл записываем его в строку  
 java.io.File file = fileChooser.getSelectedFile();  
 String filePath = file.getPath();  
  
 if (!filePath.contains(".png"))  
 file = new File(filePath + ".png"); //не указал расширение файла, добавляется автоматически  
 try { //try – определяет блок кода, в котором может произойти исключение;  
 BufferedImage displayImage = display.getImage();  
 javax.imageio.ImageIO.*write*(displayImage, "png", file);  
 } catch (Exception exception) { //catch – определяет блок кода, в котором происходит обработка исключения;  
 JOptionPane.*showMessageDialog*(display,  
 exception.getMessage(), "Cannot Save Image",  
 JOptionPane.*ERROR\_MESSAGE*);  
 }  
 } else return;  
 }  
 }  
 }  
  
 private class Clicker extends MouseAdapter {  
 @Override  
 public void mouseClicked(MouseEvent e) //Увеличение фрактала по щелчку мыши  
 {  
 int x = e.getX();  
 double xCoord = FractalGenerator.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
  
 int y = e.getY();  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
  
 drawFractal();  
 }  
 }  
 private class FractalWorker extends SwingWorker<Object, Object> {  
 int row;  
 int[] rgbValues;  
  
 private FractalWorker(int row) { this.row = row; }  
  
 @Override  
 protected Object doInBackground() { //Метод doInBackground() вызывается в фоновом потоке и отвечает за выполнение длительной задачи.  
 /\*  
 Вместо того, чтобы рисовать изображение в окне, цикл должен  
 будет сохранить каждое значение RGB в соответствующем элементе  
 целочисленного массива. Вы не сможете изменять отображение из этого  
 потока, потому что вы нарушите ограничения ограничения потоков Swing.  
 \*/  
 rgbValues = new int[displaySize];  
  
 for (int x = 0; x < rgbValues.length; x++) { // Цикл прохода по всем пикселям строки для которых мы и устанавливаем цвет  
 double xCoord = FractalGenerator.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
 range.y + range.height, displaySize, row);  
  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1) {  
 display.drawPixel(x, row, 0);  
 } else {  
 float hue = 0.5f + (float) iteration / 50;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
 rgbValues[x] = rgbColor;  
 }  
 }  
 return null;  
 }  
  
 protected void done() {  
 for (int x = 0; x < rgbValues.length; x++) {  
 display.drawPixel(x, row, rgbValues[x]); // Вычисление строки  
 }  
  
 display.repaint(0, 0, row, displaySize, 1); // Указываем область для перерисовки.  
  
 rowRemaining--;  
 if (rowRemaining == 0) enableUI(true);  
 }  
 }  
 public static void main(String[] args) {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

## Листинг класса FractalWorker

import javax.swing.\*;  
  
public class FractalWorker extends SwingWorker { //принимает object, object  
 @Override //Переопрделение  
 protected Object doInBackground() throws Exception { //Метод doInBackground() должен возвращать объект типа Object, так как это указано в объявлении SwingWorker <T, V>. Просто верните null.  
 return null;  
 }  
}

# Вывод:

В данной лабораторной работе я реализовал возможность рисования фрактала с несколькими фоновыми потоками.

# **Список используемых источников**

1.ГОСТ 7.32-2017 Система стандартов по информации, библиотечному и издательскому делу. Отчёт о научно-исследовательской работе. Структура и правила оформления.

2.ГОСТ 7.1-2003 Библиографическая запись. Библиографическое описание. Общие требования и правила составления.