ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени федеральное государственное

Бюджетное учреждение высшего образования

«Московский технический университет связи и информатики»

Кафедра информатики

Лабораторная работа №4

По дисциплине: «Технология разработки программного обеспечения»

Тема: «Рисование фракталов»

Выполнил студент:

Группы: БФИ1902

Шацкий Е. И.

Проверила:

Мосева. М.С.

Москва, 2020 г.

**Содержание**

[1 Цель работы 3](#_Toc59377026)

[2 Задачи на лабораторную работу. 3](#_Toc59377027)

[3 Ход решения лабораторной работы. 3](#_Toc59377028)

[Листинг JImageDisplay 4](#_Toc59377029)

[Листинг класса FractalExplorer 8](#_Toc59377030)

[Листинг класса Mandelbrot 10](#_Toc59377031)

[Листинг класса ComplexNum 11](#_Toc59377032)

[Листинг класса FractalGenerator 11](#_Toc59377033)

[Вывод: 15](#_Toc59377034)

[Список используемых источников 16](#_Toc59377035)

# **1 Цель работы**

Создать небольшоеJAVA-приложение, которое сможет рисовать фракталы.

# **2 Задачи на лабораторную работу.**

Для выполнения лабораторной работы необходимо:

1) Создать класс FractalExplorer, который позволит вам исследовать  
различные области фрактала, путем его создания, отображения через  
графический интерфейс Swing и обработки событий, вызванных  
взаимодействием приложения с пользователем.

2) Создать метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и  
кнопку для сброса отображения. Используйте java.awt.BorderLayout для  
содержимого окна;   
3) Реализуйте вспомогательный метод с типом доступа private для  
вывода на экран фрактала, можете дать ему имя drawFractal ().

4) Создать внутренний класс для обработки событий  
java.awt.event.ActionListener от кнопки сброса. Обработчик должен сбросить  
диапазон к начальному, определенному генератором, а затем перерисовать  
фрактал.  
5) Создайте другой внутренний класс для обработки событий  
java.awt.event.MouseListener с дисплея. \  
6) В заключении, вам необходимо создать статический метод main()  
для FractalExplorer так, чтобы можно было его запустить.

# **3 Ход решения лабораторной работы.**

Для начала создадим класс JImageDisplay, производный отjavax.swing.JComponent. Класс должен иметь одно поле с типом доступаprivate, экземпляр java.awt.image.BufferedImage. Класс BufferedImage управляетизображением, содержимое которого можно записать. Конструктор JImageDisplay должен принимать целочисленные значения ширины и высоты и инициализировать объект BufferedImage новым изображением с этой шириной и высотой, и типом изображения TYPE\_INT\_RGB. Тип определяет, как цвета каждого пикселя будут представлены в изображении. Значение TYPE\_INT\_RGB обозначает, что красные зеленые и синие компоненты имеют по 8 битов, представленные в формате int в указанном порядке. Конструктор также должен вызвать метод setPreferredSize()родительского класса метод с указанной шириной и высотой. Вы должныбудете передать эти значения в объект java.awt.Dimension) Таким образом,когда ваш компонент будет включен в пользовательский интерфейс, онотобразит на экране все изображение. Пользовательские компоненты Swing должны предоставлять свойсобственный код для отрисовки, переопределяя защищенный метод JComponentpaintComponent (Graphics g). Так как наш компонент просто выводит на экранданные изображения, реализация будет очень проста! Во-первых, нужно всегдавызывать метод суперкласса paintComponent (g) так, чтобы объектыотображались правильно. После вызова версии суперкласса, вы можетенарисовать изображение в компоненте, используя следующую операцию:g.drawImage (image, 0, 0, image.getWidth(), image.getHeight(), null);(Мы передаем значение null для ImageObserver, поскольку даннаяфункциональность не требуется.)Вы также должны создать два метода с доступом public для записиданных в изображение: метод clearImage (), который устанавливает все пикселиизображения в черный цвет (значение RGB 0), и метод drawPixel (int x, int y, intrgbColor), который устанавливает пиксель в определенный цвет. Оба методабудут необходимы для использования в методе setRGB () класса BufferedImage.

## Листинг JImageDisplay

import javax.swing.JComponent;  
import java.awt.\*;  
import java.awt.image.BufferedImage;  
  
public class JImageDisplay extends JComponent {//Наследуем от базового класса JComponent, который предназначен для Swing-компонентов, но также и для пользовательских компонентов.  
 private final BufferedImage image;  
  
 public JImageDisplay(int w, int h){//Класс принимает целочисленные значения ширины и высоты  
 if (w <= 0)  
 throw new IllegalArgumentException("w must be > 0; got " + w);  
  
 if (h <= 0)  
 throw new IllegalArgumentException("h must be > 0; got " + h);  
  
 image = new BufferedImage(w, h, BufferedImage.*TYPE\_INT\_RGB*);//Инициализация объекта BufferedImage.  
 Dimension dimension = new Dimension(w, h);//Новое изображение с этой шириной и высотой, и типом изображения TYPE\_INT\_RGB. Тип определяет, как цвета каждого пикселя будут  
  
 super.setPreferredSize(dimension);//Конструктор вызвает метод setPreferredSize() родительского класса метод с указанной шириной и высотой.  
 }  
  
 @Override // Показываю, что далее я собираюсь переопределять метод базового класса.  
 protected void paintComponent(Graphics g) {  
 super.paintComponent(g);// Нужно вызывать метод суперкласса paintComponent (g) так, чтобы объекты отображались правильно  
  
  
 g.drawImage (image, 0, 0, image.getWidth(), image.getHeight(), null);// Рисовка изображения в компоненте  
 }  
  
 public void clearImage() {// Метод, который устанавливает все пиксели изображения в черный цвет (значение RGB 0)  
 Graphics2D imageGraphics = image.createGraphics();  
 imageGraphics.setColor(new Color(0, 0, 0));  
  
 imageGraphics.fillRect(0, 0, image.getWidth(), image.getHeight());  
 }  
  
 public void drawPixel (int x, int y, int rgbColor){  
 image.setRGB(x, y, rgbColor);  
 }// Устанавливает пиксель в определенный цвет.  
}

Далее создам класс FractalExplorer, который позволит мне исследовать  
различные области фрактала. FractalExplorer состоит из JFrame, который в свою очередь содержит объект JImageDisplay, который отображает фрактал, и объект JButton для сброса изображения, необходимый для отображения целого фрактала. Данный макет можно создать, установив для фрейма BorderLayout, затем поместив отображение в центр макета и кнопку сброса в "южной" части макета. Класс FractalExplorer должен отслеживать несколько важных полей для состояния программы:  
1) Целое число «размер экрана», которое является шириной и высотой  
отображения в пикселях. (Отображение фрактала будет квадратным.)  
2) Ссылка JImageDisplay, для обновления отображения в разных  
методах в процессе вычисления фрактала.  
3) Объект FractalGenerator. Будет использоваться ссылка на базовый  
класс для отображения других видов фракталов в будущем.  
4) Объект Rectangle2D.Double, указывающий диапазона комплексной  
плоскости, которая выводится на экран.  
Все вышеприведенные поля будут иметь тип доступа private.

Следом, создам метод createAndShowGUI (), который инициализирует  
графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и  
кнопку для сброса отображения. Используйте java.awt.BorderLayout для  
содержимого окна; добавьте объект отображения изображения в позицию  
BorderLayout.CENTER и кнопку в позицию BorderLayout.SOUTH.  
Дадим окну подходящий заголовок и обеспечить  
операцию закрытия окна по умолчанию (см. метод  
JFrame.setDefaultCloseOperation()). После того, как компоненты пользовательского интерфейса  
инициализированы и размещены, добавим следующую последовательность  
операций:  
frame.pack();  
frame.setVisible(true);  
frame.setResizable(false);  
Данные операции правильно разметят содержимое окна, сделают его  
видимым (окна первоначально не отображаются при их создании для того,  
чтобы можно было сконфигурировать их прежде, чем выводить на экран), и  
затем запретят изменение размеров окна.  
 Далее реализую вспомогательный метод с типом доступа private для  
вывода на экран фрактала, дам ему имя drawFractal(). Этот метод  
должен циклически проходить через каждый пиксель в отображении (т.е.  
значения x и y будут меняться от 0 до размера отображения), сделаю  
следующее:  
⮚Вычислю количество итераций для соответствующих координат в  
области отображения фрактала. Определю координаты с  
плавающей точкой для определенного набора координат пикселей, используя  
вспомогательный метод FractalGenerator.getCoord (); например, чтобы получить координату x, соответствующую координате пикселя X, сделаю следующее:  
//x - пиксельная координата; xCoord - координата в пространстве  
фрактала double xCoord = FractalGenerator.getCoord (range.x, range.x + range.width,displaySize,x);  
⮚Если число итераций равно -1 (т.е. точка не выходит за границы,  
установлю пиксель в черный цвет (для rgb значение 0). Иначе выберу  
значение цвета, основанное на количестве итераций. Можно также для этого  
использовать цветовое пространство HSV: поскольку значение цвета  
варьируется от 0 до 1, получается плавная последовательность цветов от  
красного к желтому, зеленому, синему, фиолетовому и затем обратно красному! Для этого используюю следующий фрагмент:  
float hue = 0.7f + (float) numIters / 200f;  
int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);  
⮚Отображение буду обновлять в соответствии с цветом для  
каждого пикселя.  
⮚После того, как я закончил отрисовывать все пиксели,   
необходимо обновить JimageDisplay в соответствии с текущим изображением.  
Для этого вызову функцию repaint() для компонента.

•Создам внутренний класс для обработки событий  
java.awt.event.ActionListener от кнопки сброса. Обработчик сбрасывает  
диапазон к начальному, определенному генератором, а затем перерисовывает  
фрактал.  
После того, как я создал этот класс, обновлю метод createAndShowGUI  
().  
 Создам другой внутренний класс для обработки событий  
java.awt.event.MouseListener с дисплея. Необходимо обработать события от  
мыши, поэтому я должн унаследовать этот внутренний класс от класса  
MouseAdapterAWT. При получении события о щелчке мышью, класс должен  
отобразить пиксельные кооринаты щелчка в область фрактала, а затем вызвать  
метод генератора recenterAndZoomRange() с координатами, по которым  
щелкнули, и масштабом 0.5. Таким образом, нажимая на какое-либо место на  
фрактальном отображении, я увеличиваете его!  
Кроме того, необходимо перерисовывать фрактал после того, как я поменял область фрактала.  
Далее обновлю метод createAndShowGUI (), чтобы зарегистрировать  
экземпляр этого обработчика в компоненте фрактального отображения.  
 В заключении, необходимо создать статический метод main()  
для FractalExplorer так, чтобы можно было его запустить. В main необходимо  
будет сделать:  
⮚Инициализировать новый экземпляр класса FractalExplorer с  
размером отображения 800.  
⮚Вызовать метод createAndShowGUI () класса FractalExplorer.  
⮚Вызовать метод drawFractal() класса FractalExplorer для  
отображения начального представления.  
После выполнения приведенных выше действий, я смогу детально  
рассмотреть фрактал Мандельброта.

## **Листинг класса FractalExplorer**

import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
  
public class FractalExplorer {  
 private int displaySize;  
 private JImageDisplay display;  
 private FractalGenerator fractal;  
 private Rectangle2D.Double range;  
  
 public FractalExplorer(int size) {  
 displaySize = size;  
  
 fractal = new Mandelbrot();  
 range = new Rectangle2D.Double();  
  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
 }  
  
 public void createAndShowGUI() {  
 display.setLayout(new BorderLayout());  
 JFrame myframe = new JFrame("Fractal Explorer");  
  
 myframe.add(display, BorderLayout.*CENTER*);//Содержимое окна  
  
 JButton resetButton = new JButton("Reset Display");  
  
 Resetter handler = new Resetter();  
 resetButton.addActionListener(handler);  
  
 myframe.add(resetButton, BorderLayout.*SOUTH*);//Установка в нужную позицию  
  
 Clicker click = new Clicker();  
 display.addMouseListener(click);  
  
 myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 myframe.pack();//Правильная разметка окна  
 myframe.setVisible(true);//Правильная разметка окна  
 myframe.setResizable(false);//Правильная разметка окна  
 }  
  
 private void drawFractal() {  
 for (int x = 0; x < displaySize; x++) {//Цикл для обработки каждого пикселя  
 for (int y = 0; y < displaySize; y++) {//Цикл для обработки каждого пикселя  
  
 double xCoord = FractalGenerator.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
  
 double yCoord = FractalGenerator.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1) {  
 display.drawPixel(x, y, 0);  
 } else {  
 float hue = 0.5f + (float) iteration / 50;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
 display.drawPixel(x, y, rgbColor);  
 }  
  
 }  
 }  
 display.repaint();// Для перерисовки (JimageDisplay в соответствии с текущим изображением)  
 }  
  
 private class Resetter implements ActionListener //обработкf событий java.awt.event.ActionListener от кнопки сброса  
  
 {  
 public void actionPerformed(ActionEvent e)  
 {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 }  
  
 private class Clicker extends MouseAdapter  
 {  
 @Override  
 public void mouseClicked(MouseEvent e)  
 {  
 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();  
 }  
 }  
  
 public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

## **Листинг класса Mandelbrot**

import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator {  
 public static final int *LIMIT* = 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) {  
 ComplexNum cmplx = new ComplexNum(0, 0);  
 int iterator = 0;  
  
 while (iterator < *LIMIT* && cmplx.getSquaredModule() < 4) {  
 cmplx.makeSquaredInPoint(x, y);  
  
 iterator++;  
 }  
  
 if (iterator == *LIMIT*) return -1;  
  
 return iterator;  
 }  
}

## **Листинг класса ComplexNum**

public class ComplexNum {  
 public double rl;  
 public double im;  
  
 public ComplexNum(double rl, double im){  
 this.rl = rl;  
 this.im = im;  
 }  
  
 public double getSquaredModule() {  
 return (this.rl \* this.rl + this.im \* this.im);  
 }  
  
 public void makeSquaredInPoint(double x, double y) {  
 double real = (rl \* rl) - (im \* im) + x;//Реальная часть  
 double imagine = 2 \* rl \* im + y;// Мнимая часть  
  
 rl = real;//Реальная часть  
 im = imagine;// Мнимая часть  
 }  
}

## **Листинг класса FractalGenerator**

import java.awt.geom.Rectangle2D;  
  
  
*/\*\*  
 \* This class provides the common interface and operations for fractal  
 \* generators that can be viewed in the Fractal Explorer.  
 \*/*public abstract class FractalGenerator {  
  
 */\*\*  
 \* This static helper function takes an integer coordinate and converts it  
 \* into a double-precision value corresponding to a specific range. It is  
 \* used to convert pixel coordinates into double-precision values for  
 \* computing fractals, etc.  
 \*  
 \** ***@param*** *rangeMin the minimum value of the floating-point range  
 \** ***@param*** *rangeMax the maximum value of the floating-point range  
 \*  
 \** ***@param*** *size the size of the dimension that the pixel coordinate is from.  
 \* For example, this might be the image width, or the image height.  
 \*  
 \** ***@param*** *coord the coordinate to compute the double-precision value for.  
 \* The coordinate should fall in the range [0, size].  
 \*/* public static double getCoord(double rangeMin, double rangeMax,  
 int size, int coord) {  
  
 assert size > 0;  
 assert coord >= 0 && coord < size;  
  
 double range = rangeMax - rangeMin;  
 return rangeMin + (range \* (double) coord / (double) size);  
 }  
  
  
 */\*\*  
 \* Sets the specified rectangle to contain the initial range suitable for  
 \* the fractal being generated.  
 \*/* public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 */\*\*  
 \* Updates the current range to be centered at the specified coordinates,  
 \* and to be zoomed in or out by the specified scaling factor.  
 \*/* public void recenterAndZoomRange(Rectangle2D.Double range,  
 double centerX, double centerY, double scale) {  
  
 double newWidth = range.width \* scale;  
 double newHeight = range.height \* scale;  
  
 range.x = centerX - newWidth / 2;  
 range.y = centerY - newHeight / 2;  
 range.width = newWidth;  
 range.height = newHeight;  
 }  
  
  
 */\*\*  
 \* Given a coordinate <em>x</em> + <em>iy</em> in the complex plane,  
 \* computes and returns the number of iterations before the fractal  
 \* function escapes the bounding area for that point. A point that  
 \* doesn't escape before the iteration limit is reached is indicated  
 \* with a result of -1.  
 \*/* public abstract int numIterations(double x, double y);  
}

После создания необходимых классов осталось все запустить. На рисунке 1 показан результат работы программы.

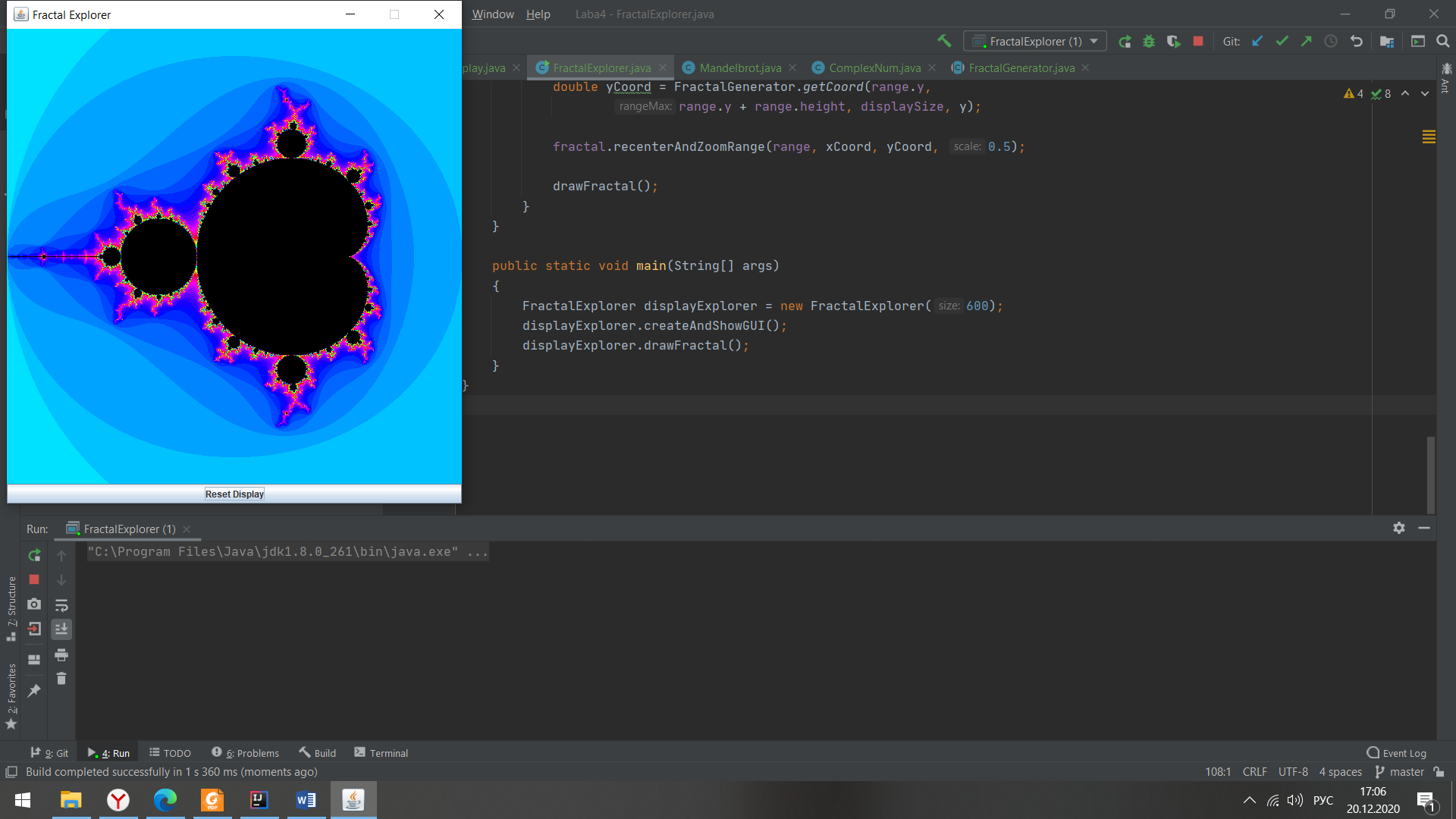


Рисунок 1 – результат работы программы

Далее проверю как работает приближение фрактала. Результат изображен на рисунке 2.

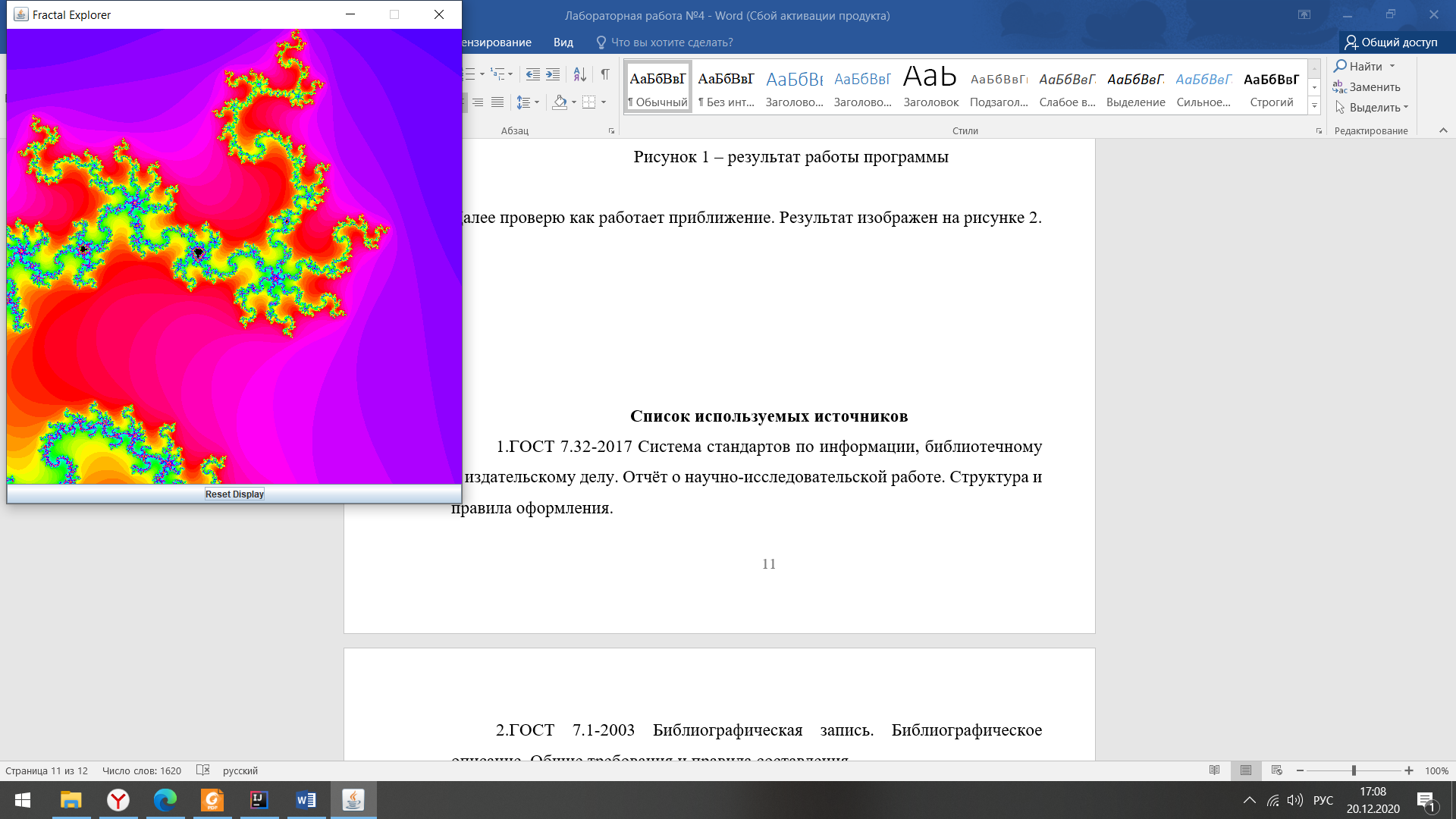


Рисунок 2 – Работа приближения фрактала

# **Вывод:**

Я создал мое первое приложение на java с использованием фреймворка Swing и Java API корректно работает.

# **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.