Московский Государственный Технический Университет

имени Н.Э. Баумана

Факультет ИУ «Информатика и системы управления»

Кафедра ИУ-3 «Информационные системы и телекоммуникации»

Отчет
к Практикуму №5

«Выбор и сохранение фракталов»

по дисциплине «Технологии программирования»

Продолжительность работы: 2 ак. часа

Выполнил: Магомедов В.О.

ИУ3-41Б

Проверила: Руденкова Ю.С.

Москва, 2020

Задание:

Добавить две новые функции к программе Fractal Explorer : возможность отображения нескольких фракталов и выбор фрактала с помощью выпадающего списка, возможность сохранить текущее изображение фрактала в файл.

Поддержка отображения нескольких фракталов

Добавить к Fractal Explorer возможность прорисовки нескольких фракталов довольно легко, у нас уже есть абстрактный класс FractalGenerator. Программный интерфейс Swing API реализует выпадающий список с помощью класса javax.swing.JComboBox, и этот класс выдает событие ActionEvent при выборе нового элемента.

Я создал новые реализации FractalGenerator.

Первая – фрактал треуголка. Сохранила ее в файле Tricorn.java. Как и раньше, сделала класс наследник FractalGenerator, и реализация его почти такая же, как реализация Мандельброта, за исключением двух отличий. Уравнение теперь такое zn = zn-1 2 + c. Разница только в том, что на каждой итерации вычисляется комплексно сопряженное значение zn-1. Начальный диапазон фрактала треуголка должен быть от (-2, -2) до (2, 2).

Второй фрактал, который надо сделать называется "Горящий Корабль", который в действительности выглядит как горящий корабль. Вот детали его реализации: Уравнение для вычисления фрактала zn = (|Re(zn-1)| + i |Im(zn-1)|)2 + c. Другими словами, надо вычислять модуль компонента zn-1 на каждой итерации. Начальный диапазон фрактала от (-2, -2.5) до (2, 1.5).

Выпадающие списки Swing могут работать с коллекциями объектов, но объекты должны иметь метод toString(). Не забуду в каждой реализации фракталов, реализовать метод toString(), который возвращает имя фрактала, т.е. "Мандельброт", "Треуголка", и "Горящий корабль".

Вставить JComboBox в интерфейс не составляет труда. Использую конструктор без аргументов, и метод addItem(Object) для добавления каждой реализации абстрактного класса FractalGenerator. Выпадающий список использует метод toString() объектов генераторов фракталов для отображения их имен.

Добавил к пользовательскому интерфейсу, перед выпадающим списком, текстовое поле, поясняющее роль выпадающего списка. Для этого создала новый объект JPanel, добавила в него объекты JLabel и JComboBox, и затем поместила панель в позицию NORTH содержимого JFrame.

Наконец, добавил обработку событий выпадающего списка к реализации ActionListener. В обработчике следует проверить, что источником события является выпадающий список, и если это так, следует получить из виджета текущий выбранный элемент списка и установить текущий генератор фрактала. (Для этого использую метод getSelectedItem().) Не забуду и сбросить начальный диапазон отображения фрактала.

Сохранение изображения фрактала

Во-первых, добавил на экран программы кнопку "Сохранить изображение". Обе кнопки Сохранить и Сброс поместила в новый объект JPanel, и затем добавила его в область SOUTH объекта JFrame программы как раньше.

Для кнопки Сохранить также надо было предусмотреть обработку событий в ActionListener. Кнопки Сохранить и Сброс имеют собственные значения команд (например, "save" и "reset"), для того чтобы обработчик мог их различить .

В обработчике кнопки "Сохранить ", надо было добавить выбор имени файла, в котором будет сохранено изображение! Это можно очень просто сделать с помощью класса javax.swing.JFileChooser. Этот класс имеет метод showSaveDialog(), который вызывает диалоговое окно "Сохранить файл " в котором пользователь может указать имя сохраняемого файла. Методу передаю ссылку на объект Component, который будет родительским окном диалога выбора файла; это нужно для того чтобы диалог центрировался относительно родительского окна. Подставил в этот параметр ссылку на объект JFrame.

Метод возвращает значение типа int, которое определяет результат операции выбора файла. Если метод возвращает константу JFileChooser.APPROVE_OPTION, можно продолжить операцию записи файла; иначе, пользователь отменил запрос, и надо просто вернуть управление. Если пользователь выбрал место для сохранения файла, его можно получить через метод getSelectedFile() который возвращает объект типа File. Настроил диалог выбора файла так, чтобы он сохранял только изображения PNG; так как это пока единственный поддерживаемый формат. Это делается с помощью класса javax.swing.filechooser.FileNameExtensionFilter, вот таким образом:

```
JFileChooser chooser = new JFileChooser();

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

chooser.setFileFilter(filter);

chooser.setAcceptAllFileFilterUsed(false);
```

Последняя строчка нужна для того чтобы нельзя было ввести файлы с другим расширением (не ".png").Если пользователь выбрал файл, следующий шаг сохранение изображения фрактала на диск. Это опять могло бы вызвать сложности, но Java снова предлагает для этого готовые функции. Класс javax.imageio.ImageIO предлагает простые операции загрузки и сохранения изображений; можно использовать метод write(RenderedImage im, String formatName, File output). Формат файла будет "png". "RenderedImage" это просто экземпляр класса theBufferedImage и вашего компонента JImageDisplay. (Этот член класса должен иметь модификатор доступа public.)

Конечно, я обратил внимание на то что метод write() может вызывать исключения, поэтому я поместила вызов этого метода в блок try/catch, чтобы

обрабатывать потенциальные ошибки. Блок catch сообщает пользователю об ошибке с помощью диалогового окна. Для этого в Swing есть класс javax.swing.JOptionPane, упрощающий процесс создания информационных диалогов, или диалогов требующих ответа да/нет. В нашем случае, надо было использовать статический метод JOptionPane.showMessageDialog(Component parent, Object message, String title, int messageType), с типом сообщения JOptionPane.ERROR_MESSAGE. Сообщение об ошибке можно получить, вызвав метод getMessage() объекта исключения, а заголовок должен иметь смысл, например "Нельзя сохранить изображение". Как и раньше, JFrame должен быть родительским компонентом, для того чтобы диалог правильно центрировался в рамке главного окна.

Код:

В IDE IJ Idea изменили класс FractalExplorer.

```
package com.company;

import java.awt.Color;
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.FileNotFoundException;
import java.io.IOException;

import javax.imageio.ImageIO;
import javax.swing.JButton;
import javax.swing.JComboBox;
import javax.swing.JFileChooser;
import javax.swing.JFrame;
import javax.swing.JLabel;
import javax.swing.JOptionPane;
import javax.swing.JPanel;
import javax.swing.filechooser.FileFilter;
import javax.swing.filechooser.FileNameExtensionFilter;

/**
 * Этот класс предоставляет графический интерфейс для отображения фракталов.
 */
public class FractalExplorer {
    /** Длина стороны квадратной области дисплея.**/
    private int dispSize;
    /** Область изображения для  фрактала. **/
    private JImageDisplay img;
    /** Используется для генерации фракталов определенного вида. **/
    private FractalGenerator fGen;
    /** Задает отображаемый диапазон в комплексной области.**/
    private Rectangle2D.Double range;
    /** Различные компоненты графического интерфейса. **/
    JFrame frame;
    JButton resetButton;
    JButton saveButton;
    JLabel label;
    JComboBox<FractalGenerator> fractalCBox;
    JPanel cbPanel;
    JPanel buttonPanel;

    /** Базовый конструктор. Инициализирует отображение изображения, генератор
фракталов,
     * и начальная зона просмотра.
```

```
     */
    public FractalExplorer(int dispSize) {
        this.dispSize = dispSize;
        this.fGen = new Mandelbrot();
        this.range = new Rectangle2D.Double(0, 0, 0, 0);
        fGen.getInitialRange(this.range);

    }
    /**
     * Настройка и отображение графического интерфейса пользователя.
     */
    public void createAndShowGUI() {
// Создание компонентов графического интерфейса.
        frame = new JFrame("Fractal Explorer");
        img = new JImageDisplay(dispSize, dispSize);
        resetButton = new JButton("Reset Display");
        resetButton.setActionCommand("reset");
        saveButton = new JButton("Save Image");
        saveButton.setActionCommand("save");
        label = new JLabel("Fractal: ");
        fractalCBox    = new JComboBox<FractalGenerator>();
        cbPanel = new JPanel();
        cbPanel.add(label);
        cbPanel.add(fractalCBox);
        buttonPanel = new JPanel();
        buttonPanel.add(saveButton);
        buttonPanel.add(resetButton);
        fractalCBox.addItem(new Mandelbrot());
        fractalCBox.addItem(new BurningShip());
        fractalCBox.addItem(new Tricorn());

        //Реакции
        ActionHandler aHandler = new ActionHandler();
        MouseHandler mHandler = new MouseHandler();
        resetButton.addActionListener(aHandler);
        saveButton.addActionListener(aHandler);
        img.addMouseListener(mHandler);
        fractalCBox.addActionListener(aHandler);

        // Рамка
        frame.setLayout(new java.awt.BorderLayout());
        frame.add(img, java.awt.BorderLayout.CENTER);
        frame.add(buttonPanel, java.awt.BorderLayout.SOUTH);
        frame.add(cbPanel, java.awt.BorderLayout.NORTH);
        frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);

        // Показываем изображение
        frame.pack();
        frame.setVisible(true);
        frame.setResizable(false);
    }

    /** Используем генератор фракталов, чтобы нарисовать фрактал пиксель за
пикселем. **/
    private void drawFractal() {
        for (int i = 0; i < dispSize; i++) {
            for (int j = 0; j < dispSize; j++) {
                double xCoord = FractalGenerator.getCoord(range.x,
                        range.x + range.width, dispSize, i);
                double yCoord = FractalGenerator.getCoord(range.y,
                        range.y + range.width, dispSize, j);
                double numIters = fGen.numIterations(xCoord, yCoord);

                if (numIters == -1) {
                    // Пикселя в наборе нет. Красим его в черный цвет.
                    img.drawPixel(i, j, 0);
                }
                else {
```

```
                    /** Пиксель находится во фрактальном множестве.
                     * Раскрасим пиксель в зависимости от количества
итераций.**/
                    float hue = 0.7f + (float) numIters / 200f;
                    int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);
                    img.drawPixel(i, j, rgbColor);
                }
            }
        }
        /** Обновляем фракатальное изубражение **/
        img.repaint();
    }

    /**
     * Сброс масштаба, изменения фрактального типа, сохранение изображений
     **/
    public class ActionHandler implements ActionListener {
        /** Действия.**/
        public void actionPerformed(ActionEvent e) {
            /** Если нажата кнопка сброса, сбрасываем масштаб.**/
            if (e.getActionCommand() == "reset") {
                fGen.getInitialRange(range);
                drawFractal();
            }
            /** Если нажимаем кнопку Сохранить, сохраняем изображение. **/
            else if (e.getActionCommand() == "save") {
                JFileChooser fileChooser = new JFileChooser();
                FileFilter filter
                        = new FileNameExtensionFilter("PNG Images", "png");
                fileChooser.setFileFilter(filter);
                fileChooser.setAcceptAllFileFilterUsed(false);
                int res = fileChooser.showSaveDialog(img);

                if (res == JFileChooser.APPROVE_OPTION) {
                    try {
                        javax.imageio.ImageIO.write(img.getBufferedImage(),
                                "png", fileChooser.getSelectedFile());
                    } catch (NullPointerException | IOException e1) {
                        javax.swing.JOptionPane.showMessageDialog(img,
                                e1.getMessage(), "Cannot Save Image",
                                JOptionPane.ERROR_MESSAGE);
                    }
                }
                else {
                    return;
                }
            }
            /** Если нажимаем кнопку combobox, изменяем фрактальные типы. **/
            else if (e.getSource() == (Object) fractalCBox) {
                fGen = (FractalGenerator) fractalCBox.getSelectedItem();
                fGen.getInitialRange(range);
                drawFractal();
            }
        }
    }

    /** Увеличиваем масштаб **/
    public class MouseHandler extends MouseAdapter {
        @Override
        public void mouseClicked(MouseEvent e) {
            double xCoord = FractalGenerator.getCoord(range.x,
                    range.x + range.width, dispSize, e.getX());
            double yCoord = FractalGenerator.getCoord(range.y,
                    range.y + range.width, dispSize, e.getY());
            fGen.recenterAndZoomRange(range, xCoord, yCoord, 0.5);
            drawFractal();
        }
    }
```

```
    /** Запуск приложения **/
    public static void main(String[] args) {
        FractalExplorer fracExp = new FractalExplorer(800);
        fracExp.createAndShowGUI();
        fracExp.drawFractal();
    }
}
```

В IDE IJ Idea создали файл BurningShip.java и в нем класс BurningShip.

```
package com.company;

import java.awt.geom.Rectangle2D;

public class BurningShip extends FractalGenerator {
    /**
     * Максимальное число итераций перед объявлением точки BurningShip set.
     **/
    public static final int MAX_ITERATIONS = 2000;

    /** Интересующая нас область комплексной плоскости для фрактала.**/
    public void getInitialRange(Rectangle2D.Double range) {
        range.x = -2;
        range.y = -2.5;
        range.width = 4;
        range.height = 4;
    }

    /**
     * Эта функция вычисляет необходимое количество итераций.
     * Для BurningShip set итерационная функция выглядит следующим образом:
     * z_n = (z_{n-1})^2 + c. Тут используем модуль на каждой итерации
     **/
    public int numIterations(double a, double b) {
        // Квадрат величины c.
        double magSq;
        // Действительная часть z_i.
        double re = a;
        // Мнимая часть z_i.
        double im = b;
        // Действительная часть z_{i+1}.
        double nextRe;
        // Мнимая часть z_{i+1}.
        double nextIm;
        // Переменная для подсчета итераций.
        int i = 0;

        while (i < MAX_ITERATIONS) {
            i += 1;
            nextRe = a + Math.abs(re) * Math.abs(re)
                    - Math.abs(im) * Math.abs(im);
            nextIm = b + 2 * Math.abs(re) * Math.abs(im);
            re = nextRe;
            im = nextIm;
            magSq = re * re + im * im;
            if (magSq > 4) {
                return i;
            }
        }
        return -1;
    }

    public String toString() {
```

```
        return "Burning Ship";
    }
}
```

В IDE IJ Idea создали файл Tricorn.java и в нем класс Tricorn.

```
package com.company;

import java.awt.geom.Rectangle2D;

public class Tricorn extends FractalGenerator {
    /**
     * Максимальное число итераций перед объявлением точки в Tricorn set.
     **/
    public static final int MAX_ITERATIONS = 2000;

    /** Интересующая нас область комплексной плоскости для фрактала.**/
    public void getInitialRange(Rectangle2D.Double range) {
        range.x = -2;
        range.y = -2;
        range.width = 4;
        range.height = 4;
    }

    /**
     * Эта функция вычисляет необходимое количество итераций.
     * Для Tricorn set итерационная функция выглядит следующим образом:
     * zn = zn-1^2 + c. Тут на каждой итерации вычисляется комплексно
сопряженное значение.
     **/
    public int numIterations(double a, double b) {
        // Квадрат величины c.
        double magSq;
        // Действительная часть z_i.
        double re = a;
        // Мнимая часть z_i.
        double im = b;
        // Действительная часть z_{i+1}.
        double nextRe;
        // Мнимая часть z_{i+1}.
        double nextIm;
        // Переменная для подсчета итераций.
        int i = 0;

        while (i < MAX_ITERATIONS) {
            i += 1;
            nextRe = a + re * re - im * im;
            nextIm = b + -2 * re * im;
            re = nextRe;
            im = nextIm;
            magSq = re * re + im * im;
            if (magSq > 4) {
                return i;
            }
        }
        return -1;
    }

    public String toString() {
        return "Tricorn";
    }
}
```

Скриншоты

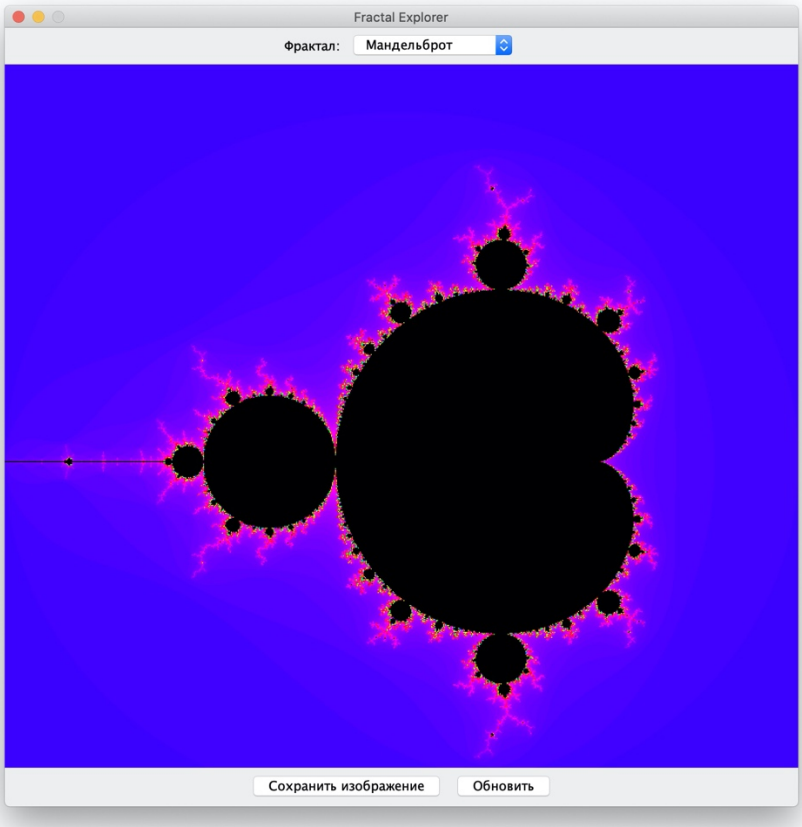

Рис.1 Отображение фрактала Mandelbrot

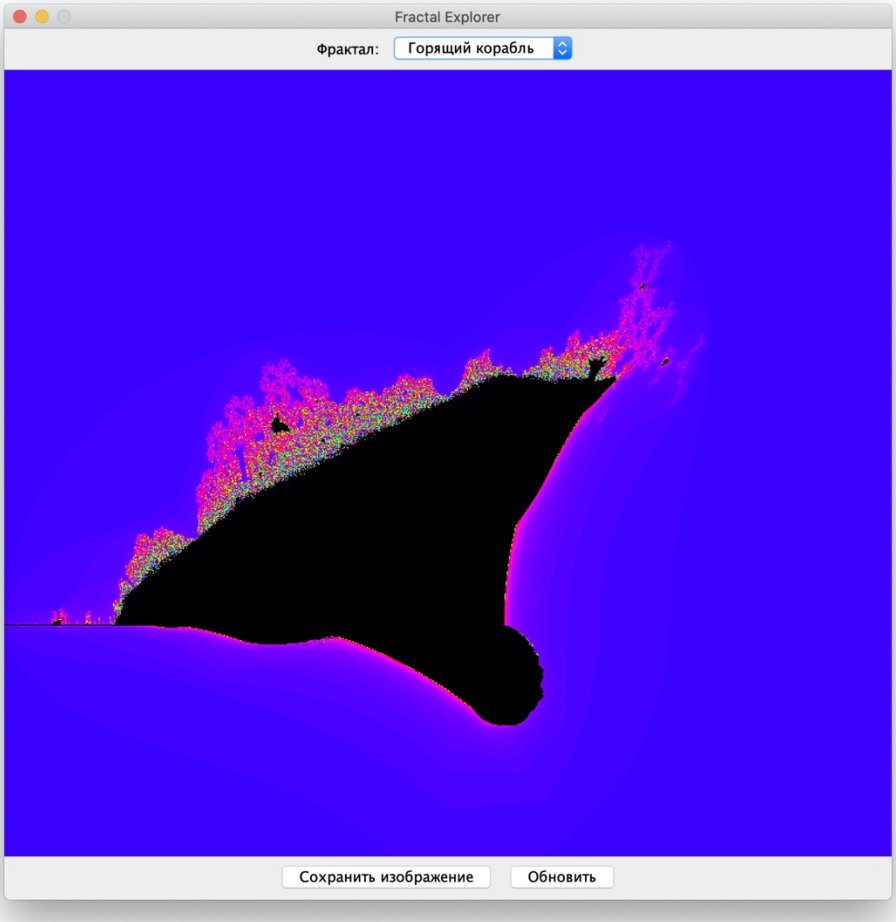

Рис.2 Отображение фрактала «горящий корабль»

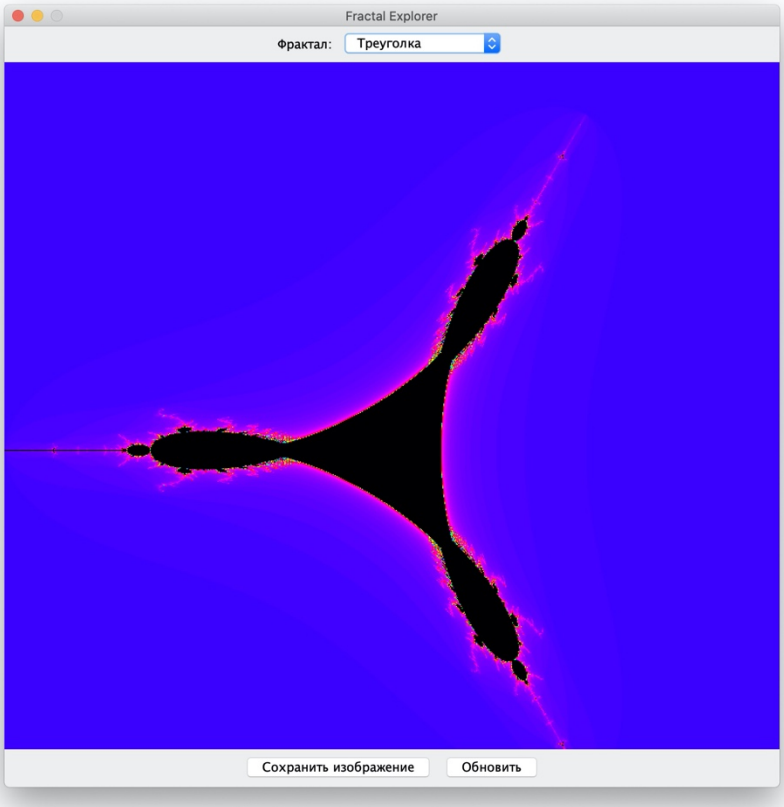

Рис.3 Отображение фрактала «треуголка»

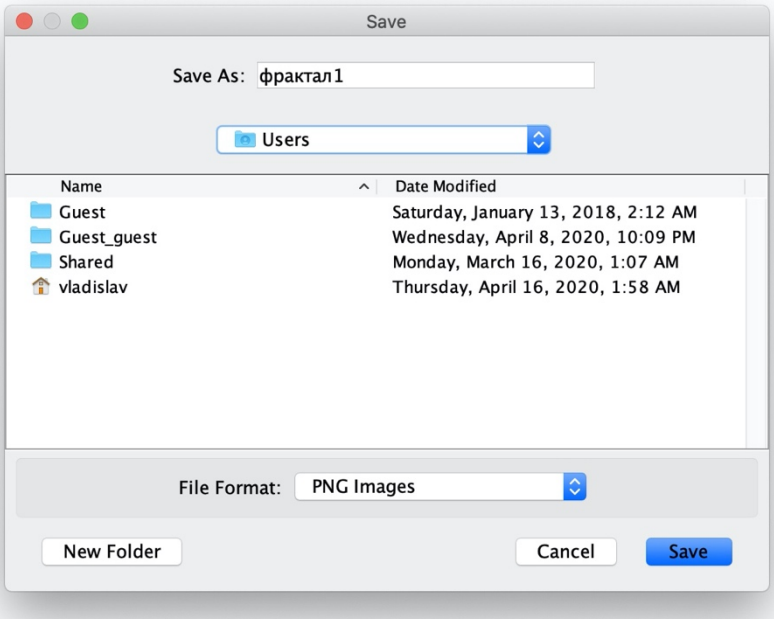

Рис.4 Сохранение изображения

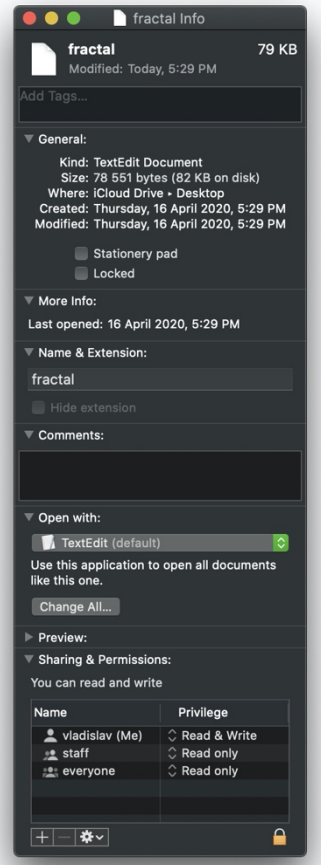

Рис.5 Свойства сохраненной картинки

Вывод:

В этой лабораторной работе было изменено Java приложение, которое может рисовать удивительные изображения фракталов. Мы добавили возможность сохранять изображения фракталов на диск, выбор реализации фракталов: «Горящий корабль» и «Треуголка» и сами эти реализации. Мы создали файлы Tricorn.java и BurningShip.java и в них классы для реализации этих двух методов.