ФЕДЕРАЛЬНОЕ АГЕНСТВО СВЯЗИ

Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего образования

«Московский Технический Университет Связи и Информатики» (МТУСИ)

Кафедра МКиИТ

Лабораторная работа №4

«Выбор и сохранение фрактала»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Смышляев Владислав

Вариант №15

Москва, 2018

Содержание

[1. Цель и задачи 3](#_Toc512781123)

[2. Анализ предметной области, выбор инструментария 3](#_Toc512781124)

[3. Объяснение функций 3](#_Toc512781125)

[3.1. Класс act\_listener 3](#_Toc512781126)

[3.2. Класс BurningShip 4](#_Toc512781127)

[3.3. Класс Tricorn 5](#_Toc512781128)

[4. Пример работы программы 7](#_Toc512781129)

[4.1. Фрактал Мандельброта 7](#_Toc512781130)

[4.2. Фрактал горящий корабль 8](#_Toc512781131)

[4.3. Фрактал трёхконечник 9](#_Toc512781132)

[5. Выводы 9](#_Toc512781133)

1. Цель и задачи

Цель работы: Изучить основы работы с интерфейсом сохранения файлов в Java и создать выбор из рисования фракталов.

Задачи: Написать программу сохранение нарисованного фрактала, используя части кода из предыдущей работы, а также добавить классы новых фракталов и способ выбора между ними.

1. Анализ предметной области, выбор инструментария

Для написания данной программы использоваться IDE IntelliJ Idea, так как в данной и последующих задачах требуются средства для отладки программы. Здесь также представлены средства для работы с контролем версий, но для собственного удобства коммиты будет совершаться через Visual Studio Code.

1. Объяснение функций

Так как, данная работа основа на предыдущей здесь будут представлены лишь модификации кода, которые требовались для выполнения задачи.

* 1. Класс act\_listener

*/\*\*  
 \* Inner class for listening button clicks  
 \*/*private class act\_listener implements ActionListener {  
 */\*\*  
 \* Any object using this listener will reset the fractal to its 1.0 scale  
 \* any time an action on it is performed (ex. Button was clicked)  
 \** ***@param*** *e - occured event  
 \*/* @Override  
 public void actionPerformed(ActionEvent e)  
 {  
 //Getting the action command  
 String cmd = e.getActionCommand();  
  
 if (e.getSource() == box) { //Check for source being combo\_box  
 generator = (FractalGenerator) box.getSelectedItem();  
 generator.getInitialRange(plane\_range);  
 display.clearImage();  
 drawFractal();  
 }  
 else if (cmd.equals("Reset")) { //Check for source being reset button  
 generator.getInitialRange(plane\_range);  
 display.clearImage();  
 drawFractal();  
 }  
 else if (cmd.equals("Save")) { //Check for source being save button  
 JFileChooser chooser = new JFileChooser();  
 FileFilter filter = new FileNameExtensionFilter("PNG Files","png");  
 chooser.setFileFilter(filter);  
 chooser.setAcceptAllFileFilterUsed(false);  
 if (chooser.showSaveDialog(frame) == JFileChooser.*APPROVE\_OPTION*) { //Attempt to save file if user chose to save  
 try { //Exception handling if anything went wrong during writing  
 ImageIO.*write*(display.getImage(),"png",chooser.getSelectedFile());  
 }  
 catch (IOException ex) {  
 JOptionPane.*showMessageDialog*(frame,ex.getMessage(),"Couldn't save the image",JOptionPane.*ERROR\_MESSAGE*);  
 }  
 }  
 else {  
 return;  
 }  
  
 }  
 }  
}

Внутренний класс класса FractalExplorer изменён для определения части интерфейса, которым воспользовался пользователь, будь то, выбор фрактала, сброс или сохранение фрактала в файл.

* 1. Класс BurningShip

import java.awt.geom.Rectangle2D;  
  
public class BurningShip extends FractalGenerator {  
  
 */\*\*Maximum amount of iterations until concluding that the point was set\*\*/* public static final int *MAX\_ITER* = 2000;  
  
 */\*\*  
 \* Resets the given range to 1.0 scale of the fractal  
 \** ***@param*** *range - fractal range to modify  
 \*/* @Override  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2.5;  
 range.width = 4;  
 range.height = 4;  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *x - x coordinate to check  
 \** ***@param*** *y - y coordinate to check  
 \** ***@return*** *returns number of iterations that was required to reach the x,y otherwise returns -1 reaching the limit  
 \*/* @Override  
 public int numIterations(double x, double y)  
 {  
 double re = 0;  
 double im = 0;  
 for(int i=1;i<*MAX\_ITER*;i++)  
 {  
 //Using the simple sum of 2 numbers squared formula  
 //(a+b)^2 = a^2 + 2\*a\*b + b^2  
 //(re-im)^2 = re^2 - 2\*re\*im + im^2  
 double nextRe = re\*re - im\*im + x;  
 double nextIm = Math.*abs*(2\*re\*im) + y;  
  
 // If we reached the Mandelbrot's condition return the amount of iterations  
 if ((im\*im+re\*re) > 4)  
 return i;  
  
 //Reassigning the real and imaginary parts for next iteration  
 re = nextRe;  
 im = nextIm;  
 }  
 return -1;  
 }  
  
 @Override  
 public String toString() {  
 return "Burning Ship";  
 }  
}

Такой же, как и класс Mandelbrot только использована другая математическая функция, также была переписана функция toString, для отображении имени фрактала.

* 1. Класс Tricorn

import java.awt.geom.Rectangle2D;  
  
public class Tricorn extends FractalGenerator {  
  
 */\*\*Maximum amount of iterations until concluding that the point was set\*\*/* public static final int *MAX\_ITER* = 2000;  
  
 */\*\*  
 \* Resets the given range to 1.0 scale of the fractal  
 \** ***@param*** *range - fractal range to modify  
 \*/* @Override  
 public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -2;  
 range.width = 4;  
 range.height = 4;  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *x - x coordinate to check  
 \** ***@param*** *y - y coordinate to check  
 \** ***@return*** *returns number of iterations that was required to reach the x,y otherwise returns -1 reaching the limit  
 \*/* @Override  
 public int numIterations(double x, double y)  
 {  
 double re = 0;  
 double im = 0;  
 for(int i=1;i<*MAX\_ITER*;i++)  
 {  
 //Using the simple sum of 2 numbers squared formula  
 //(a+b)^2 = a^2 + 2\*a\*b + b^2  
 //(re-im)^2 = re^2 - 2\*re\*im + im^2  
 double nextRe = re\*re - im\*im + x;  
 double nextIm = -2\*re\*im + y;  
  
 // If we reached the Mandelbrot's condition return the amount of iterations  
 if ((im\*im+re\*re) > 4)  
 return i;  
  
 //Reassigning the real and imaginary parts for next iteration  
 re = nextRe;  
 im = nextIm;  
 }  
 return -1;  
 }  
  
 @Override  
 public String toString() {  
 return "Tricorn";  
 }  
  
}

То же, что и Mandelbrot и BurningShip.

В остальном программа не изменилась.

1. Пример работы программы
   1. Фрактал Мандельброта

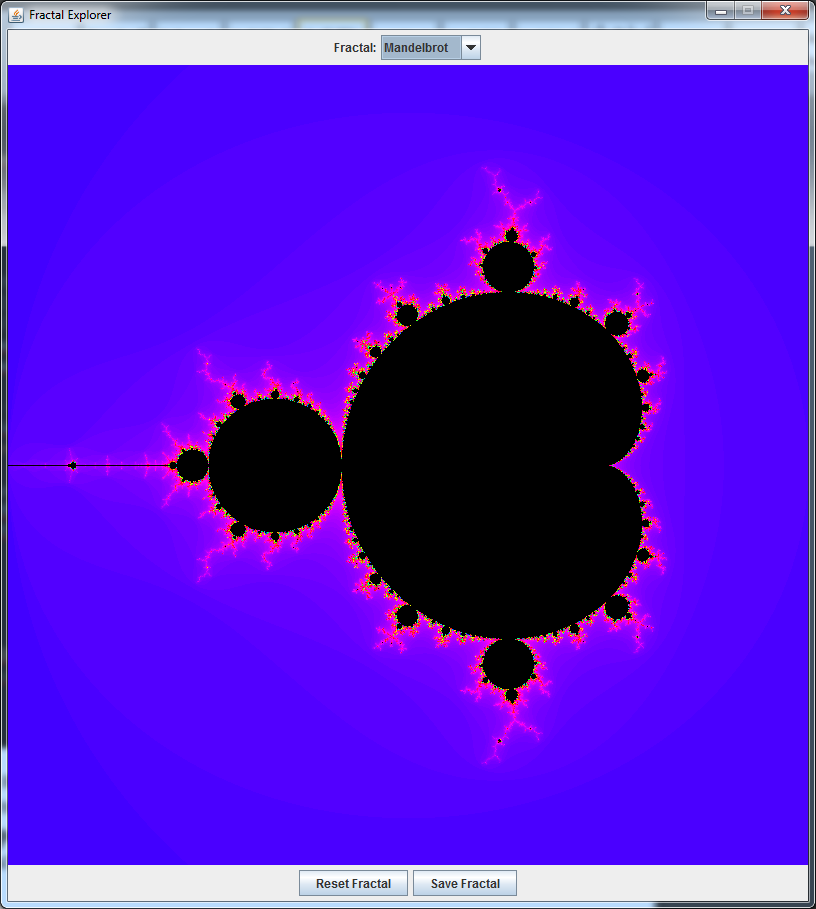


Рисунок 1

* 1. Фрактал горящий корабль

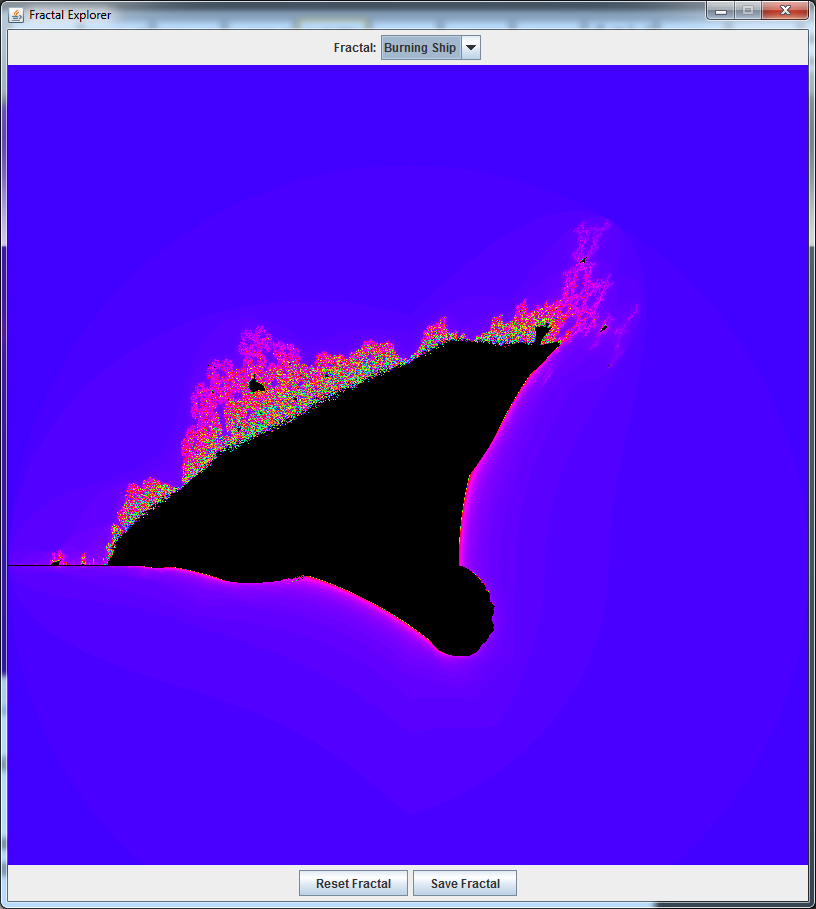


Рисунок 2

* 1. Фрактал трёхконечник

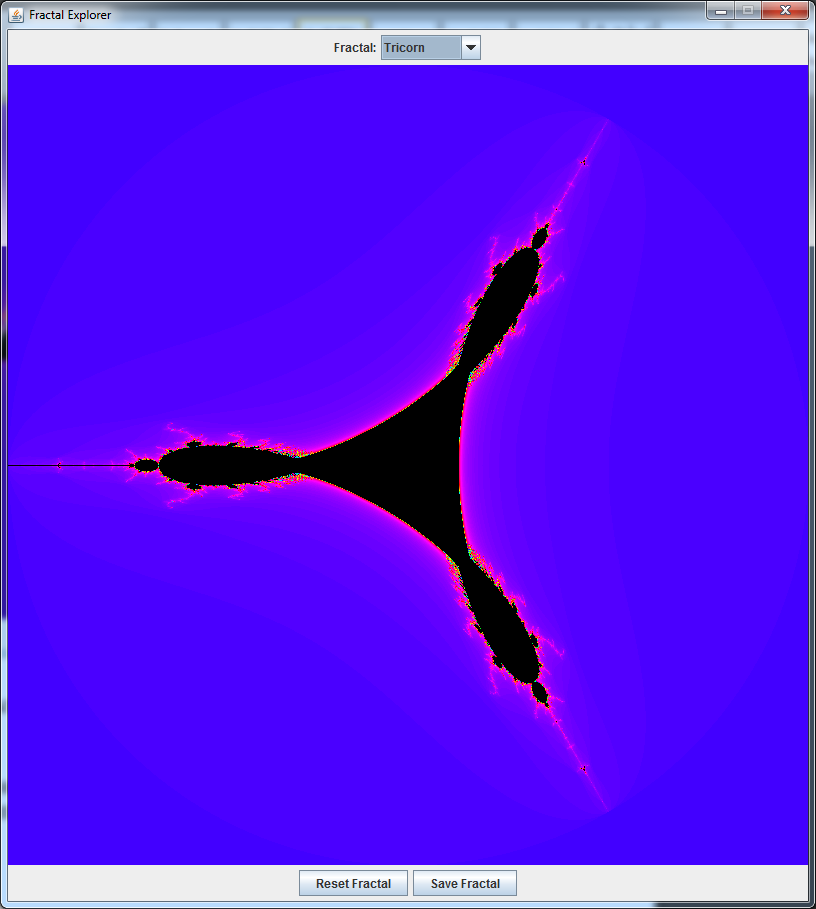


Рисунок 3

1. Выводы

Таким образом, были изучены основы сохранения в файл, а также способы использования уже написанного кода.