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 №3

«Фрактал Эксплорер»

Выполнил:

Студент 2 курса

Группы БСТ-1602

Смышляев Владислав

Вариант №15

Москва, 2018

Содержание

[1. Цель и задачи 3](#_Toc512780081)

[2. Анализ предметной области, выбор инструментария 3](#_Toc512780082)

[3. Объяснение функций 3](#_Toc512780083)

[3.1. Класс JImageDisplay 3](#_Toc512780084)

[3.2. Класс Mandelbrot 4](#_Toc512780085)

[3.3. Класс FractalGenerator 5](#_Toc512780086)

[3.4. Класс FractalExplorer 7](#_Toc512780087)

[4. Пример работы программы 10](#_Toc512780088)

[5. Выводы 11](#_Toc512780089)

1. Цель и задачи

Цель работы: Изучить основы принципа наследования и познакомится с так называемыми событиями в Java, а также составлять собственный графический интерфейс.

Задачи: Написать программу рисующую фрактал Мандельброта, используя принцип наследования.

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
   1. Класс JImageDisplay

import javax.swing.JComponent;  
import java.awt.image.BufferedImage;  
import java.awt.Graphics;  
  
public class JImageDisplay extends JComponent {  
 private BufferedImage image;  
  
 */\*\*  
 \* Constructor of our display  
 \** ***@param*** *imWid - width of the display  
 \** ***@param*** *imHei - height of the display  
 \*/* JImageDisplay(int imWid,int imHei)  
 {  
 image = new BufferedImage(imWid,imHei,BufferedImage.*TYPE\_INT\_RGB*);  
 super.setPreferredSize(new java.awt.Dimension(imWid,imHei));  
 }  
  
 */\*\*  
 \*  
 \** ***@param*** *g - Drawing the display on screen  
 \*/* @Override  
 public void paintComponent(Graphics g)  
 {  
 super.paintComponent(g);  
 g.drawImage(image,0,0,image.getWidth(),image.getHeight(),null);  
 }  
  
 */\*\*  
 \* Set the image to black color  
 \*/* public void clearImage()  
 {  
 //Initialize the array of 0s  
 int arr\_len;  
 if (image.getHeight() > image.getWidth()) arr\_len = image.getHeight();  
 else arr\_len = image.getWidth();  
 int[] rgb = new int[arr\_len];  
 //Set the whole image to black color  
 image.setRGB(0,0,image.getWidth()-1,image.getHeight()-1,rgb,0,0);  
 }  
  
 */\*\*  
 \* Draws a pixel with a given color to coordinates x,y  
 \** ***@param*** *x - x position of pixel  
 \** ***@param*** *y - y position of pixel  
 \** ***@param*** *rgbColor - color of pixel in RGB color set  
 \*/* public void drawPixel(int x,int y, int rgbColor)  
 {  
 image.setRGB(x,y,rgbColor);  
 }  
}

Данный класс будет холстом, куда будет отрисовываться фрактал. Для обеспечения работы данный класс, во-первых, наследует класс JComponent. Во-вторых, определяет ряд функций таких как, очистка изображение, рисование пикселя и переопределение функции отрисовки.

* 1. Класс Mandelbrot

import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot 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 = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
  
 */\*\*  
 \*  
 \** ***@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;  
 }  
  
}

Данный класс позволяет найти количество итераций для достижения точки, а также задаёт размеры фрактала по умолчанию. Для нахождения количества итераций используется формула фрактала, то есть для каждого фрактала придётся писать отдельный класс. Стоит отметить, что данный класс наследует пользовательский класс FractalGenerator.

* 1. Класс 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);  
}

Данный класс служит основой для каждого фрактала, который необходимо будет рисовать. Здесь определяются такие методы как конвертация координат из пикселей в координаты необходимые фракталу и центровка положения фрактала на картинке.

* 1. Класс FractalExplorer

import javax.swing.\*;  
import java.awt.\*;  
import java.awt.event.\*;  
import java.awt.geom.Rectangle2D;  
  
public class FractalExplorer {  
  
 */\*\*  
 \* Program's entry point  
 \** ***@param*** *args - Array of strings (Unused)  
 \*/* public static void main(String[] args)  
 {  
 //Create the explorer, initialize GUI and draw the fractal  
 FractalExplorer expl = new FractalExplorer(800);  
 expl.createAndShowGUI();  
 expl.drawFractal();  
 }  
  
  
 private int dispSize;  
 private JImageDisplay display;  
 private FractalGenerator generator;  
 private Rectangle2D.Double plane\_range = new Rectangle2D.Double(0,0,0,0);  
  
 FractalExplorer(int disp\_size)  
 {  
 dispSize = disp\_size;  
 display = new JImageDisplay(disp\_size,disp\_size);  
 generator = new Mandelbrot();  
 generator.getInitialRange(plane\_range);  
 }  
  
 */\*\*  
 \* GUI Initialization  
 \*/* public void createAndShowGUI()  
 {  
 //Create the main window  
 JFrame frame = new JFrame("Fractal Explorer");  
  
 //Add our fractal display  
 display.addMouseListener(new mouse\_listener());  
 frame.add(display,BorderLayout.CENTER);  
  
 //Add the button  
 JButton bt = new JButton("Reset");  
 bt.addActionListener(new act\_listener());  
 frame.add(bt,BorderLayout.SOUTH);  
  
 //Set the default close action to "exit"  
 frame.setDefaultCloseOperation(JFrame.EXIT\_ON\_CLOSE);  
  
 //Finallize the window  
 frame.pack();  
 frame.setVisible(true);  
 frame.setResizable(false);  
 }  
  
 */\*\*  
 \* Fractal drawing  
 \*/* private void drawFractal()  
 {  
 for (int i=0;i<dispSize;i++)  
 {  
 for (int j=0;j<dispSize;j++)  
 {  
 //Map display coordinates to fractal coordinates  
 double xCoord = FractalGenerator.getCoord(plane\_range.x,plane\_range.x + plane\_range.width,dispSize,i);  
 double yCoord = FractalGenerator.getCoord(plane\_range.y,plane\_range.y + plane\_range.height,dispSize,j);  
  
 //Get the iterations amount  
 int iters = generator.numIterations(xCoord,yCoord);  
  
 //Default pixel color to black  
 int px\_color = 0;  
  
 //Check if we are in the limit of our fractal  
 if (iters != -1)  
 {  
 //Calcualte the pixel color  
 float hue = 0.7f + (float) iters / 200f;  
 px\_color = Color.HSBtoRGB(hue,1f,1f);  
 }  
 //Draw the pixel  
 display.drawPixel(i,j,px\_color);  
 }  
 }  
 //Display what we drew on screen  
 display.repaint();  
 }  
  
 */\*\*  
 \* 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)  
 {  
 generator.getInitialRange(plane\_range);  
 //generator.recenterAndZoomRange(plane\_range,plane\_range.x/2,plane\_range.y/2,1.0);  
 display.clearImage();  
 drawFractal();  
 }  
 }  
  
 */\*\*  
 \* Inner class for listening mouse click events  
 \*/* private class mouse\_listener extends MouseAdapter {  
  
 */\*\*  
 \* Any instance of this listener zooms into fractal to where the mouse was positioned on the image  
 \* after a mouse click was performed  
 \** ***@param*** *e - mouse event that occured (basically mouse object)  
 \*/* @Override  
 public void mouseClicked(MouseEvent e) {  
 super.mouseClicked(e);  
 //Get mouse's x and y position  
 int mouse\_x = e.getX();  
 int mouse\_y = e.getY();  
  
 //Zoom scaling  
 final double SCALE = 0.5;  
  
 //Map mouse position to fractal coordinates  
 double xCoord = FractalGenerator.getCoord(plane\_range.x,plane\_range.x+plane\_range.width,dispSize,mouse\_x);  
 double yCoord = FractalGenerator.getCoord(plane\_range.y,plane\_range.y+plane\_range.height,dispSize,mouse\_y);  
  
 //Finally zoom in, clear the image and draw the fractal  
 generator.recenterAndZoomRange(plane\_range,xCoord,yCoord,SCALE);  
 display.clearImage();  
 drawFractal();  
 }  
  
 }  
}

С данного класса начинается выполнение программы, здесь определяется весь графический пользовательский интерфейс. Определяются внутренние классы для “прослушивания” событий и, конечно же, рисование самого фрактала. Как обычно работа и назначение каждой функции описаны в комментариях к ним.

1. Пример работы программы

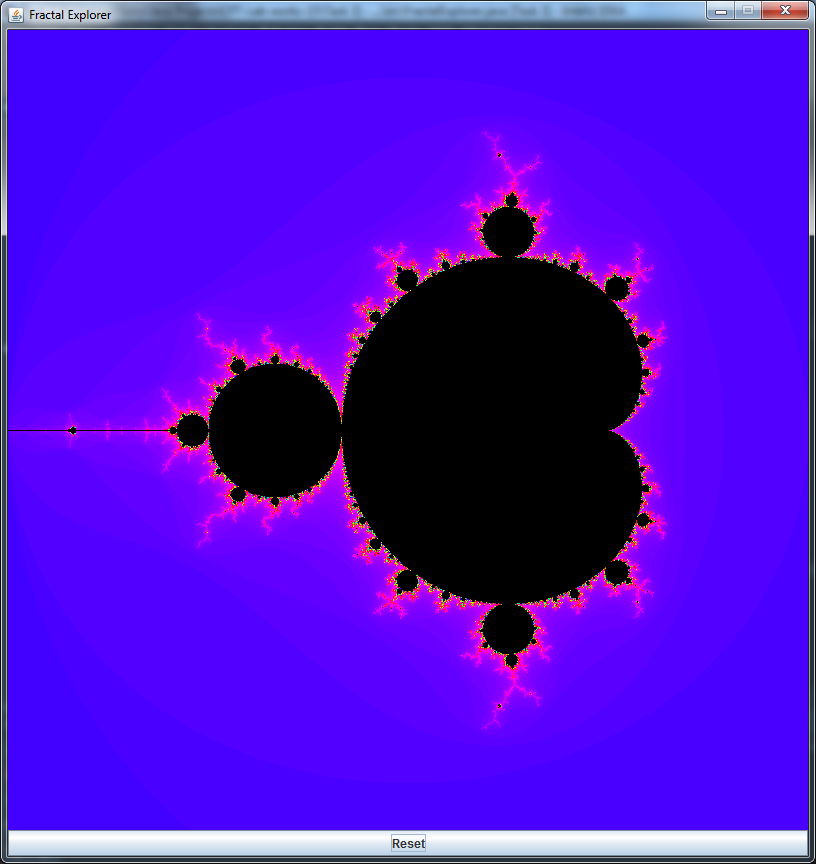


Рисунок 1 – Программа в действии

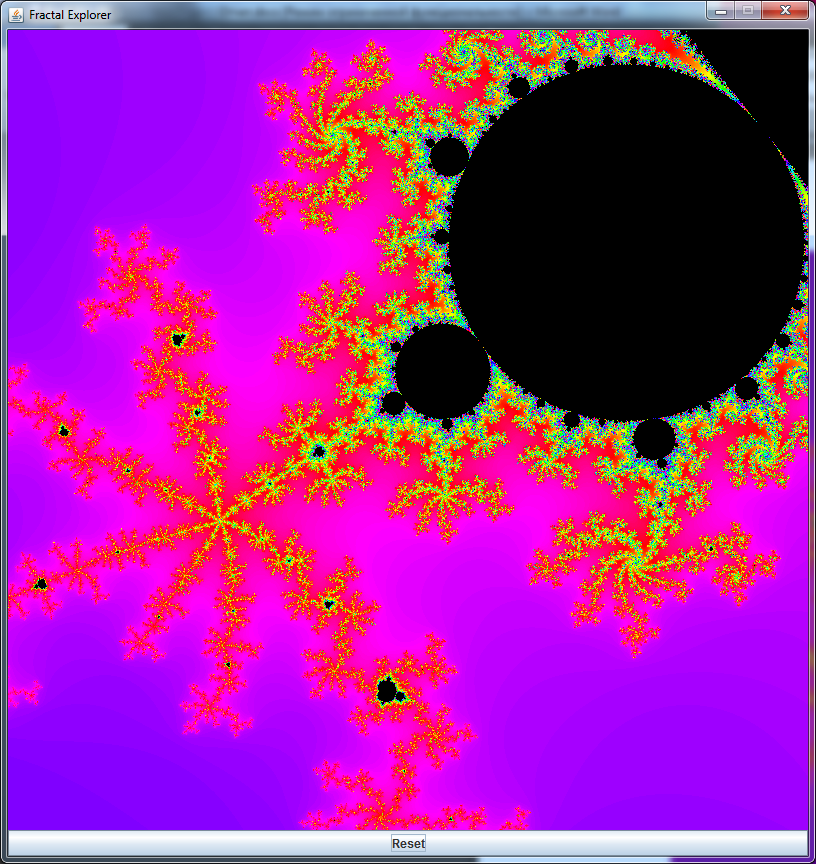


Рисунок 2 – Приближение фрактала

1. Выводы

Таким образом, были изучены базовые способы использования такого мощного принципа ООП как наследование в языке Java.