**Федеральное агентство связи**

**Ордена Трудового Красного Знамени**

**Федеральное государственное бюджетное образовательное учреждение высшего образования**

**«Московский технический университет связи и информатики»**

**Кафедра Информатики**



**Отчет по лабораторной работе №4**

по предмету «КТП»

Выполнил: студент группы БВТ1802

Ткачев Александр Владимирович

Руководитель:

Ксения Андреевна Полянцева

Москва 2020

**1 Цель работы**

Цель работы: изучить алгоритм расчета фрактала, а также познакомиться с java.swing.

**2 Задание**

Дополнить исходный текст программы таким образом, чтобы она выводила окно с фракталом и интерфейсом работы с ним.

**3 Текст программы**

**Class FractalExplorer**

package com.company;  
  
import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
  
*/\*\*  
 \* Этот класс позволяет исследовать различные части фрактала,  
 \* создавая и отображая графический интерфейс Swing, и  
 \* обрабатывает события, вызванные различными взаимодействиями с пользователем.  
 \*/*public class FractalExplorer  
{  
 */\*\* Целочисленный размер экрана является ширина и высота в пиксеях. \*\*/* private int displaySize;  
  
 */\*\*  
 \* Ссылка JImageDisplay для обновления отображения из различных методов при  
 \* вычислении фрактала.  
 \*/* private JImageDisplay display;  
  
 */\*\*  
 \* Объект FractalGnerator, использующий ссылку на базовый класс для отображения  
 \* других типов фракталов в будущем.  
 \*/* private FractalGenerator fractal;  
  
 */\*\*  
 \* Объект Rectangle2D.Double, который определяет диапазон  
 \* комплекса, который мы в данный момент отображаем.  
 \*/* private Rectangle2D.Double range;  
  
 */\*\*  
 \* Конструктор, который принимает размер экрана, сохраняет его и  
 \* инициализирует объекты диапазона и фрактального генератора.  
 \*/* public FractalExplorer(int size) {  
 */\*\* Хранит размер экрана \*\*/* displaySize = size;  
  
 */\*\* Устанавливает fractal-generator диапазон объекта. \*\*/* fractal = new Mandelbrot();  
 range = new Rectangle2D.Double();  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
  
 }  
 */\*\*  
 \* Этот метод инициализирует Swing GUI с помощью JFrame,  
 \* содержащего объект JImageDisplay и кнопку для сброса отображения.  
 \*/* public void createAndShowGUI()  
 {  
 */\*\* Установите фрейм, чтобы использовать java.awt.BorderLayout для его содержимого. \*\*/* display.setLayout(new BorderLayout());  
 JFrame myframe = new JFrame("Fractal Explorer");  
  
 */\*\*  
 \* Добавляет картинку объекта в позицию BorderLayout.CENTER  
 \*/* myframe.add(display, BorderLayout.*CENTER*);  
  
 */\*\* Создает кнопку сброса. \*/* JButton resetButton = new JButton("Reset Display");  
  
 */\*\* Экземпляр ResetHandler на кнопку сброса. \*/* ResetHandler handler = new ResetHandler();  
 resetButton.addActionListener(handler);  
  
 */\*\* Добавляет кнопку сброса в положение BorderLayout.SOUTH. \*/* myframe.add(resetButton, BorderLayout.*SOUTH*);  
  
 */\*\*  
 62/5000  
 Экземпляр MouseHandler на компоненте фрактального дисплея. \*/* MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
  
 */\*\* Ставит для операции закрытия кадра по умолчанию значение «выход». \*/* myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 */\*\*  
 \* Раскладывает содержимое фрейма, делает его видимым  
 \* и запрещает изменение размера окна.  
 \*/* myframe.pack();  
 myframe.setVisible(true);  
 myframe.setResizable(false);  
 }  
  
 */\*\*  
 \* Частный вспомогательный метод для отображения фрактала. Этот метод просматривает  
 \* каждый пиксель на экране и вычисляет количество итераций для соответствующих координат в области отображения фрактала.  
 \* для соответствующих координат в области отображения фрактала.  
 \* Если число итераций равно -1, установливает цвет пикселя в черный.  
 \* В противном случае выбирает значение на основе количества итераций.  
 \* Обновляет изображение с цветом для каждого пикселя и перерисовывет  
 \* JImageDisplay когда все пиксели нарисованы.  
 \*/* private void drawFractal()  
 {  
 */\*\* Проходит через каждый пиксель на дисплее \*/* for (int x=0; x<displaySize; x++){  
 for (int y=0; y<displaySize; y++){  
  
 */\*\*  
 \* Находит координаты xCoord и yCoord  
 \* в области отображения фрактала  
 \*/* double xCoord = fractal.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
 double yCoord = fractal.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 */\*\*  
 \* Вычисляет количество итераций в области отображени фрактала.  
 \*/* int iteration = fractal.numIterations(xCoord, yCoord);  
  
 */\*\* Если количество итераций -1 - красит пиксель в черный. \*/* if (iteration == -1){  
 display.drawPixel(x, y, 0);  
 }  
  
 else {  
 */\*\*  
 \* В противном случае выбирает значение оттенка в зависимости от количества итераций.  
 \*/* float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
 */\*\* Обновляет картинку с цветом для каждого пикселя. \*/* display.drawPixel(x, y, rgbColor);  
 }  
  
 }  
 }  
 */\*\*  
 \* Когда все пиксели нарисованы, перекрашивает JImageDisplay в соответствии  
 \* с текущим содержимим его изображения.  
 \*/* display.repaint();  
 }  
 */\*\*  
 \* Внутренний класс для обработки событий ActionListener от кнопки сброса.  
 \*/* private class ResetHandler implements ActionListener  
 {  
 */\*\*  
 \* Обработчик сбрасывает диапазон в начальный диапазон,  
 \* заданный генератором, а затем рисует фрактал.  
 \*/* public void actionPerformed(ActionEvent e)  
 {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 }  
 */\*\*  
 \* Внутренний класс для обработки событий MouseListener с дисплея.  
 \*/* private class MouseHandler extends MouseAdapter  
 {  
 */\*\*  
 \* Когда обработчик получает событие щелчка мыши, он отображает  
 \* пиксельные координаты щелчка в отображаемой области фрактала,  
 \* а затем вызывает метод recenterAndZoomRange ()  
 \* генератора с координатами, по которым щелкнули, и масштабом 0,5.  
 \*/* @Override  
 public void mouseClicked(MouseEvent e)  
 {  
 */\*\* Получить координату х области отображения щелчка мыши. \*/* int x = e.getX();  
 double xCoord = fractal.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
  
 */\*\* Получить координату y области отображения щелчка мыши. \*/* int y = e.getY();  
 double yCoord = fractal.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 */\*\*  
 \* Вызывает метод recenterAndZoomRange () генератора с координатами,  
 \* по которым щелкнули, и масштабом 0,5.  
 \*/* fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
  
 */\*\*  
 \* Перерисовывает фрактал после изменения отображаемой области.  
 \*/* drawFractal();  
 }  
 }  
  
 */\*\*  
 \* Статический метод main () для запуска FractalExplorer. Инициализирует новый экземпляр  
 \* FractalExplorer с размером дисплея 600, вызывает  
 \* createAndShowGUI() для объекта проводника, а затем вызывает  
 \* drawFractal() для проводника, чтобы увидеть начальный вид.  
 \*/* public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

**Class FractalGenerator**

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
  
*/\*\*  
 \* Этот класс предоставляет общий интерфейс и операции для генераторов фракталов,  
 \* которые можно просматривать в Fractal Explorer.  
 \*/*public abstract class FractalGenerator {  
  
 */\*\*  
 \* Эта статическая вспомогательная функция принимает целочисленную координату и преобразует  
 \* ее в значение двойной точности, соответствующее определенному диапазону.  
 \* Он используется для преобразования координат пикселей в  
 \* значения двойной точности для вычисления фракталов и т.д.  
 \*  
 \** ***@param*** *rangeMin минимальное значение диапазона с плавающей точкой  
 \** ***@param*** *rangeMax максимальное значение диапазона с плавающей точкой  
 \*  
 \** ***@param*** *size размер измерения, из координата пикселя.  
 \* Например, это может быть ширина изображения или высота изображения.  
 \*  
 \** ***@param*** *coord координата для вычисления значения двойной точности.  
 \* Координата должна попадать в диапазон [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);  
 }  
  
  
 */\*\*  
 \* Устанавливает указанный прямоугольник, чтобы содержать начальный диапазон,  
 \* подходящий для генерируемого фрактала.  
 \*/* public abstract void getInitialRange(Rectangle2D.Double range);  
  
  
 */\*\*  
 \* Обновляет текущий диапазон с центром в указанных координатах,  
 \* и увеличивает или уменьшает с помощью указанного коэффициента масштабирования.  
 \*/* 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;  
 }  
  
  
 */\*\*  
 \* Учитывая координату <em>x</em> + <em>iy</em> в комплексной плоскости,  
 \* вычисляет и возвращает количество итераций до того, как фрактальная функция покинет  
 \* ограничивающую область для этой точки. Точка, которая не  
 \* проходит до достижения предела итерации, указывается с результатом -1.  
 \*/* public abstract int numIterations(double x, double y);  
  
}

}

**Class Mandelbrot**

package com.company;  
  
import java.awt.geom.Rectangle2D;  
  
*/\*\*  
 \* Этот класс является подклассом FractalGenerator. Он используется для вычисления  
 \* вычисления фрактала Мандельброта.  
 \*/*public class Mandelbrot extends FractalGenerator  
{  
 */\*\*  
 \* Константа для числа максимальных итераций.  
 \*/* public static final int *MAX\_ITERATIONS* = 2000;  
  
 */\*\*  
 \* Этот метод позволяет генератору фракталов указать,  
 \* какая часть комплексной плоскости наиболее интересна для фрактала.  
 \* Передается объект прямоугольника, и метод изменяет поля прямоугольника,  
 \* чтобы показать правильный начальный диапазон для фрактала.  
 \* Эта реализация устанавливает начальный диапазон (-2 - 1.5i) - (1 + 1.5i)  
 \* или x=-2, y=-1.5, width=height=3.  
 \*/* public void getInitialRange(Rectangle2D.Double range)  
 {  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
  
 */\*\*  
 \* Этот метод реализует итерационную функцию для фрактала Мандельброта.  
 \* Он принимает два двойных для действительной и мнимой частей комплексной плоскости и возвращает  
 \* количество итераций для соответствующей координаты.  
 \*/* public int numIterations(double x, double y)  
 {  
 */\*\* Start with iterations at 0. \*/* int iteration = 0;  
 */\*\* Initialize zreal and zimaginary. \*/* double zreal = 0;  
 double zimaginary = 0;  
  
 */\*\*  
 \* Compute Zn = Zn-1^2 + c where values are complex numbers represented  
 \* by zreal and zimaginary, Z0=0, and c is the particular point in the  
 \* fractal that we are displaying (given by x and y). It is iterated  
 \* until Z^2 > 4 (absolute value of Z is greater than 2) or maximum  
 \* number of iterations is reached.  
 \*/* while (iteration < *MAX\_ITERATIONS* &&  
 zreal \* zreal + zimaginary \* zimaginary < 4)  
 {  
 double zrealUpdated = zreal \* zreal - zimaginary \* zimaginary + x;  
 double zimaginaryUpdated = 2 \* zreal \* zimaginary + y;  
 zreal = zrealUpdated;  
 zimaginary = zimaginaryUpdated;  
 iteration += 1;  
 }  
  
 */\*\*  
 \* If the number of maximum iterations is reached, return -1 to  
 \* indicate the point didn't escape outside of the boundary.  
 \*/* if (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
  
 return iteration;  
 }  
  
}

**Class JImageDisplay**

package com.company;  
  
import javax.swing.\*;  
import java.awt.image.\*;  
import java.awt.\*;  
  
*/\*\*  
 \* Этот класс отображает фракталы  
 \* с помощью javax.swing.JComponent.  
 \*/*class JImageDisplay extends JComponent  
{  
 */\*\*  
 \* Экземпляр буферизованного изображения.  
 \* Управляет изображением, содержимое которого мы можем записать.  
 \*/* private BufferedImage displayImage;  
  
 */\*\*  
 \* Конструктор принимает целочисленную ширину и высоту и  
 \* инициализирует свой объект BufferedImage новым  
 \* изображением с этой шириной высотой типа изображения TYPE\_INT\_RGB.  
 \*/* public JImageDisplay(int width, int height) {  
 displayImage = new BufferedImage(width, height,  
 BufferedImage.*TYPE\_INT\_RGB*);  
  
 */\*\*  
 \* Вызов метода setPreferredSize() родительского класса с заданной шириной и высотой.  
 \*/* Dimension imageDimension = new Dimension(width, height);  
 super.setPreferredSize(imageDimension);  
  
 }  
 */\*\*  
 \* Реализация суперкласса paintComponent(g).  
 \*/* @Override  
 public void paintComponent(Graphics g)  
 {  
 super.paintComponent(g);  
 g.drawImage(displayImage, 0, 0, displayImage.getWidth(),  
 displayImage.getHeight(), null);  
 }  
 */\*\*  
 \* Устанавливает все пиксели в изображении на черный.  
 \*/* public void clearImage()  
 {  
 int[] blankArray = new int[getWidth() \* getHeight()];  
 displayImage.setRGB(0, 0, getWidth(), getHeight(), blankArray, 0, 1);  
 }  
 */\*\*  
 \* Устанавливает пиксель в определенный цвет.  
 \*/* public void drawPixel(int x, int y, int rgbColor)  
 {  
 displayImage.setRGB(x, y, rgbColor);  
 }  
}

**4 Работа программы**

