**МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ**

**Ордена трудового Красного Знамени федеральное государственное бюджетное**

**образовательное учреждение высшего образования**

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

Кафедра Математическая кибернетика и информационные технологии

Отчет по лабораторной работе № 4

по дисциплине «Информационные технологии и программирование»

Выполнил: студент группы БФИ2102

Голубева Елена Александровна

Проверила:

Мосева Марина Сергеевна

Москва, 2022

Цель работы

Создание JAVA-приложения, которое сможет рисовать фракталы при помощи фреймфорка Swing, Java API.

Задание

- Создать графический виджет, который будет отображать фракталы.

• Создайте класс JImageDisplay, производный от javax.swing.JComponent.

• 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. Конструктор также должен вызвать метод setPreferredSize() родительского класса метод с указанной шириной и высотой.

• Пользовательские компоненты Swing должны предоставлять свой собственный код для отрисовки, переопределяя защищенный метод JComponent paintComponent (Graphics g). Нужно всегда вызывать метод суперкласса paintComponent (g) так, чтобы объекты отображались правильно.

• Создать два метода с доступом public для записи данных в изображение: метод clearImage (), который устанавливает все пиксели изображения в черный цвет (значение RGB 0), и метод drawPixel (int x, int y, int rgbColor), который устанавливает пиксель в определенный цвет.

• Создайте подкласс FractalGenerator с именем Mandelbrot. в нем вам необходимо будет обеспечить только два метода: getInitialRange() и numIterations().

• getInitialRange (Rectangle2D.Double) - метод позволяет генератору фракталов определить наиболее «интересную» область комплексной плоскости для конкретного фрактала.

• Метод numIterations(double, double) реализует итеративную функцию для фрактала Мандельброта.

- Создайте 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.

• 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.

• У класса должен быть конструктор, который принимает значение размера отображения в качестве аргумента, затем сохраняет это значение в соответствующем поле, а также инициализирует объекты диапазона и фрактального генератора.

• Создайте метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и кнопку для сброса отображения. Добавьте объект отображения изображения в позицию BorderLayout.CENTER и кнопку в позицию BorderLayout.SOUTH. Вам необходимо дать окну подходящий заголовок и обеспечить операцию закрытия окна по умолчанию. После того, как компоненты пользовательского интерфейса инициализированы и размещены, добавьте следующую последовательность операций: frame.pack (); frame.setVisible (true); frame.setResizable (false);

• Реализуйте вспомогательный метод с типом доступа private для вывода на экран фрактала, можете дать 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)

- Вычислите количество итераций для соответствующих координат в области отображения фрактала. Вы можете определить координаты с плавающей точкой для определенного набора координат пикселей, используя вспомогательный метод FractalGenerator.getCoord ();

- Отображение необходимо обновлять в соответствии с цветом для каждого пикселя.

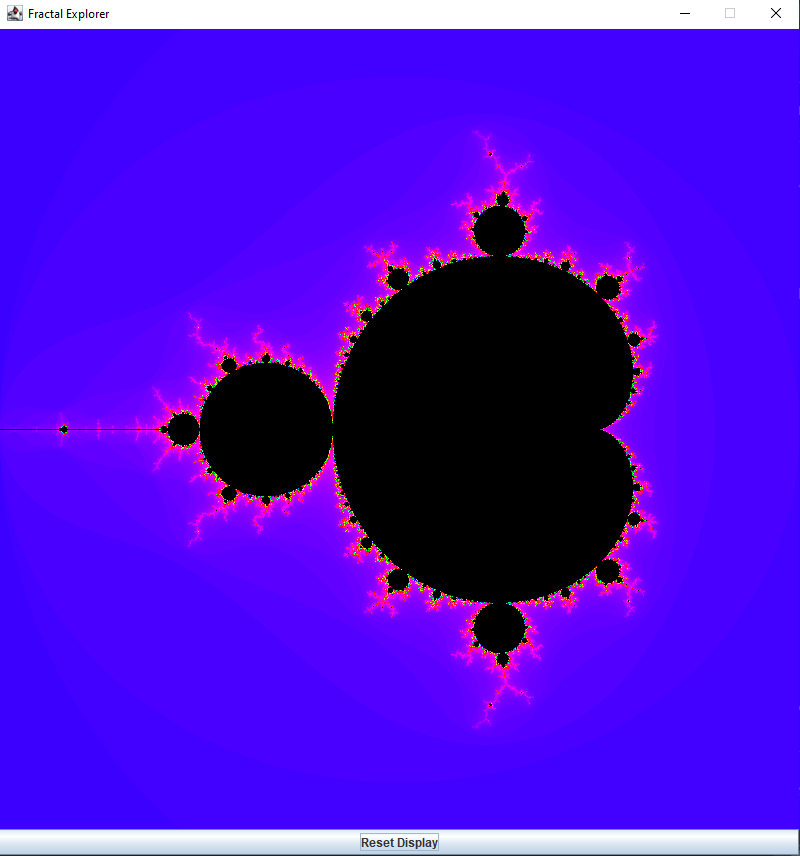
• Создайте внутренний класс для обработки событий java.awt.event.ActionListener от кнопки сброса. Обработчик должен сбросить диапазон к начальному, определенному генератором, а затем перерисовать фрактал. После того, как вы создали этот класс, обновите метод createAndShowGUI ().

• Создайте другой внутренний класс для обработки событий java.awt.event.MouseListener с дисплея. Вам необходимо обработать события от мыши, поэтому вы должны 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.

• В заключении, вам 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. ¬ Вызовите метод createAndShowGUI () класса FractalExplorer. ¬ Вызовите метод drawFractal() класса FractalExplorer для отображения начального представления.

Ход работы

1. Результат работы кода.



Код класса FractalExplorer

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
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;  
  
public class FractalExplorer { //исследование различных областей фрактала, путем его создания, отображения и обработки событий  
  
 private int displaySize; //размер экрана  
  
 private JImageDisplay display; //ссылка для обновления отображения в разных  
 // методах в процессе вычисления фрактала  
  
 private FractalGenerator fractal; //ссылка на базовый класс  
 // для отображения других видов фракталов в будущем  
  
 private Rectangle2D.Double range; //указывает диапозон комплексной плоскости, выводимой на экран  
  
 //принимает значение размера отображения в качестве  
 //аргумента, затем сохраняет это значение в соответствующем поле,  
 //а также инициализирует объекты диапазона и фрактального генератора  
 public FractalExplorer(int size) {  
  
 displaySize = size;  
  
 fractal = new Mandelbrot(); //инициализация объектов диапозона и фрактального генератора  
 range = new Rectangle2D.Double();  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
  
 }  
  
 //инициализирует и размещает графический интерфейс Swing,  
 public void createAndShowGUI()  
 {  
 //Установите фрейм для использования java.awt.BorderLayout для его содержимого  
 display.setLayout(new BorderLayout());  
 JFrame myframe = new JFrame("Fractal Explorer");  
  
 //отображение изображения в центре  
 myframe.add(display, BorderLayout.*CENTER*);  
  
 //сброс  
 JButton resetButton = new JButton("Reset Display");  
  
 //экземпляр оброботчика сброса  
 ResetHandler handler = new ResetHandler();  
 resetButton.addActionListener(handler);  
  
 //добавление кнопки сброса внизу окна  
 myframe.add(resetButton, BorderLayout.*SOUTH*);  
  
 //экземпляр оброботчика мыши  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
  
 //операция закрытия фрейма  
 myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 //разметка содержимого окна и запрет на изменение размера окна, сделают его видимым  
 myframe.pack();  
 myframe.setVisible(true);  
 myframe.setResizable(false);  
 }  
  
 //метод для вывода фрактала на экран  
 private void drawFractal()  
 { //проходит через все пиксели  
 for (int x=0; x<displaySize; x++){  
 for (int y=0; y<displaySize; y++){  
  
 //определение координат пикселя х - пиксельная координата хСoord - в пространстве 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);  
  
 ////количество итераций для соответствующих координат в области отображения фрактала  
 //  
 int iteration = fractal.numIterations(xCoord, yCoord);  
  
 if (iteration == -1){  
 display.drawPixel(x, y, 0); //установка в черный цвет  
 }  
  
 else {  
 //значение цвета, основанное на количестве итераций  
 float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
  
 //обновление дисплея цветом для каждого пикселя  
 display.drawPixel(x, y, rgbColor);  
 }  
  
 }  
 }  
 //обновление JimageDisplay в соответствии c текущим изображением  
 display.repaint();  
 }  
 //внутренний класс для обработки событий java.awt.event.ActionListener от кнопки сброса, сброс диапозона к начальному  
 private class ResetHandler implements ActionListener  
 {  
 //сбрасывает диапазон до начального диапазона, заданного генератором, а затем перерисовывает фрактал  
 public void actionPerformed(ActionEvent e)  
 {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
 }  
 //класс для обработки событий MouseListener с дисплея, события от мыши  
 private class MouseHandler extends MouseAdapter  
 {  
 //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.  
 @Override  
 public void mouseClicked(MouseEvent e)  
 {  
  
 int x = e.getX();  
 double xCoord = fractal.*getCoord*(range.x,  
 range.x + range.width, displaySize, x);  
  
  
 int y = e.getY();  
 double yCoord = fractal.*getCoord*(range.y,  
 range.y + range.height, displaySize, y);  
  
 // вызов метода генератора, по которым щелкнули, и масштабом 0.5  
 fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
  
 // перерисовка фрактала после того, как вы изменяется область фрактала  
 drawFractal();  
 }  
 }  
  
 // статический метод main() для FractalExplorer так, чтобы можно было его запустить  
 public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(800);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal(); //отображение начального представления  
 }  
}

Код класса JImageDisplay

package com.company;  
  
import javax.swing.\*;  
import java.awt.\*;  
import java.awt.image.BufferedImage; //управляет изображением, содержимое которого можно записать  
  
public class JImageDisplay extends JComponent {  
  
 private BufferedImage forImageD; //экземпляр класса  
  
 public JImageDisplay (int width, int height){ //конструктор  
 forImageD = new BufferedImage(width, height, BufferedImage.*TYPE\_INT\_RGB*);  
  
 Dimension dimensionImg = new Dimension(width, height);  
  
 super.setPreferredSize(dimensionImg);  
  
 } //компонент будет включен в пользовательский интерфейс, отобразит все изображения на экране  
  
 public void paintComponent(Graphics g){ //защищенный метод, выводит на экран изображения  
 super.paintComponent(g); //правильное отображение  
 g.drawImage (forImageD, 0, 0, forImageD.getWidth(), forImageD.getHeight(), null); //рисование изображения в компоненте  
  
 }  
  
 public void clearImage(){ //устанавливает все пиксели в черный цвет  
 int[] sizeImageArray = new int[getWidth()\*getHeight()];  
  
 forImageD.setRGB(0, 0, getWidth(), getHeight(), sizeImageArray, 0, 1);  
  
 }  
  
 public void drawPixel(int x, int y, int rgbColor){ //устанавливает пиксель в опрееленный цвет  
  
 forImageD.setRGB(x, y, rgbColor);  
 }  
}

Код класса Mandelbrot

package com.company;  
import java.awt.geom.Rectangle2D;  
  
public class Mandelbrot extends FractalGenerator{  
 public static final int *MAX\_ITERATIONS* = 2000;  
  
 @Override  
 public void getInitialRange(Rectangle2D.Double range) { //определяет область комплексной плоскости для фрактала  
  
 range.x = -2;  
 range.y = -1.5;  
 range.width = 3;  
 range.height = 3;  
 }  
 @Override  
 public int numIterations(double x, double y) { //итеративная функция для фрактала  
 int iteration = 0;  
  
 double zreal = 0;  
 double zimaginary = 0;  
  
 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 (iteration == *MAX\_ITERATIONS*)  
 {  
 return -1;  
 }  
 return iteration;  
  
 }  
}

Вывод

В процессе выполнения данной лабораторной работы мы cоздали JAVA-приложения, которое сможет рисовать фракталы при помощи фреймфорка Swing, Java API..

Ссылка на Github: https://github.com/ElenaGolubeva/Laboratory-4-java

Список использованной литературы

1. Камаев В.А., Костерин В.В. Технологии программирования. М.: Высшая школа,

2006.

1. Жоголев Е.А.Технология программирования. – М.: Научный мир, 2004.