**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**

Факультет РиТ

**ЛАБОРАТОРНАЯ РАБОТА №4**

по дисциплине «Java-программирование»

на тему:

«Рисование фракталов»

Выполнил: студ. гр. БПЗ1901

Неживлева Ксения

Проверил: Харрасов Камиль Раисович

Москва 2021

Цель работы:

В следующих нескольких лабораторных работ вы создадите небольшое JAVA-приложение, которое сможет рисовать фракталы. Если вы никогда не играли с фракталами раньше, вы будете удивлены тем, как просто можно создать красивые изображения. Это будет сделано с помощью фреймворка Swing и Java API, который позволяет создавать графические пользовательские интерфейсы.

Выполнение:

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. Как видно их приведенных выше изображений пользовательского интерфейса, FractalExplorer очень прост, он состоит из JFrame, который в свою очередь содержит объект JImageDisplay, который отображает фрактал,и объект JButton для сброса изображения, необходимый для отображения целого фрактала. Данный макет можно создать, установив для фрейма BorderLayout, затем поместив отображение в центр макета и кнопку сброса в "южной" части макета.

• 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.

import java.awt.\*;  
import javax.swing.\*;  
import java.awt.geom.Rectangle2D;  
import java.awt.event.\*;  
  
*/\*\* Этот класс позволяет исследовать различные части фрактала, создавая и показывая графический интерфейс Swing,  
 \* и обрабатывает события, вызванные различными взаимодействиями пользователя. \*/*public class FractalExplorer  
{  
 private int displaySize; // Целое число «размер экрана», которое является шириной и высотой отображения в пикселях.  
 private JImageDisplay display; // Ссылка JImageDisplay, для обновления отображения в разных методах в процессе вычисления фрактала.  
 private FractalGenerator fractal; // Объект FractalGenerator, использующий ссылку базового класса для отображения других типов фракталов в будущем.  
 private Rectangle2D.Double range; // Объект Rectangle2D.Double, указывающий диапазона комплексной плоскости, которая выводится на экран.

Рис.1-класс FractalExplorer

• 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. Данный конструктор не должен устанавливать какиелибо компоненты Swing; они будут установлены в следующем методе.

// Конструктор, который принимает размер отображения, сохраняет его и инициализирует объекты диапазона и генератора фракталов.  
 public FractalExplorer(int size) {  
 displaySize = size;  
 fractal = new Mandelbrot();  
 range = new Rectangle2D.Double();  
 fractal.getInitialRange(range);  
 display = new JImageDisplay(displaySize, displaySize);  
  
 }

Рис.2-Констркутор класса FractalExplorer

• Создайте метод createAndShowGUI (), который инициализирует графический интерфейс Swing: JFrame, содержащий объект JimageDisplay, и кнопку для сброса отображения. Используйте java.awt.BorderLayout для содержимого окна; 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 (см. метод JFrame.setDefaultCloseOperation ()). После того, как компоненты пользовательского интерфейса инициализированы и размещены, добавьте следующую последовательность операций:

frame.pack ();

frame.setVisible (true);

frame.setResizable (false);

Данные операции правильно разметят содержимое окна, сделают его видимым (окна первоначально не отображаются при их создании для того, чтобы можно было сконфигурировать их прежде, чем выводить на экран), и затем запретят изменение размеров окна.

// Этот метод инициализирует графический интерфейс Swing с помощью JFrame, содержащего объект JImageDisplay и кнопку для сброса отображения.  
public void createAndShowGUI()  
{  
 // Настройка фрейма на использование java.awt.BorderLayout для его содержимого.  
 display.setLayout(new BorderLayout());  
 JFrame myframe = new JFrame("Fractal Explorer");  
 // Добавление объекта отображения изображения в позицию BorderLayout.CENTER.  
 myframe.add(display, BorderLayout.*CENTER*);  
 // Создание кнопки сброса.  
 JButton resetButton = new JButton("Reset Display");  
 // Создание объекта ResetHandler на кнопке сброса.  
 ResetHandler handler = new ResetHandler();  
 resetButton.addActionListener(handler);  
 // Добавление кнопки сброса в позицию BorderLayout.SOUTH.  
 myframe.add(resetButton, BorderLayout.*SOUTH*);  
 // Создание объекта MouseHandler для компонента фрактального отображения.  
 MouseHandler click = new MouseHandler();  
 display.addMouseListener(click);  
 // Установка операции закрытия фрейма по умолчанию на «выход».  
 myframe.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
 // Размещение содержимого фрейма, чтобы оно было видимым и запрещение изменения размера окна.  
 myframe.pack();  
 myframe.setVisible(true);  
 myframe.setResizable(false);  
}

Рис.3-Инициализация графического итерфейса

• 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), и сделайте следующее:

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 (); например, чтобы получить координату x, соответствующую координате пикселя X, сделайте следующее:

//x - пиксельная координата;

xCoord - координата в пространстве фрактала double xCoord = FractalGenerator.getCoord (range.x, range.x + range.width, displaySize, x);

Если число итераций равно -1 (т.е. точка не выходит за границы, установите пиксель в черный цвет (для rgb значение 0). Иначе выберите значение цвета, основанное на количестве итераций. Можно также для этого использовать цветовое пространство HSV: поскольку значение цвета варьируется от 0 до 1, получается плавная последовательность цветов от красного к желтому, зеленому, синему, фиолетовому и затем обратно к красному! Для этого вы можете использовать следующий фрагмент:

float hue = 0.7f + (float) numIters / 200f;

int rgbColor = Color.HSBtoRGB(hue, 1f, 1f);

Если вы придумали другой способ отображения пикселей в зависимости от количества итераций, попробуйте реализовать его!

Отображение необходимо обновлять в соответствии с цветом для каждого пикселя.

*/\*\* Приватный вспомогательный метод для отображения фрактала. Этот метод просматривает каждый пиксель на дисплее  
 \* и вычисляет количество итераций для соответствующих координат в области отображения фрактала.  
 \* Если количество итераций равно -1, цвет пикселя становится черным.  
 \* В противном случае выбираем значение на основе количества итераций, обновляем дисплей с использованием  
 \* цвета для каждого пикселя и перерисовываем JImageDisplay, когда все пиксели были нарисованы. \*/*private void drawFractal()  
{  
 // Циклический просмотр каждого пикселя на дисплее  
 for (int x=0; x<displaySize; x++)  
 for (int y=0; y<displaySize; y++){  
 // Нахождение соответствующих координат 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);  
 // Вычисление количества итераций для координат в области отображения фрактала.  
 int iteration = fractal.numIterations(xCoord, yCoord);  
 if (iteration == -1) // Если количество итераций равно -1, устанавливаем пиксель черным.  
 display.drawPixel(x, y, 0);  
 else {  
 // В противном случае выбираем значение оттенка в зависимости от количества итераций.  
 float hue = 0.7f + (float) iteration / 200f;  
 int rgbColor = Color.*HSBtoRGB*(hue, 1f, 1f);  
 display.drawPixel(x, y, rgbColor); // Обновление дисплея для каждого пикселя.  
 }  
 }

Рис.4-Вспомогательный метод для вывода фрактала на экран drawFractal

После того, как вы закончили отрисовывать все пиксели, вам необходимо обновить JimageDisplay в соответствии с текущим изображением. Для этого вызовите функцию repaint() для компонента. В случае, если вы не воспользуетесь данным методом, изображение на экране не будет обновляться!

// Когда все пиксели нарисованы, перерисовываем JImageDisplay в соответствии с текущим содержимым его изображения.  
 display.repaint();  
}

Рис.5- Обновление JimageDisplay

• 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 ().

private class ResetHandler implements ActionListener // Внутренний класс для обработки событий ActionListener от кнопки сброса.  
{  
 // Обработчик сбрасывает диапазон до начального диапазона, заданного генератором, а затем рисует фрактал.  
 public void actionPerformed(ActionEvent e)  
 {  
 fractal.getInitialRange(range);  
 drawFractal();  
 }  
}

Рис.6 - Внутренний класс для обработки событий ActionListener от кнопки сброса.

• 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. Таким образом, нажимая на какое-либо место на фрактальном отображении, вы увеличиваете его! Не забывайте перерисовывать фрактал после того, как вы меняете область фрактала. Далее обновите метод createAndShowGUI (), чтобы зарегистрировать экземпляр этого обработчика в компоненте фрактального отображения.

private class MouseHandler extends MouseAdapter // Внутренний класс для обработки событий MouseListener с дисплея.  
{  
 */\*\* Когда обработчик получает событие щелчка мыши, он сопоставляет пиксельные координаты щелчка  
 \* с областью отображаемого фрактала, а затем вызывает метод генератора correnterAndZoomRange() с координатами,  
 \* по которым щелкнули, и шкалой 0,5. \*/* @Override  
 public void mouseClicked(MouseEvent e)  
 {  
 // Получение координаты x области щелчка мыши.  
 int x = e.getX();  
 double xCoord = fractal.*getCoord*(range.x,range.x + range.width, displaySize, x);  
 // Получение координаты y области щелчка мыши.  
 int y = e.getY();  
 double yCoord = fractal.*getCoord*(range.y,range.y + range.height, displaySize, y);  
 // Вызов метода генератора RecenterAndZoomRange () с координатами, по которым был выполнен щелчок, и масштабом 0,5.  
 fractal.recenterAndZoomRange(range, xCoord, yCoord, 0.5);  
 drawFractal(); // Перерисовка фрактала после изменения отображаемой области.  
 }  
}

Рис.7- Внутренний класс для обработки событий MouseListener с дисплея

• 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.

*/\*\* Статический метод main() для запуска FractalExplorer.  
 \* Инициализирует новый экземпляр FractalExplorer с размером отображения 600,  
 \* вызывает createAndShowGUI() в объекте проводника,  
 \* а затем вызывает drawFractal() в проводнике. \*/* public static void main(String[] args)  
 {  
 FractalExplorer displayExplorer = new FractalExplorer(600);  
 displayExplorer.createAndShowGUI();  
 displayExplorer.drawFractal();  
 }  
}

Рис.8- Статический метод main() для запуска FractalExplorer.

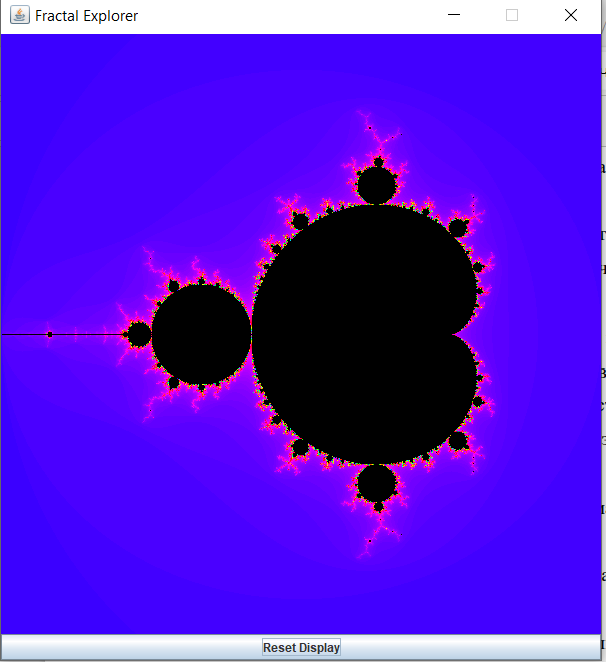


Рис.9 – Результат работы программы