Ордена Трудового Красного Знамени

федеральное государственное бюджетное

образовательное учреждение высшего

образования «Московский технический

университет связи и информатики»

Кафедра информатики

Лабораторная работа № 2 «I'm A Star!»

Выполнил:

Студент 2 курса

Факультета «ОТФ-2»

Группы БСТ-1602

Марченко Дмитрий

Вариант № 11

Москва, 2018

**Цель работы:** Для следующих нескольких задач вы соедините небольшое JAVA- приложение, которое может нарисовать некоторые удивительные фракталы. Если вы никогда не играли с фракталами прежде, вы будете поражены тем, как просто можно создать некоторые захватывающие дух красивые изображения. Мы сделаем это все с платформами Swing, API Java, которые позволяют вам создавать графические интерфейсы пользователя.

**Задание:**

1. Создайте класс JImageDisplay, который возьмём из javax.swing. JComponent. У класса должны быть одно частное поле, пример java.awt.image. BufferedImage. Класс BufferedImage управляет изображением, в содержание которого мы можем записать информацию. Конструктор JImageDisplay должен взять целочисленную ширину и высоту и инициализировать ее члена BufferedImage, чтобы быть новым изображением той ширины и высоты, и типом изображения TYPE\_INT\_RGB.
2. Пользовательские компоненты Swing должны обеспечить свой собственный код для прорисовки, переопределив защищенный paintComponent (графика g) - метод JComponent
3. Вы также должны обеспечить два открытых метода для записи данных в изображение: clearImage () - метод, который устанавливает все пиксели в данных изображения к черному цвету
4. Затем вы запишете код, чтобы вычислить очень известный фрактал Мальдерброта. Чтобы поддерживать многократные фракталы в будущем, вам предоставляют исходный файл FractalGenerator.java,
5. Создайте подкласс FractalGenerator по имени Mandelbrot. Вы будете видеть, что есть только два метода, которые вы должны обеспечить в подклассе, getInitialRange () и numIterations ()
6. getInitialRange (Rectangle2D.Double) метод позволяет фрактальному генератору определять, какая часть комплексной плоскости является самой "интересной" для определенного фрактала.
7. numIterations - метод реализует итеративную функцию для фрактала Мальдерброта. Вы можете определить константу для "максимальных итераций"
8. Наконец мы готовы начать отображать фракталы! Теперь вы создадите класс FractalExplorer, который позволяет вам исследовать различные части фрактала, создавая и показывая GUI Swing и обрабатывая события, вызванные различным взаимодействием с пользователем. Ваш класс FractalExplorer должен будет отслеживать несколько важных полей для состояния программы
9. Обеспечьте createAndShowGUI () метод, который инициализирует GUI Swing: JFrame, содержащий JImageDisplay, возращает и кнопку для сброса дисплея. Вы должны установить фрейм, использующий java.awt. BorderLayout для его содержания; добавьте объект дисплея изображения в BorderLayout.
10. Создайте внутренний класс, чтобы обработать java.awt.event. События ActionListener от кнопки сброса
11. Создайте другой внутренний класс, чтобы обработать java.awt.event. События MouseListener от дисплея.

**Анализ предметной области и инструментарий:** для разработки программы обработки информации, поступающей от пользователя в циклах и ее вывода взят пакетjava development kit-9.0.4 с классом Scanner для организации ввода и стандартный редактор тхт.

**Код программы:**

**Описание класса** JImageDisplay:

|  |
| --- |
|  |
|  | import java.awt.image.BufferedImage; |
|  | import java.awt.Graphics; |
|  | import java.awt.Dimension; |
|  | import java.swing.JComponent; |
|  | public class JImageDisplay extends JComponent |
|  | { |
|  | private BufferedImage MyImage; //хранимое изображения |
|  | public JImageDisplay(int width, int height) //конструктор инициализации |
|  | { |
|  | MyImage = new BufferedImage(width,height,BufferedImage.TYPE\_INT\_RGB); |
|  | Dimension DM= new Dimension(width,height); //созание размерности |
|  | super.setPreferredSize(DM); //задание размерности |
|  | } |
|  | public void paintComponent(Graphics g)//вставка изображения в граф.компонент |
|  | { |
|  | super.paintComponent(g); |
|  | g.drawImage(MyImage,0,0,MyImage.getWidth(),MyImage.getHeight(),null); |
|  | } |
|  | public void clearImage()//установка всех пикселей ихображения в 0 |
|  | { |
|  | for(int i=0;i<MyImage.getWidth();i++) |
|  | { |
|  | for(int j=0;j<MyImage.getHeight();j++) |
|  | { |
|  | drawPixel(i,j,0); |
|  | } |
|  | } |
|  | } |
|  | public void drawPixel(int x,int y, int rgbColor)//уст. цвета пикс. На коорд. |
|  | { |
|  | MyImage.setRGB(x,y,rgbColor); |
|  | } |
|  | } |

**Описание класса** **Mandelbrot**:

|  |
| --- |
| import java.awt.geom.Rectangle2D; |
|  | public class Mandelbrot extends FractalGenerator |
|  | { |
|  | public static int MAX\_ITERATIONS =2000;//макс. число итераций |
|  | public void getInitialRange(Rectangle2D.Double range) |
|  | {//установка начального «комплексного значения» и границ |
|  | range.x = -2; |
|  | range.y = -1.5; |
|  | range.width = 3; |
|  | range.height = 3; |
|  | } |
|  | public int numIterations(double x,double y) |
|  | { |
|  | int iter=0; //номер итерации |
|  | double re=0; //реальная составляющая |
|  | double im=0; //мнимая |
|  | double moduleZ; //модуль комплексного |
|  | while(iter<MAX\_ITERATIONS) //пока число итер<макс. |
|  | { |
|  | double nextRe=re\*re-im\*im+x;  //подсчет мнимой и реальной частей |
|  | double nextIm=2\*re\*im+y; |
|  | moduleZ=nextRe\*nextRe+nextIm\*nextIm;  //подсчет модуля получившегося комплексного |
|  | re=nextRe;  //установка новых значений |
|  | im=nextIm; |
|  | iter++; //номер итерации + 1 |
|  | if (moduleZ>4) //если модуль>4 |
|  | return iter; //вернуть номер итер. |
|  | } |
|  | return -1; //точка не вышла за границу |
|  | } |
|  | } |

**Описание класса** **FractalExplorer:**

|  |
| --- |
|  |
|  | import java.awt.BorderLayout; |
|  | import java.awt.\*; |
|  | import java.awt.event.\*; |
|  | import javax.swing.\*; |
|  | public class FractalExplorer |
|  | { |
|  | public static void main(String[] args) |
|  | { |
|  | FractalExplorer MyExplorer = new FractalExplorer(500); //иниц. изобр в 500 пикс. |
|  | MyExplorer.createAndShowGUI(); //вызов метода интерфейса |
|  | MyExplorer.drawFractal(); //расчет фрактала |
|  | } |
|  | private int MyDispalySize; |
|  | private int MySize; |
|  | private JImageDisplay MyImage; |
|  | private FractalGenerator MyGenerator; |
|  | private Rectangle2D.Double MyRange= new Rectangle2D.Double(0,0,0,0); |
|  | FractalExplorer(int NewSize) |
|  | { |
|  | MySize=NewSize;//задание размерности изображения |
|  | MyImage = new JImageDisplay(NewSize,NewSize); // |
|  | MyGenerator= new Mandelbrot(); |
|  | MyGenerator.getInitialRange(MyRange); |
|  | } |
|  | public void createAndShowGUI() |
|  | { |
|  | JFrame MyFrame =new JFrame(); |
|  | MyFrame.setLayout(new BorderLayout()); |
|  | Button B=new Button("Reset"); |
|  | MyFrame.add(B, BorderLayout.SOUTH); |
|  | MyFrame.add(MyImage, BorderLayout.CENTER); |
|  | MyFrame.setDefaultCloseOperation(MyFrame.EXIT\_ON\_CLOSE); |
|  | MyFrame.pack(); |
|  | MyFrame.setVisible(true); |
|  | MyFrame.setResizable(false); |
|  | MouseMyAdapter handler=new MouseMyAdapter(); |
|  | MyEventListener handler2=new MyEventListener(); |
|  | MyImage.addMouseListener(handler); |
|  | B.addActionListener(handler2); |
|  | } |
|  | public void drawFractal() //установка цвета каждого пикселя изображения |
|  | { |
|  | for(int x=0;x<MySize;x++) |
|  | { |
|  | for(int y=0;y<MySize;y++) |
|  | {  Поиск пискля для расчета при помощи мтолов в FractalGenerator |
|  | double X = FractalGenerator.getCoord(MyRange.x,MyRange.x+MyRange.width,MySize,x); |
|  | double Y = FractalGenerator.getCoord(MyRange.y,MyRange.y+MyRange.height,MySize,y); |
|  | int iteration = MyGenerator.numIterations(X,Y); |
|  | int ColoR=0; |
|  | if(iteration!=-1) //если пиксель не за пределами |
|  | { |
|  | float hue = 0.7f + (float)iteration/200f; //установка цвета согласно номеру последней итерации |
|  | ColoR = Color.HSBtoRGB(hue,1f,1f); |
|  | } |
|  | MyImage.drawPixel(x,y,ColoR); //покраска пикселя |
|  | } |
|  | } |
|  | MyImage.repaint(); //обновление изображения |
|  | } |
|  | public class MouseMyAdapter extends MouseAdapter //подкласс для получения //действий пользователя при помощи мыши |
|  | { |
|  | public void mouseClicked(MouseEvent e) |
|  | {  // приближение в точку на 20% при нажатии на изображение |
|  | MyGenerator.recenterAndZoomRange(MyRange,FractalGenerator.getCoord(MyRange.x,MyRange.x+MyRange.width,MySize,e.getX()),FractalGenerator.getCoord(MyRange.y,MyRange.y+MyRange.height,MySize,e.getY()),0.2); |
|  | drawFractal(); |
|  | } |
|  | } |
|  | public class MyEventListener implements ActionListener |
|  | { |
|  | public void actionPerformed(ActionEvent e) |
|  | { |
|  | MyGenerator.getInitialRange(MyRange); //установка изначального масштаба //при нажатии на кнопку |
|  | drawFractal(); |
|  | } |
|  | } |
|  | } |

**Классы не подвергшиеся изменениям:**

FractalGenerator.java – обеспечивает обработку координат а также приближение.

**Выводы:** построены алгоритмы работы с объектами swing, обработки изображения, работы с интерфейсами и устройствами.