**Федеральное агентство связи**

Ордена Трудового Красного Знамени

федеральное государственное бюджетное образовательное учреждение высшего образования   
«Московский технический университет связи и информатики»

Кафедра «Информатика»

Лабораторная работа №4: Рисование фракталов

по дисциплине «Кроссплатформенные технологии программирования»

Создать красивые изображения. Это будет сделано с помощью фреймворка Swing и Java API, который позволяет создавать графические пользовательские интерфейсы.

Выполнил: студент

группы БСТ1803

Корнеев Александр

Вариант 12

Москва 2020

**Содержание**

[1 Задание на разработку программы 3](#_Toc41583959)

[2 Разработанные функции и классы 8](#_Toc41583960)

[2.1 Класс FractalExplorer 8](#_Toc41583961)

[2.1.1 Метод void createAndShowGUI 8](#_Toc41583962)

[2.1.2 Метод void drawFractal 8](#_Toc41583963)

[2.1.3 Подкласс ButtonEventListener с методом actionPerformed 8](#_Toc41583964)

[2.1.4 Подкласс MouseEventListener с методом mouseClicked 8](#_Toc41583965)

[2.1.5 Метод main 8](#_Toc41583966)

[2.2. Класс JimageDisplay 9](#_Toc41583967)

[2.2.1 Метод void paintComponent 9](#_Toc41583968)

[2.2.2 Метод void clearImage 9](#_Toc41583969)

[2.2.3 Метод void drawPixel 9](#_Toc41583970)

[2.3 Класс Mandelbrot 9](#_Toc41583971)

[2.3.1 void getInitialRange 9](#_Toc41583972)

[2.3.2 int numIterations 9](#_Toc41583973)

[2.4 Класс Compl 10](#_Toc41583974)

[2.4.1 Метод double getX 10](#_Toc41583975)

[2.4.2 Метод double getY 10](#_Toc41583976)

[2.4.3 Метод void squaring 10](#_Toc41583977)

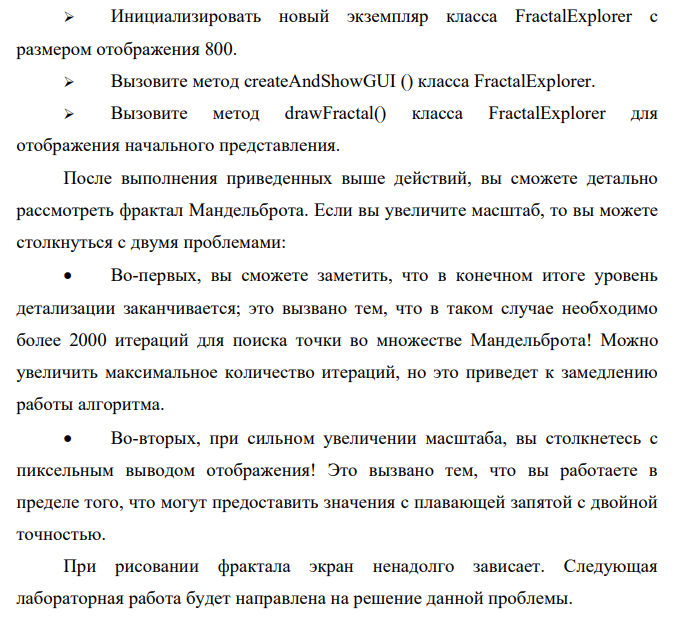
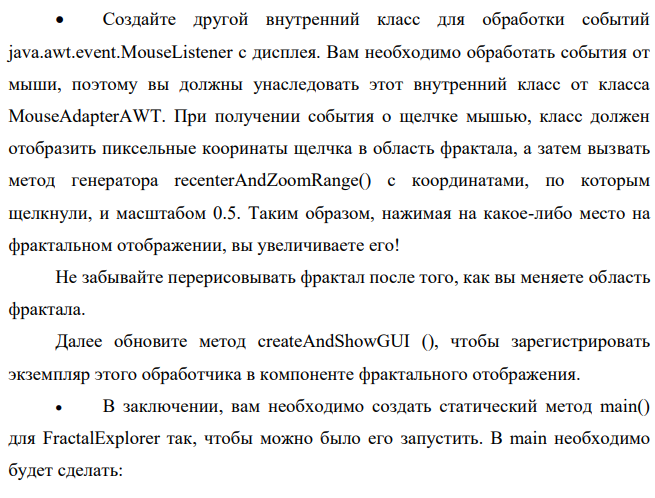
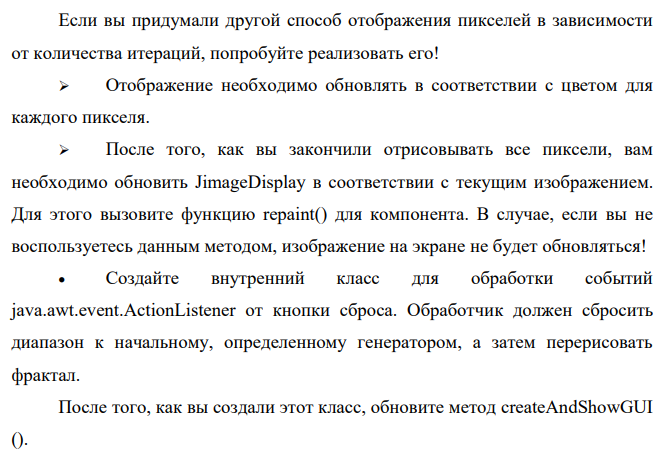
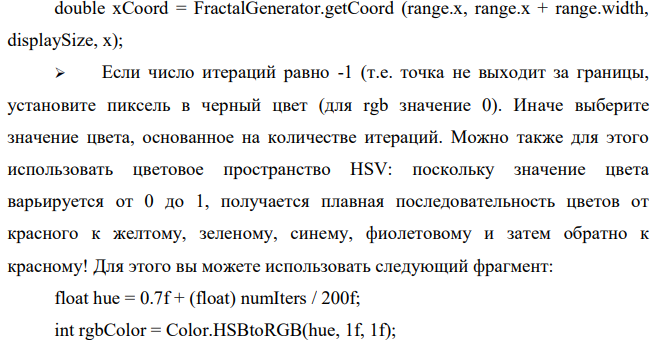
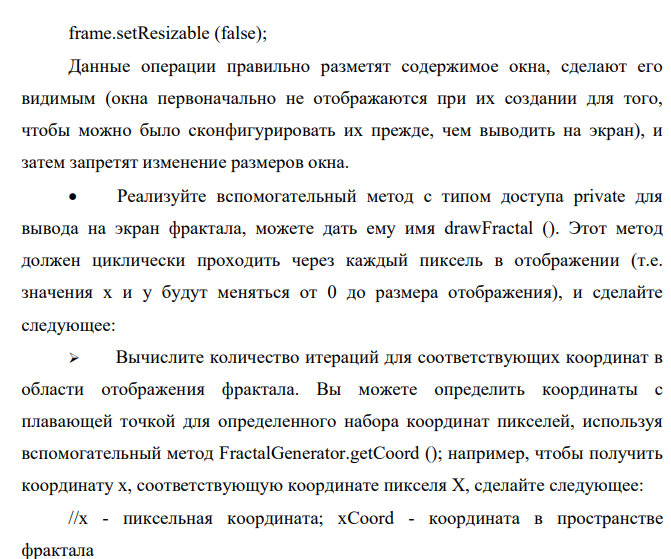
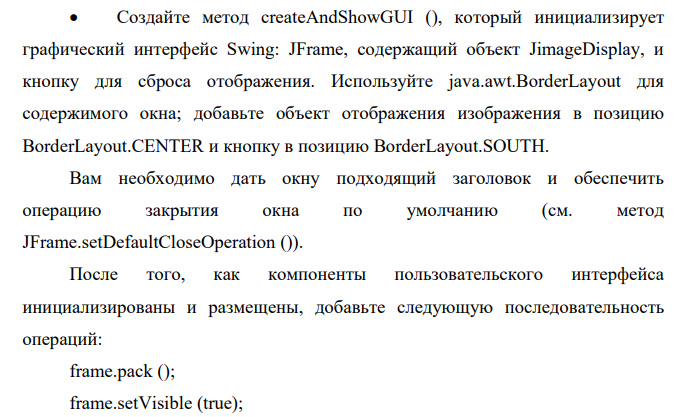
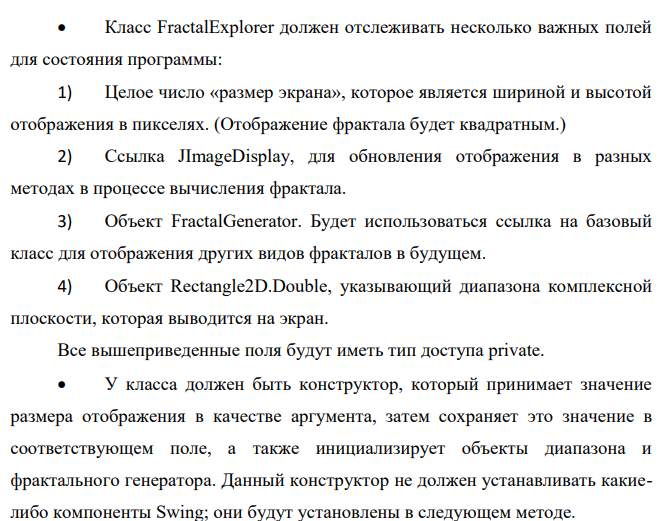
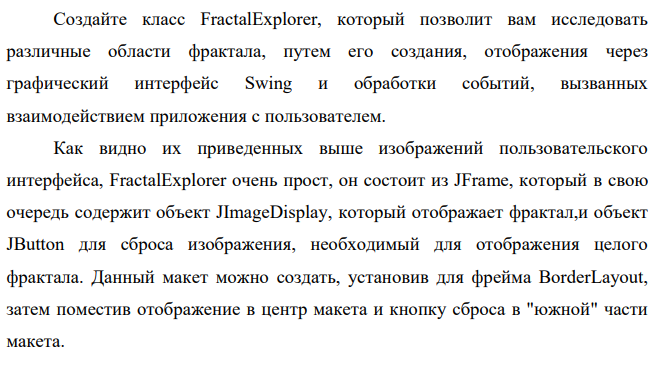
[2.4.4 Метод double squarModul 10](#_Toc41583978)

[2.4.5 Метод void addCompl 10](#_Toc41583979)

[3 Код программы 10](#_Toc41583980)

[4 Результат работы программы 16](#_Toc41583981)

# Задание на разработку программы



# Разработанные функции и классы

## Класс FractalExplorer

Класс отрисовывающий графический интерфейс Swing.

* + 1. Метод void createAndShowGUI

Установка компонентов Swing, создание и показ формы

* + 1. Метод void drawFractal

Метод вывода фрактала на экран

* + 1. Подкласс ButtonEventListener с методом actionPerformed

Обработчик нажатия на кнопку

* + 1. Подкласс MouseEventListener с методом mouseClicked

Обработчик нажатия на изображение

* + 1. Метод main

Создает объект формы, создает на ней и отрисовывает объекты Swing и фрактал

## 2.2. Класс JimageDisplay

Класс для управления изображением на форме Swing

* + 1. Метод void paintComponent

Перерисовывает изображение на экране

* + 1. Метод void clearImage

Метод, очищающий изображение, устанавливая все пиксели в черный цвет

* + 1. Метод void drawPixel

Метод устанавливающий пиксель в определённый цвет

## Класс Mandelbrot

Реализует методы, необходимые для генерации фрактала Мандельброта

* + 1. void getInitialRange

Метод определяющий область комплексной плоскости для фрактала Мандельброта

* + 1. int numIterations

Метод, вычисляющий количество итераций для отрисовки точки

## Класс Compl

Класс описывающий комплексную переменную

* + 1. Метод double getX

Метод получения действительной части

* + 1. Метод double getY

Метод получения мнимой части

* + 1. Метод void squaring

Метод возведения комплексного числа в квадрат

* + 1. Метод double squarModul

Метод вычисления квадратного модуля комплексного числа

* + 1. Метод void addCompl

Метод сложения комплексных чисел

# Код программы

/\*\*

\* Класс описывающий коплексную переменную

\*/

**public** **class** Compl

{

/\*\*

\* Переменная, описывающая действительную часть

\*/

**private** **double** x;

/\*\*

\* Переменная, описывающая мнимую часть

\*/

**private** **double** iy;

/\*\*

\* Конструктор без входных параметров

\*/

Compl()

{

**this**.x = 0;

**this**.iy = 0;

}

/\*\*

\* Конструктор создающий комплексную точку в пространстве

\* **@param** x - действительная часть

\* **@param** y - мнимая часть

\*/

Compl(**double** x, **double** y)

{

**this**.x = x;

**this**.iy = y;

}

/\*\*

\* Метод получения действительной части

\*/

**public** **double** getX()

{

**return** **this**.x;

}

/\*\*

\* Метод получения мнимой части

\*/

**public** **double** getY()

{

**return** **this**.iy;

}

/\*\*

\* Метод возведения комплексного числа в квадрат

\*/

**public** **void** squaring()

{

**double** newx = **this**.x\***this**.x-**this**.iy\***this**.iy;

**this**.iy = 2\***this**.x\***this**.iy;

**this**.x = newx;

}

/\*\*

\* Метод вычисления квадратного модуля комплексного числа

\*/

**public** **double** squarModul()

{

**return** **this**.x\***this**.x +**this**.iy\***this**.iy;

}

/\*\*

\* Метод сложения комплексных чисел

\* **@param** add - второй слагаемое

\*/

**public** **void** addComlp(Compl add)

{

**this**.x += add.getX();

**this**.iy += add.getY();

}

}

**import** java.awt.geom.Rectangle2D.Double;

/\*\*

\* Класс реализующий методы необходимые для генерации

\* фрактала Мандельброта

\*/

**public** **class** Mandelbrot **extends** FractalGenerator {

/\*\* Константа определяющая максимальное количество

\* итераций при вычислении фрактала

\*/

**public** **static** **final** **int** ***MAX\_ITERATIONS*** = 2000;

**public** Mandelbrot() {

// **TODO** Auto-generated constructor stub

}

/\*\* Метод, определяющий область комплексной

\* плоскости для фрактала Мандельброта

\*/

@Override

**public** **void** getInitialRange(Double range) {

range.x = -2;

range.y = -1.5;

range.width = 3;

range.height = 3;

}

/\*\*

\* Метод, вычисляющий количество итераций для отрисовки точки

\*/

@Override

**public** **int** numIterations(**double** x, **double** y) {

// Счетчик итераций

**int** iter = 0;

// Комлпексная переменная c

Compl c = **new** Compl(x,y);

// Комплексная переменная изменяемая, на каждой итераци

Compl z = **new** Compl();

**while** (**true**)

{

z.squaring();

z.addComlp(c);

iter++;

**if** (z.squarModul()>4) **return** iter;

**if** (iter>***MAX\_ITERATIONS***) **return** -1;

}

}

}

/\*\*

\* Класс для управления изображением на форме Swing

\*/

**public** **class** JImageDisplay **extends** JComponent {

/\*\* Экземпляр управляемого изображения \*\*/

**private** BufferedImage bImage;

/\*\* Конструктор, принимает значения

\* int width - ширина изображения

\* int height - высота изображения

\*/

**public** JImageDisplay(**int** width, **int** height) {

**this**.bImage = **new** BufferedImage(width,height,1);

**super**.setPreferredSize(**new** java.awt.Dimension(width,height));

}

/\*\* Переопределение родительского метода,

\* Выводит на экран данные изображения

\*/

@Override

**public** **void** paintComponent(Graphics g)

{

**super**.paintComponents(g);

g.drawImage(**this**.bImage, 0, 0, **this**.bImage.getWidth(), **this**.bImage.getHeight(), **null**);

}

/\*\* Метод, очищающий изображение

\* Устанавливая все пиксели в черный цвет

\*/

**public** **void** clearImage()

{

**for**(**int** x = 0;x<**this**.bImage.getWidth();x++)

{

**for** (**int** y = 0;y<**this**.bImage.getHeight();y++)

{

**this**.bImage.setRGB(x, y, 0);

}

}

}

/\*\* Метод устанавливающий пиксель в определенный цвет\*\*/

**public** **void** drawPixel(**int** x, **int** y, **int** rgbColor)

{

**this**.bImage.setRGB(x, y, rgbColor);

}

}

/\*\*

\* Класс отрисовывающий графический интерфейс Swing

\*/

**public** **class** FractalExplorer **extends** JFrame {

/\*\*

\* Ширина и высота создаваемого окна

\*/

**private** **int** size;

/\*\*

\* Элемент управляющий изображением на форме

\*/

**private** JImageDisplay image;

/\*\*

\* Ссылка на генератор фракталов

\*/

**private** FractalGenerator fg;

/\*\*

\* Диапазон комплексной плоскости, выводимой на экран

\*/

**private** Rectangle2D.Double range;

/\*\* Конструктор, входные параметры:

\*

\* **@param** width - ширина создаваемого окна

\* **@param** height - длина создаваемого окна

\*/

**public** FractalExplorer(**int** size)

{

**this**.size = size;

image = **new** JImageDisplay(size, size);

fg = **new** Mandelbrot();

range = **new** Rectangle2D.Double();

fg.getInitialRange(range);

}

/\*\*

\* Установка компонентов Swing, создание и показ формы

\*/

**private** **void** createAndShowGUI(){

**this**.setTitle("Fractal Explorer");

**this**.setSize(size, size);

**this**.setDefaultCloseOperation(JFrame.***EXIT\_ON\_CLOSE***);

**this**.add(image);

JButton button = **new** JButton("Reset Display");

**this**.add(button, "South");

button.addActionListener(**new** ButtonEventListener());

image.addMouseListener(**new** MouseEventListener());

// this.pack(); - После данного метода появляются отступы от картинки, почему не знаю

**this**.setVisible (**true**);

**this**.setResizable (**false**);

}

/\*\*

\* Метод вывода фрактала на экран

\*/

**private** **void** drawFractal()

{

**for**(**int** x = 0;x<**this**.size;x++)

{

**for** (**int** y = 0;y<**this**.size;y++)

{

**double** xCoord = FractalGenerator.*getCoord* (range.x, range.x + range.width,size, x);

**double** yCoord = FractalGenerator.*getCoord* (range.y, range.y + range.height,size, y);

**int** iter = **this**.fg.numIterations(xCoord,yCoord);

**if** (iter == -1) **this**.image.drawPixel(x, y, 0);

**else**

{

// Выбор цвета в диапазоне от 0 до 16777216

// При возможных 2000 значениях

**int** color = 16777216/2000\*iter;

**this**.image.drawPixel(x, y, color);

}

}

}

**this**.image.repaint();

}

/\*\* Обработчик нажатия на кнопку \*\*/

**class** ButtonEventListener **implements** ActionListener {

**public** **void** actionPerformed(ActionEvent e) {

fg.getInitialRange(range);

image.clearImage();

drawFractal();

}

}

/\*\* Обработчик нажатия на изображение \*\*/

**class** MouseEventListener **implements** MouseListener {

@Override

**public** **void** mouseClicked(MouseEvent e) {

**int** xCoord = e.getX();

**int** yCoord = e.getY();

**double** xCoor = FractalGenerator.*getCoord*(range.x, range.x + range.width,size, xCoord);

**double** yCoor = FractalGenerator.*getCoord*(range.y, range.y + range.width,size, yCoord);

fg.recenterAndZoomRange(range, xCoor, yCoor, 0.5);

drawFractal();

}

@Override

**public** **void** mouseEntered(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseExited(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mousePressed(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

@Override

**public** **void** mouseReleased(MouseEvent arg0) {

// **TODO** Auto-generated method stub

}

}

**public** **static** **void** main(String[] args) {

FractalExplorer app = **new** FractalExplorer(800);

app.createAndShowGUI();

app.drawFractal();

app.repaint();

}

}

# Результат работы программы

