**Федеральное агентство связи**

Ордена Трудового Красного Знамени

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

Кафедра «Информатика»

Лабораторная работа №5: Выбор и сохранение фракталов

по дисциплине «Кроссплатформенные технологии программирования»

В данной лабораторной работе генератор фракталов будет расширен двумя новыми функциями. Во-первых, вы добавите поддержку нескольких фракталов и реализуете возможность выбирать нужный фрактал из выпадающего списка. Во-вторых, вы добавите поддержку сохранения текущего изображения в файл.

Выполнил: студент

группы БСТ1803

Корнеев Александр

Вариант 12

Москва 2020

**Содержание**

[1 Задание на разработку программы 3](#_Toc41758209)

[2 Разработанные функции и классы 7](#_Toc41758210)

[2.1 Класс FractalExplorer 7](#_Toc41758211)

[2.1.1 Подкласс ButtonEventListener с методом actionPerformed 7](#_Toc41758212)

[2.2 Класс BurningShip 7](#_Toc41758213)

[2.2.1 Метод void getInitialRange 7](#_Toc41758214)

[2.2.2 Метод int numIterations 8](#_Toc41758215)

[2.3 Класс Tricorn 8](#_Toc41758216)

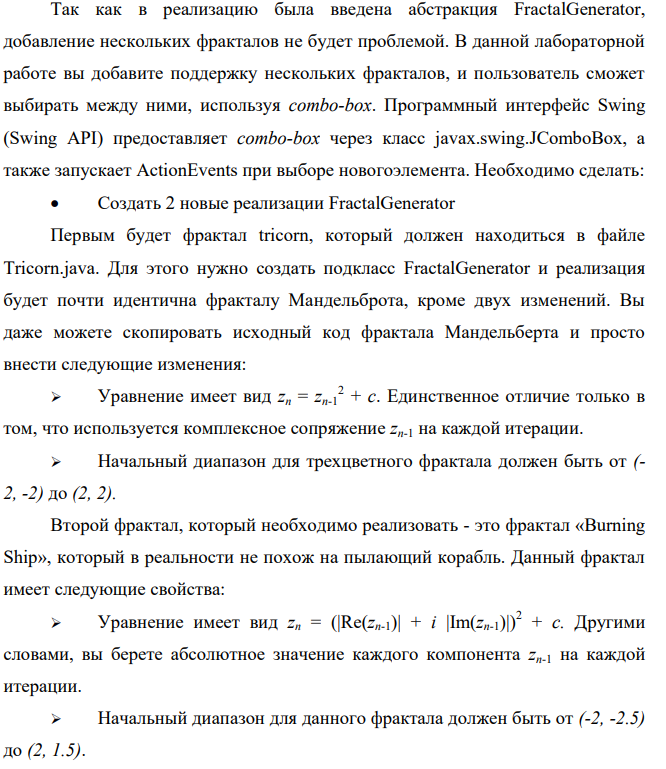
[2.3.1 Метод void getInitialRange 8](#_Toc41758217)

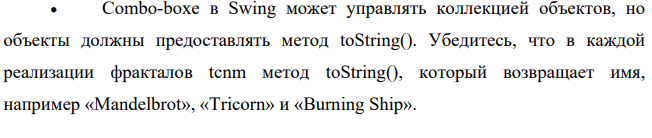
[2.3.2 Метод int numIterations 8](#_Toc41758218)

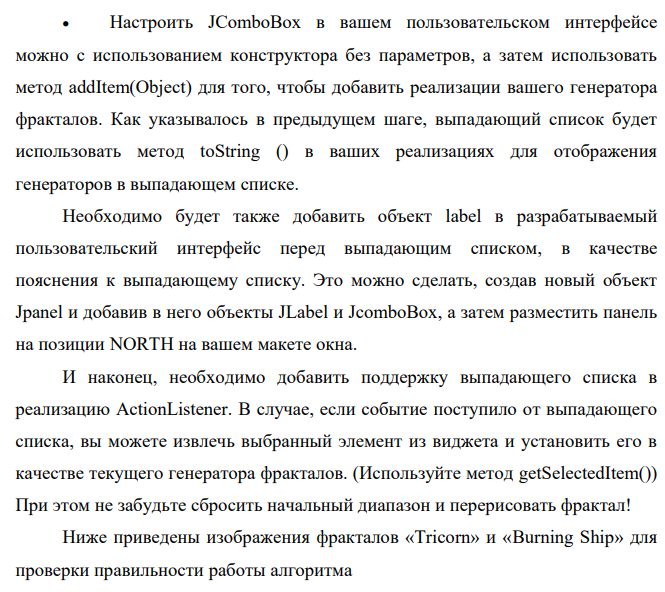
[3 Код программы 8](#_Toc41758219)

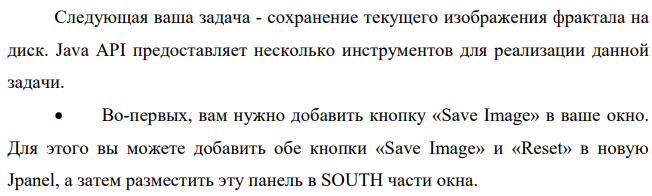
[4 Результат работы программы 14](#_Toc41758220)

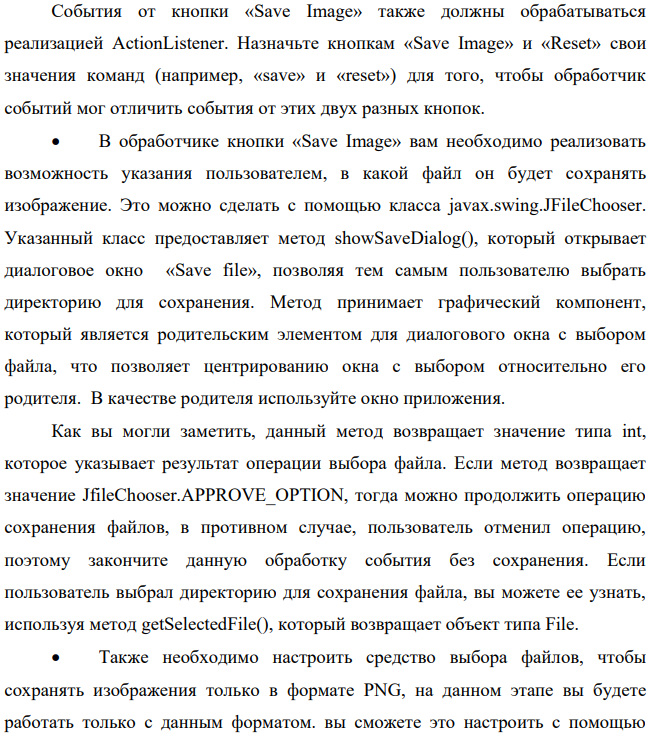
# Задание на разработку программы

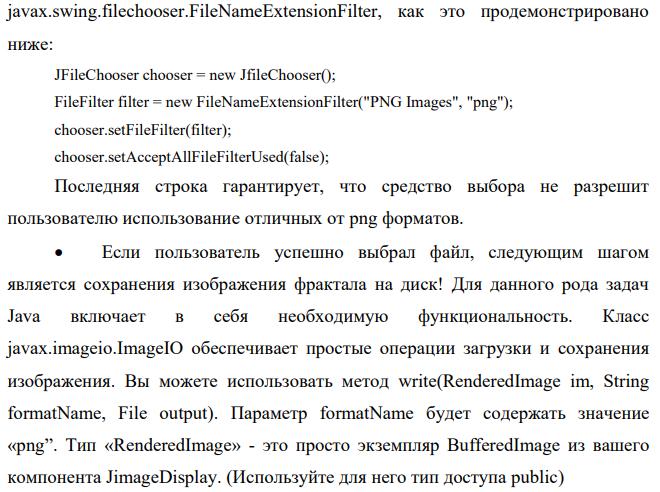


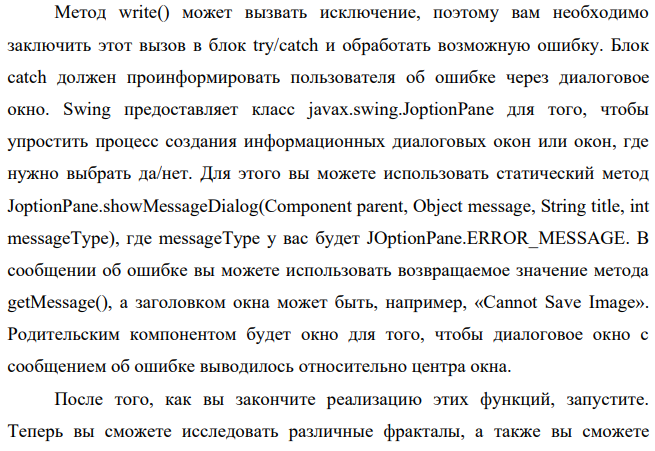


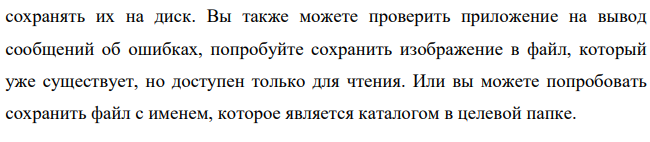












# Разработанные функции и классы

## Класс FractalExplorer

Класс отрисовывающий графический интерфейс Swing.

* + 1. Подкласс ButtonEventListener с методом actionPerformed

Добавлена обработка кнопок сохранения, выбора фрактала.

## Класс BurningShip

Реализует методы, необходимые для генерации фрактала “Горящий корабль”

* + 1. Метод void getInitialRange

Метод определяющий область комплексной плоскости для фрактала “Горящий корабль”

* + 1. Метод int numIterations

Метод, вычисляющий количество итераций для отрисовки точки

## Класс Tricorn

Реализует методы, необходимые для генерации фрактала “Треуголка”

* + 1. Метод void getInitialRange

Метод определяющий область комплексной плоскости для фрактала “Треуголка ”

* + 1. Метод int numIterations

Метод, вычисляющий количество итераций для отрисовки точки

# Код программы

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

/\*\*

\* Класс реализующий методы необходимые для генерации

\* фрактала вида "Треуголка"

\*/

**public** **class** Tricorn **extends** FractalGenerator {

/\*\* Константа определяющая максимальное количество

\* итераций при вычислении фрактала

\*/

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

**public** Tricorn() {

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

}

/\*\* Метод, определяющий область комплексной

\* плоскости для фрактала "Треуголка"

\*/

@Override

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

range.x = -2;

range.y = -2;

range.width = 4;

range.height = 4;

}

@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.Sopr();

z.Squaring();

z.AddComlp(c);

iter++;

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

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

}

}

/\*\*

\* **@return** Название фрактала - Tricorn

\*/

**public** String toString()

{

**return** "Tricorn";

}

}

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

/\*\*

\* Класс реализующий методы необходимые для генерации

\* фрактала вида "Горящий корабль"

\*/

**public** **class** BurningShip **extends** FractalGenerator {

/\*\* Константа определяющая максимальное количество

\* итераций при вычислении фрактала

\*/

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

**public** BurningShip() {

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

}

/\*\* Метод, определяющий область комплексной

\* плоскости для фрактала Горящий корабль

\*/

@Override

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

range.x = -2;

range.y = -2.5;

range.width = 4;

range.height = 4;

}

@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.Abs();

z.Squaring();

z.AddComlp(c);

iter++;

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

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

}

}

/\*\*

\* **@return** Название фрактала - BurningShip

\*/

**public** String toString()

{

**return** "Burning Ship";

}

}

/\*\*

\* 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);

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

fg = **new** Mandelbrot();

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

rbutton.setActionCommand("reset");

rbutton.addActionListener(**new** ActionEventListener());

JButton sbutton = **new** JButton("Save Button");

sbutton.setActionCommand("save");

sbutton.addActionListener(**new** ActionEventListener());

JPanel bpanel = **new** JPanel();

bpanel.add(rbutton);

bpanel.add(sbutton);

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

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

JComboBox box = **new** JComboBox();

box.addItem(**new** Mandelbrot());

box.addItem(**new** Tricorn());

box.addItem(**new** BurningShip());

box.addActionListener(**new** ActionEventListener());

JLabel label = **new** JLabel("Fractal:");

JPanel panel = **new** JPanel();

panel.add(label);

panel.add(box);

**this**.add(panel,"North");

// 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 событий\*\*/

**class** ActionEventListener **implements** ActionListener {

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

/\*

\* Обработка запускается в случае, если

\* действие вызвано объектом JComboBox

\*/

**if** (e.getSource() **instanceof** JComboBox)

{

JComboBox box = (JComboBox)e.getSource();

fg = (FractalGenerator)box.getSelectedItem();

fg.getInitialRange(range);

image.clearImage();

drawFractal();

}

/\*

\* Обработка запускается в случае, если действие

\* вызвано кнопкой с действием "save"

\*/

**else** **if**(((JButton)e.getSource()).getActionCommand() == "save")

{

// Получение объекта JFrame из кнопки количество getParent найдено методом тыка

JFrame jf = (JFrame)(((JButton)e.getSource()).getParent().getParent().getParent().getParent().getParent());

JFileChooser fc = **new** JFileChooser();

FileFilter filter = **new** FileNameExtensionFilter("PNG Images", "png");

fc.setFileFilter(filter);

fc.setAcceptAllFileFilterUsed(**false**);

**if**(fc.showSaveDialog(jf) != fc.***APPROVE\_OPTION***) System.***out***.println("Отмена сохранения");

**else**

{

File f = fc.getSelectedFile();

**try** {

ImageIO.*write*(image.bImage, "png", f);

} **catch** (IOException e1) {

JOptionPane.*showMessageDialog*(jf, e1.getMessage(), "Cannot Save Image", JOptionPane.***ERROR\_MESSAGE***);

}

}

}

/\*

\* Обработка зпускается в случае если действие

\* вызвано кнопкой не с действием "save"

\*/

**else**

{

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 e) {

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

}

@Override

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

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

}

@Override

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

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

}

@Override

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

// **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

