 Министерство образования и науки Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский Государственный Электротехнический Университет «ЛЭТИ» им. В.И. Ульянова (Ленина)»

Факультет компьютерных технологий и информатики

Кафедра автоматики и процессов управления

**ОТЧЕТ**

по лабораторной работе №2

по дисциплине «СМиСПИС»

Вариант №2

|  |  |  |
| --- | --- | --- |
| Студент гр. 5371 |  | Нургазы Б. |
| Студентка гр. 5371 |  | Есенбаев Ч. |
| Студент гр. 5371 |  | Ильясов Е. |
| Преподаватель |  | Кораблев Ю.А. |

Санкт-Петербург

2020

1. **Цель работы**

Изучение объектной модели Java, классы, интерфейсы, наследование, полиморфизм.

1. **Задание на лабораторную работу №1. Вариант №2.**

Разработать упрощенную иерархическую модель классов предметной области согласно варианту задания в таблице 1. Реорганизовать код в соответствии с разработанной диаграммой классов.

Таблица 1 – Вариант задания

|  |  |  |
| --- | --- | --- |
| Вариант | Предметная область | Программная реализация |
| 3 | Фрактальные поверхности | Фрактал с алгоритмом срединного смещения |

1. **Выполнение лабораторной работы**

**Диаграмма классов предметной области**

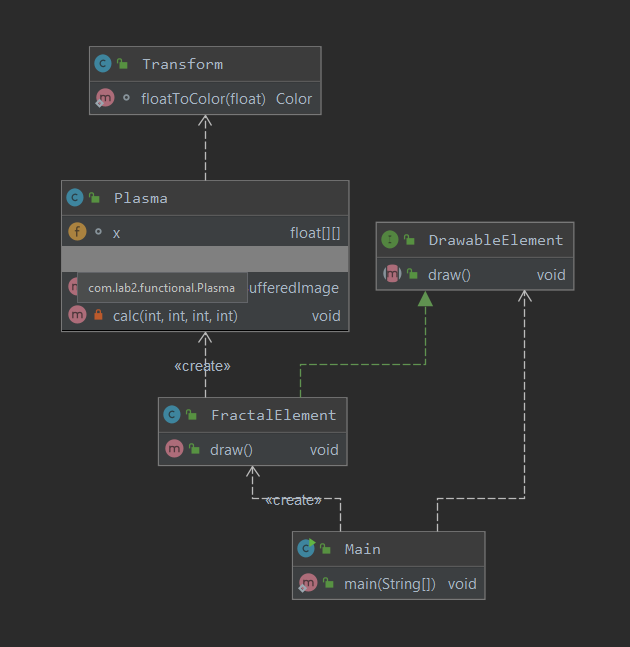


Рисунок 1 – UML диаграмма классов разработанной программы

**Описание классов предметной области**

В табл. 2 представлены описания основных классов переработанной программы.

Таблица 2 – Описание классов предметной области

|  |  |
| --- | --- |
| Имя класса | Описание |
| Main | Класс, предназначенный для точки входа в программу |
| DrawableElement | Основной интерфейс для элемента, который может быть нарисован, содержащий единственный метод draw() |
| Plasma | Класс реализующий Фрактал с алгоритмом срединного смещения |
| FractalElement | Класс реализующий вывод Фрактала с алгоритмом срединного смещения |
| Transform | Класс изменяющий цвет элемента |

1. **Тестирование переписанной программы**

На рис.2 показана отрисовка фрактала в исходной программе, написанной в парадигме функционального стиля.

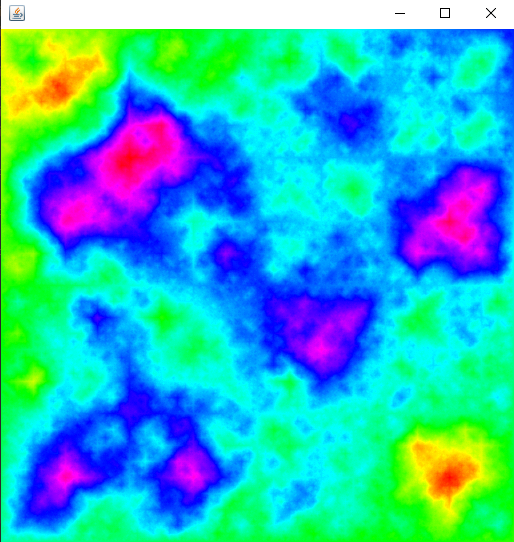


Рисунок – 2. Результат работы переписанной программы.

**Исходный код переписанной программы**

package com.lab2.functional;  
  
  
  
import java.awt.Color;  
import java.awt.Graphics;  
import java.awt.Graphics2D;  
import java.awt.Image;  
import java.awt.image.BufferedImage;  
import javax.swing.JFrame;  
import javax.swing.JPanel;  
  
public class Main  
{  
 public static void main(String[] args) {  
 DrawableElement drawableElement = new FractalElement();  
 drawableElement.draw();  
 }  
}

package com.lab2.functional;  
  
public interface DrawableElement {  
 void draw();  
}

package com.lab2.functional;  
  
import javax.swing.\*;  
import java.awt.\*;  
  
public class FractalElement implements DrawableElement {  
 public void draw() {  
 final Image image = new Plasma().createPlasma(10);  
 JFrame frame = new JFrame();  
 frame.addNotify();  
 frame.setSize(frame.getInsets().left  
 + frame.getInsets().right + image.getWidth(null),  
 frame.getInsets().top  
 + frame.getInsets().bottom + image.getHeight(null));  
 frame.setDefaultCloseOperation(JFrame.*EXIT\_ON\_CLOSE*);  
  
 frame.add(new JPanel()  
 {  
 @Override  
 public void paintComponent(Graphics g) {  
 Graphics2D G = (Graphics2D) g;  
 if (image != null) {  
 G.drawImage(image, 0, 0, null);  
 }  
 }  
 });  
 frame.setVisible(true);  
 }  
}

package com.lab2.functional;  
  
import java.awt.\*;  
  
public class Transform {  
  
 static Color floatToColor(float x)  
 {  
 if (x <= 0)  
 {  
 return new Color(1f, 0f, 0f);  
 }  
 if (x <= 1f/6)  
 {  
 return new Color(1f, x \* 6f, 0f);  
 }  
 if (x <= 2f/6)  
 {  
 return new Color(1f - (x - 1f/6) \* 6f, 1f, 0f);  
 }  
 if (x <= 3f/6)  
 {  
 return new Color(0f, 1f, (x - 2f/6f) \* 6f);  
 }  
 if (x <= 4f/6)  
 {  
 return new Color(0f, 1f - (x - 3f/6f) \* 6f, 1f);  
 }  
 if (x <= 5f/6)  
 {  
 return new Color((x - 4f/6) \* 6f, 0f, 1f);  
 }  
 if (x < 1f)  
 {  
 return new Color(1f, 0f, 1f - (x - 5f/6f) \* 6f);  
 }  
 return new Color(1f, 0f, 0f);  
 }  
}

package com.lab2.functional;  
  
import java.awt.image.BufferedImage;  
  
import static com.lab2.functional.Transform.*floatToColor*;  
  
public class Plasma {  
  
 float[][] x;  
  
 public BufferedImage createPlasma(int countLevels)  
 {  
 int size = (int)Math.*round*(Math.*pow*(2, countLevels));  
 BufferedImage image = new BufferedImage(size + 1, size + 1,  
 BufferedImage.*TYPE\_INT\_RGB*);  
 x = new float[size + 1][size + 1];  
 x[0][0] = (float)Math.*random*() \* size / 2 - size / 4;  
 x[0][size] = (float)Math.*random*() \* size / 2 - size / 4;  
 x[size][0] = (float)Math.*random*() \* size / 2 - size / 4;  
 x[size][size] = (float)Math.*random*() \* size / 2 - size / 4;  
 calc(0, 0, size, size);  
 float min = Float.*MAX\_VALUE*;  
 float max = Float.*MIN\_VALUE*;  
 for (int i = 0; i <= size; ++i)  
 {  
 for (int j = 0; j <= size; ++j)  
 {  
 if (x[i][j] < min)  
 {  
 min = x[i][j];  
 }  
 if (x[i][j] > max)  
 {  
 max = x[i][j];  
 }  
 }  
 }  
  
 for (int i = 0; i <= size; ++i)  
 {  
 for (int j = 0; j <= size; ++j)  
 {  
 image.setRGB(j, i,  
 *floatToColor*((x[i][j] - min) / (max - min)).getRGB());  
 }  
 }  
 return image;  
 }  
  
 private void calc(int a, int b, int c, int d)  
 {  
 if (a + 1 == c || b + 1 == d)  
 {  
 return;  
 }  
 float r = (c - a) + (d - b);  
 int e = (a + c) / 2;  
 int f = (b + d) / 2;  
 x[e][f] =  
 (x[a][b] + x[a][d] + x[c][b] + x[c][d]) / 4 +  
 r \* (float)(Math.*random*() - .5);  
 x[e][b] = (x[a][b] + x[c][b]) / 2;  
 x[e][d] = (x[a][d] + x[c][d]) / 2;  
 x[a][f] = (x[a][b] + x[a][d]) / 2;  
 x[c][f] = (x[c][b] + x[c][d]) / 2;  
 calc(a, b, e, f);  
 calc(a, f, e, d);  
 calc(e, b, c, f);  
 calc(e, f, c, d);  
 }  
}

1. **Вывод**

В ходе выполнения лабораторной работы была изучена объектная модель Java, а также классы, интерфейсы, наследование и полиморфизм, полученные знания были применены на практике и использованы при переписывании программы, написанной в функциональном стиле, в ООП стиль.

1. **Зачетное задание**

**Описание задания**

Введите в структуру Ваших классов элементы для подсчета, хранения и чтения числа минимальных элементов, формирующих фрактал. Что Вы будете понимать под минимальным элементом определите сами.

**Решение**

Под минимальными элементами, формирующими фрактал будем понимать количество треугольников, из которых фрактал состоит.

Таким образом, нужно добавить подсчет количества вызовов функции calc() у объектов типа Plasma.

Модифицированная UML-диаграмма изображена ниже на рисунке 3.

