Министерство образования и науки Российской Федерации Федеральное государственное автономное образовательное учреждение высшего профессионального образования «Севастопольский государственный университет»

ОТЧЕТ

о выполнении лабораторной работы № 1 по дисциплине

«Рефакторинг ПО»

Выполнил:

ст. гр. ИС/б-22-1-о

Гюнтер М. Ю .

Проверил:

ст. преподаватель кафедры

“Информационные системы”

Петраков В. А.

Севастополь,2025

# Цель работы: исследовать эффективность составления методов при рефакторинга программного кода. Получить практические навыки применения приемов рефакторинга методов.

# Ход работы

## Исходный программный код, покрытый модульными тестами

Напишем на языке Java код, содержащий фрагменты, предполагающие рефакторинг с помощью 7 приёмов составления методов, а также покрытый модульными тестами, которые гарантируют работоспособность кода после проведения рефакторинга (листинги ‎2.1-).

Листинг ‎2.1 – Класс Order

package refactoring.model;

public class Order {

private final String id;

private final int quantity;

private final double unitPrice;

private final double weightKg;

public Order(String id, int quantity, double unitPrice, double weightKg) {

this.id = id;

this.quantity = quantity;

this.unitPrice = unitPrice;

this.weightKg = weightKg;

}

public String getId() { return id; }

public int getQuantity() { return quantity; }

public double getUnitPrice() { return unitPrice; }

public double getWeightKg() { return weightKg; }

@Override

public String toString() {

return "Order{" + id + ", q=" + quantity + ", price=" + unitPrice + "}";

}

}

Листинг ‎2.1 – Класс Order

package refactoring.model;

public class Order {

private final String id;

private final int quantity;

private final double unitPrice;

private final double weightKg;

public Order(String id, int quantity, double unitPrice, double weightKg) {

this.id = id;

this.quantity = quantity;

this.unitPrice = unitPrice;

this.weightKg = weightKg;

}

public String getId() { return id; }

public int getQuantity() { return quantity; }

public double getUnitPrice() { return unitPrice; }

public double getWeightKg() { return weightKg; }

@Override

public String toString() {

return "Order{" + id + ", q=" + quantity + ", price=" + unitPrice + "}";

}

}

С помощью метода loadUi модуля uic происходит импорт созданного ранее GUI. С помощью метода setText объекта Label происходит изменение текстового содержимого заголовка.

Результат запуска кроссплатформенного приложения с использованием вертикальной схемы размещения приведен на рисунке ‎2.3.

Изображение выглядит как текст, снимок экрана, дисплей, Шрифт

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок ‎2.3 – Результат запуска приложения с вертикальной схемой

Результат запуска кроссплатформенного приложения с использованием горизонтальной схемы размещения приведен на рисунке ‎2.4.

Изображение выглядит как текст, снимок экрана, дисплей, программное обеспечение

Содержимое, созданное искусственным интеллектом, может быть неверным.

Рисунок ‎2.4 – Результат запуска приложения с горизонтальной схемой

# Вывод:

В результате выполнения лабораторной работы были изучены базовые принципы работы фреймворка Qt, в частности PyQt. Было получено представление о проектирования GUI на основе виджетов с выбором схемы размещения элементов. Также были использованы различные методы объектов интерфейса пользователя, в частности метод setText объекта Label.