КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ КОМП’ЮТЕРНИХ НАУК ТА КІБЕРНЕТИКИ

Звіт

про виконання

Лабораторної роботи №1

Виконала

студентка групи ПІ-23

Петрик Юлія Олександрівна

Київ 2023

Завдання 1а

**Design/Implementation Modeling**

У цій лабораторній роботі я працювала із програмою, яка реалізовує роботу зі списками. Для цієї програми я створила такі діаграми: Use Case, Class, Sequence, State, Activity. Також я створила глосарій, де описала всі використані терміни. Після рефакторингу змін зазнала лише Class Diagram. Для порівняння я додала обидві діаграми в репозиторій.

**Рефакторинг**

Я відокремила роботу з користувачем в окремі класи. Тепер методи, які на діаграмі класів належали Global, належать класу Menu і його дочірнім класам MenuInt (реалізуємо роботу зі списками, елементи якого мають цілий тип) MenuString (реалізуємо роботу зі списками, елементи якого є рядками). Така ієрархія дозволяє уникнути дублювання коду і дотримуватися принципу наслідування. Вони своєю чергою використовують клас List як параметр своїх методів. Також роботу з користувачем я розбила на декілька методів. Для того щоб слідувати принципу DRY, я винесла спільні методи класів MenuInt і MenuString до Menu і застосувала Template method pattern. Методи класу Main я зробила protected для того, щоб слідувати принципу інкапсуляції, адже вони використовуються тільки дочірніми класами.

У методі megre у класі MergeSort я винесла деякі частини в окремі методи, які використовуються даним. Зокрема, створила метод split\_array, який копіює елементи списку до тимчасових списків, і метод complete, який записує елементи, які залишилися після виконання певної частини алгоритму в основний список. Тепер ця задача реалізується більшою кількістю простіших методів, що слідує принципу KISS.

Під час рефакторингу методу sort класу BubbleSort я вирішила винести порівняння двох елементів списку в окремий метод compare. Після цього у класах BubbleSort i MergeSort з’явився однаковий метод, тому для того, щоб слідувати принципу DRY, я перенесла цей метод у батьківський клас Sort і зробила protected для того, щоб його могли використовувати дочірні класи класу Sort.

**Аналіз**

На мою думку, у програмі дотримано принципів ООП (інкапсуляція, успадкування, поліморфізм), KISS, DRY, абстракції. Вище згадано, що рефакторинг коду також було проведено з урахуванням цих принципів. Ця програма також підтримує можливість розвитку. Наприклад, введення нових типів списків, типів елементів у списках, видів сортування та інших функцій для роботи зі списком. Крім того, у результаті рефакторигну код стало легше читати, його поменшало, стало легше розібратися, що роблять ці чи інші методи. Уся функціональність збереглася і працює коректно. Підбиваючи підсумки, я можу сказати, що рефакторинг було проведено успішно.

**Посилання**

*Репозиторій:*

<https://github.com/yuliapetryk/OOP_Labs_sem4/tree/master/Lab1>

*Документація:*

<https://yuliapetryk.github.io/OOP_Labs_sem4/Lab1/doc/html/>