1МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ



**Дніпровський національний університет  
залізничного транспорту імені академіка В. Лазаряна**

Кафедра «Комп’ютерні інформаційні технології»

**Лабораторна робота №2**

**з дисципліни «Архітектура та проектування ПЗ»**

**на тему: «Файловий менеджер»**

Виконав: студент гр. ПЗ1911

Сіньков Г.О.

Прийняла: ас. каф. КІТ

Куроп'ятник О. С.

Дніпро, 2020

**Лабораторна робота №2**

**Тема**. Основи процесу розробки програмного забезпечення. Класифікація варіантів використання.

**Мета**. Ознайомитися з основними етапами розробки програмного забезпечення (ПЗ). Отримати практичні навички класифікації варіантів використання ПЗ.

**Постановка задачі**

1.Загальна постановка: Виконати збір, аналіз та формалізацію вимог до ПЗ для предметної області згідно з індивідуальним завданням.

В ході збору вимог виконати огляд та аналіз документації (не менше п’яти джерел) та аналогів (повних та/або часткових не менше п’яти програмних продуктів (модулів)). В ході огляду документації можуть бути розглянуті:

- нормативні документи, які регулюють особливості розробки чи використання ПЗ;

- наукова та технічна література, що містить описи методики, програмних методів, алгоритмів, технологій і таке інше, які є необхідними для реалізації ПЗ;

- технічна документація до існуючого ПЗ.

При розгляді методів, алгоритмів, технологій тощо навести їх опис, основні характеристики. В частині аналізу вказати їх можливе застосування для даного проекту, недоліки та переваги.

При розгляді програмних аналогів зазначити назву, розробника (власника), основні функціональні можливості, близькі до передбаченої індивідуальним завдання предметної області. В ході аналізу виявити їх переваги та недоліки функціональної та нефункціональної частин. До останньої можна віднести інтерфейс програми, вимоги до програмного та апаратного забезпечення, необхідного для роботи ПЗ, його доступності, порядку роботи, надійності, супроводу та підтримки

На основі виконаного аналізу документації та аналогів сформулювати функціональні та нефункціональні вимоги. Розробити для них зовнішні функціональні специфікації, які включають:

- опис зовнішнього інформаційного середовище, в якому буде експлуатуватися ПЗ;

- визначення функцій ПЗ (див. п. «Теоретичні відомості»).

Виходячи з функціональних вимог, побудувати діаграму варіантів використання розроблюваного ПЗ, дати опис її елементів (включаючи сценарії взаємодії) та обґрунтувати зв’язки між ними.

2.Індивідуальна постановка: Виконати збір, аналіз та формалізацію вимог до ПЗ для предметної області – файловий менеджер.

**Специфікація класів сутностей у вигляді CRC-карток**

**Картка класу «Forma»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| - | - |
| Обов’язки | Зв’язки |
| Зберігає в собі форму меню | MenuController, Menu |

**Картка класу «Menu»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Forma | - |
| Обов’язки | Зв’язки |
| Зберігає в собі інтерфейс | Forma |

**Картка класу «MenuController»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Forma | - |
| Обов’язки | Зв’язки |
| Зв’язує інтерфейс с функціональною метою кнопки | Провайдер |

**Картка класу «Провайдер»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Forma | - |
| Обов’язки | Зв’язки |
| Обрання та відміна команди | Команда |

**Картка класу «Команда»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| - | - |
| Обов’язки | Зв’язки |
| Виповнення потрібної команди | Файл, каталог, функції роботі з файлами(пошук за назвою, переміщення, перейменування, видалення, копіювання) |

**Картка класу «Файл»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| - | - |
| Обов’язки | Зв’язки |
| Зберігає інформацію про файл | Команда |

**Картка класу «Каталог»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| - | - |
| Обов’язки | Зв’язки |
| Зберігає інформацію про каталог | Команда |

**Картка класу «Пошук за назвою»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Команда | - |
| Обов’язки | Зв’язки |
| Пошук файлу або каталог за назвою | Команда |

**Картка класу «Переміщення»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Команда | - |
| Обов’язки | Зв’язки |
| Переміщення файлу, каталога | Команда |

**Картка класу «Перейменування»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Команда | - |
| Обов’язки | Зв’язки |
| Зміна назви файлу, каталогу | Команда |

**Картка класу «Видалення»**

|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Команда | - |
| Обов’язки | Зв’язки |
| Видалення файлу, каталогу | Команда |

**Картка класу «Копіювання»**

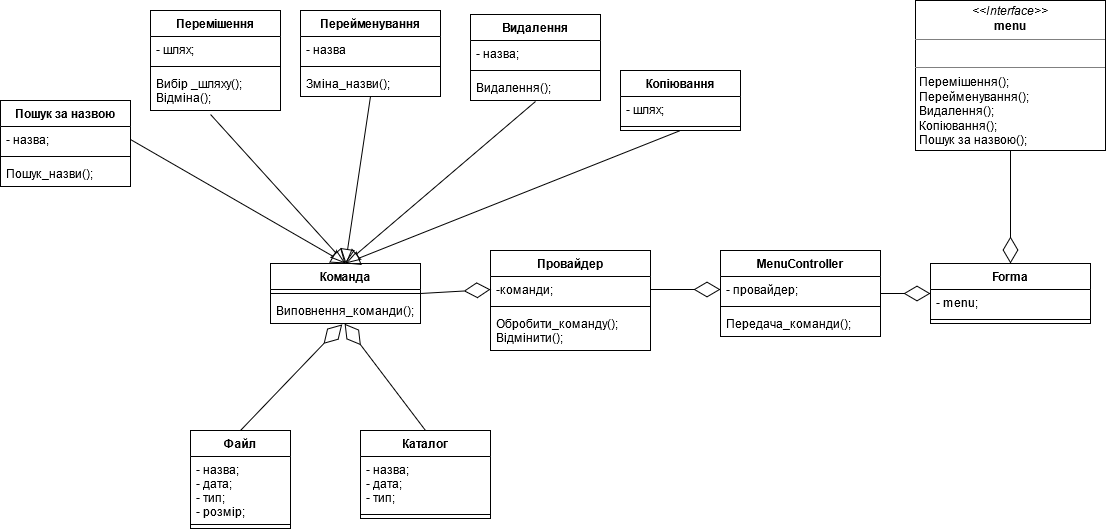
|  |  |
| --- | --- |
| Базовий клас | Похідні класи (нащадки) |
| Команда | - |
| Обов’язки | Зв’язки |
| Копіювання шляху файлу, каталогу | Команда |

**Опис та обґрунтування зав’язків між класами**

В об'єктно-орієнтованому програмуванні шаблон проектування Команда є загальнодоступним шаблоном, у якому об'єкт використовується для інкапсуляції всієї інформації, необхідної для виконання дій або вибору подій у більш пізній час. Ця інформація включає в себе ім'я методу, об'єкт, який є власником методу та значення параметрів методу.

Чотири термінали завжди пов'язані з шаблоном Команда: команди (command), приймач команди (receiver), викликач команди (invoker) та клієнт (client). Об'єкт Command знає про прийом і вибирає метод прийнятності. Значення параметрів приймач зберігаються в командах. Викликач об'єкт (invoker) знає, як виконати команду і, можливо, робить облік і записувати виконану команду. Викликач об'єкт (invoker) нічого не знає про конкретну команду, він знає лише про інтерфейс. В якості об'єкта (визивання об'єкта та декількох об'єктів команди) належить об'єкт клієнта (client). Клієнт вирішує, які команди виконують і коли. Щоб виконати команду, передаючи об'єкт команди, що шукає об'єкт (виклик).

**Діаграма класів**



Висновок: Діаграма класів (англ. Class diagram) - структурна діаграма мови моделювання UML, що демонструє загальну структуру ієрархії класів системи, їх кооперацій, атрибутів (полів), методів, інтерфейсів і взаємозв'язків між ними. Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування.