Міністерство освіти і науки

Національний університет «Львівська політехніка»

Інститут комп’ютерних наук та інформаційних технологій

Кафедра автоматизованих систем управління



**Лабораторна робота №5**

**З дисципліни «Прикладне програмування»**

Тема:

***« Комплексна лабораторна робота(етап 2) »***

Виконав:

Данич Іван

Група:

КН-202

Перевірив:

Цимбал Ю. В.

**Львів 2019**

**Тема:** Комплексна лабораторна робота(етап 2)

**Мета роботи:**Освоїти паттерн «Команда». Створити консольне меню програми.

1. **Опис лабораторної роботи**

Створити консольну програму на мові програмування Java згідно до індивідуального завдання (ваш номер варіанту, згенерований випадковим чином, можна знайти у файлі «Варіанти завдань.pdf»). Саме завдання описане досить поверхово, якщо ви претендуєте на високу оцінку – його потрібно розширити. Наприклад, у завданні може бути сказано про «створити кілька квіток, з наявних квіток сформувати букет». Цілком логічно буде додати можливість видалити квітку, поміняти параметри існуючої квітки тощо. ***Оцінка за лабораторну роботу залежатиме і від глибини пропрацювання предметної області.***

Окрім цього, програма повинна відповідати таким додатковим вимогам:

• Використовувати можливості ООП: класи, успадкування, поліморфізм, інкапсуляцію.

• Кожний клас повинен мати назву, яка повністю описує його суть, і інформативний склад.

• Успадкування потрібно використовувати тільки тоді, коли воно має сенс.

• При записі програми потрібно використовувати домовленості щодо оформлення коду java code convention.

• Класи потрібно грамотно розкласти по пакетах.

• Для збереження параметрів ініціалізації потрібно використовувати файли.

• Повинне бути консольне меню для вибору різних дій, наприклад: «завантажити з файлу», «довідка», «вихід» тощо.

* 1. **Завдання**

Домашні електроприлади. Визначити ієрархію електроприладів. Включити деякі в розетку. Підрахувати споживану потужність. Здійснити сортування приладів у квартирі на основі потужності. Знайти прилад в квартирі, що відповідає заданому діапазону параметрів.

* 1. **Етап**

Реалізуйте консольне меню до вашого завдання з використанням паттерну «Команда». На цьому етапі не потрібно реалізовувати функціонал конкретних пунктів меню – тільки створити «кістяк» вашої програми.

1. **Хід роботи**

**2.1 Структура меню**

**Рівень: будинок**

1 – Переглянути список приладів

2 – Відсортувати список приладів

2.1 За станом

2.2 За потужнісю

2.3 За квартирами

3 – Вивести загальну потужність

4 – Вивести вартість спожитої енергії за певний період часу

5 – Змінити список приладів

5.1 Вкл\Вимк приладу

5.2 Змінити характеристики приладу

5.3 Видалити прилад

5.4 Перемістити прилад

6 – Пошук приладу за заданими параметрами

7 – Переглянути список квартир

8 – Перейти до квартири

* 1. **Реалізація меню в коді**

Меню організоване через окремий клас: Menu

Основний метод це головне меню: openMenu()

Інші методи це підменю.

Клас меню є взаємодіє з користувачем, і в залежності від вибору є ініціатором певної команди. Всі команди проходять через клас відправник, який веде список виконаних команд

**Висновок:**

Отже, вигляд меню залежить від рівня споруди. Меню грає роль ініціатора в паттерні команда, команди викликаються через проміжний класс відправника. В класі відправника можна зберігати список виконаних команд. Паттерн команда дає можливість відділити бізнес-логіку від інтерфейсу взаємодії з користувачем, але недоліком є велика кількість класів команд.

1. **Додатки**

**3.1 Код програми**

**import** Buildings.Building;  
**import** Buildings.House;  
**import** Command.Commands.ApilianceCommands.ChangeParametersCommand;  
**import** Command.Commands.BuildingCommands.ChangeLocationCommand;  
**import** Command.Commands.BuildingCommands.DeleteApplianceCommand;  
**import** Command.Commands.BuildingCommands.PrintCommands.\*;  
**import** Command.Commands.BuildingCommands.SearchApplianceCommand;  
**import** Command.Commands.BuildingCommands.SetLevelCommand;  
**import** Command.Commands.BuildingCommands.SortCommands.SortListByApartmentCommand;  
**import** Command.Commands.BuildingCommands.SortCommands.SortListByConditionCommand;  
**import** Command.Commands.BuildingCommands.SortCommands.SortListByPowerCommand;  
**import** Command.Commands.BuildingCommands.SortCommands.SortListByRoomCommand;  
**import** Command.Invoker;  
  
**import** java.io.BufferedReader;  
**import** java.io.InputStreamReader;  
  
**public class** Menu {  
 Building **house**;  
 Invoker **invoker**;  
 **int level**; *// 1 - House; 2 - Apartment; 3 - Room* **public** Menu(){  
 **house** = **new** House();  
 **invoker** = **new** Invoker();  
 **level** = 1;  
 }  
  
 **public void** openMenu() **throws** Exception{  
 **int** key = 10;  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 **while**(**true**) {  
 System.***out***.println(**"1 – Переглянути список приладів\n"** +  
 **"2 – Відсортувати список приладів\n"** +  
 **"3 – Вивевести загальну потужність\n"** +  
 **"4 – Вивести вартість спожитої енергії за певний період часу\n"** +  
 **"5 – Змінити список приладів\n"** +  
 **"6 – Пошук приладів за заданими параметрами"**);  
 **switch** (**level**) {  
 **case** 1: {  
 System.***out***.println(**"7 – Переглянути список квартир\n"** +  
 **"8 – Перейти до квартири"**);  
 **break**;  
 }  
 **case** 2: {  
 System.***out***.println(**"7 – Переглянути список кімнат\n"** +  
 **"8 – Перейти до кімнати\n"** +  
 **"9 – Повернутись до будинку"**);  
 **break**;  
 }  
 **case** 3: {  
 System.***out***.println(**"8 – Повернутись до квартири\n"** +  
 **"9 – Повернутись до будинку"**);  
 **break**;  
 }  
 }  
 System.***out***.println(**"10 - Вихід"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **if** (key == 10) **break**;  
 **switch** (key){  
 **case** 1:{  
 **invoker**.setCommand(**new** PrintListCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 2:{  
 openSortMenu();  
 **break**;  
 }  
 **case** 3:{  
 **invoker**.setCommand(**new** PrintTotalPowerCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 4:{  
 **double** days = 0;  
 System.***out***.println(**"Введіть кількість днів:"**);  
 Double.*parseDouble*(reader.readLine());  
 **invoker**.setCommand(**new** PrinPriceOfEnergyCommand(**house**,days));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 5:{  
 openChangeMenu();  
 **break**;  
 }  
 **case** 6:{  
 **invoker**.setCommand(**new** SearchApplianceCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 7:{  
 **if**(**level** == 1)  
 **invoker**.setCommand(**new** PrintApartmentListCommand(**house**));  
 **else invoker**.setCommand(**new** PrintRoomListCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 8:{  
 **if**(**level** != 3) {  
 **invoker**.setCommand(**new** SetLevelCommand(**house**, **level** + 1));  
 **level** +=1;  
 }  
 **else** {  
 **invoker**.setCommand(**new** SetLevelCommand(**house**,**level**-1));  
 **level** -=1;  
 }  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 9:{  
 **invoker**.setCommand(**new** SetLevelCommand(**house**,1));  
 **level** = 1;  
 **invoker**.invoke();  
 **break**;  
 }  
 **default**:{  
  
 **break**;  
 }  
 }  
 }  
 }  
  
 **private void** openChangeMenu() **throws** Exception{  
 **int** key = 1;  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 System.***out***.println(**"1 - Вкл\\Вимк прилад\n"** +  
 **"2 - Змінити характеристики приладу\n"** +  
 **"3 - Видалити прилад\n"** +  
 **"4 - Перемістити прилад"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **switch** (key){  
 **case** 1:{  
 System.***out***.println(**"Список приладів:"**);  
 **invoker**.setCommand(**new** PrintListCommand(**house**));  
 **invoker**.invoke();  
 System.***out***.println(**"Введіть ід приладу стан якого хочете змінити:"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **invoker**.setCommand(**new** DeleteApplianceCommand(**house**,key));  
 **break**;  
 }  
 **case** 2:{  
 System.***out***.println(**"Список приладів:"**);  
 **invoker**.setCommand(**new** PrintListCommand(**house**));  
 **invoker**.invoke();  
 System.***out***.println(**"Введіть ід приладу параметри якого хочете змінити:"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **invoker**.setCommand(**new** ChangeParametersCommand(**house**.getApilianceById(key)));  
 **break**;  
 }  
 **case** 3:{  
 System.***out***.println(**"Список приладів:"**);  
 **invoker**.setCommand(**new** PrintListCommand(**house**));  
 **invoker**.invoke();  
 System.***out***.println(**"Введіть ід приладу який хочете видалити:"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **invoker**.setCommand(**new** DeleteApplianceCommand(**house**,key));  
 **break**;  
 }  
 **case** 4:{  
 System.***out***.println(**"Список приладів:"**);  
 **invoker**.setCommand(**new** PrintListCommand(**house**));  
 **invoker**.invoke();  
 System.***out***.println(**"Введіть ід приладу який хочете перемістити:"**);  
 key = Integer.*parseInt*(reader.readLine());  
 **invoker**.setCommand(**new** ChangeLocationCommand(**house**,key));  
 **break**;  
 }  
 **default**:{  
 **break**;  
 }  
 }  
 }  
  
 **private void** openSortMenu() **throws** Exception {  
 **int** key = 1;  
 BufferedReader reader = **new** BufferedReader(**new** InputStreamReader(System.***in***));  
 System.***out***.println(**"1 - За станом\n"** +  
 **"2 - За потужнісю"**);  
 **switch** (**level**){  
 **case** 1:{  
 System.***out***.println(**"3 - За квартирами"**);  
 **break**;  
 }  
 **case** 2:{  
 System.***out***.println(**"3 - За кімнатами"**);  
 **break**;  
 }  
 **default**:{  
 **break**;  
 }  
 }  
 key = Integer.*parseInt*(reader.readLine());  
 **switch** (key){  
 **case** 1:{  
 **invoker**.setCommand(**new** SortListByConditionCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 2:{  
 **invoker**.setCommand(**new** SortListByPowerCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **case** 3:{  
 **if**(**level** == 1)  
 **invoker**.setCommand(**new** SortListByApartmentCommand(**house**));  
 **else invoker**.setCommand(**new** SortListByRoomCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 **default**:{  
 **invoker**.setCommand(**new** SortListByConditionCommand(**house**));  
 **invoker**.invoke();  
 **break**;  
 }  
 }  
 }  
  
}

**3.2 Реалізація**

