Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЛЬВІВСЬКА ПОЛІТЕХНІКА»



Звіт

З лабораторної роботи №2

З дисципліни «Кросплатформні засоби програмування»

На тему:

**КЛАСИ ТА ПАКЕТИ**

Виконав: Тупісь М.Н.

ст. гр.KI-304

Прийняв:

Іванов Ю.С.

Львів – 2025

**Мета**: ознайомитися з процесом розробки класів та пакетів мовою Java.

**Теоретичні відомості**

Мова Java є повністю об’єктно-орієнтованою мовою програмування, тому вона дозволяє писати програми лише з використанням об’єктно-орієнтованих парадигм програмування, що базуються на понятті класів.

Необов’язковий специфікатор доступу public робить клас загальнодоступним. У кожному файлі з кодом програми може бути лише один загальнодоступний клас, ім’я якого співпадає з назвою файлу, та безліч класів без специфікатора public. Створення об’єкту класу складається з двох етапів: оголошення та ініціалізації посилання на об’єкт.

Ініціалізація посилання на об’єкт класу здійснюється за допомогою оператора new і вказування конструктора, який має збудувати об’єкт. Одержаний в результаті цих операцій об’єкт розташується у області оперативної пам’яті що зветься ”куча”.

При створенні об’єктів дозволяється суміщати оголошення та ініціалізацію об’єктів, а також створювати анонімні об’єкти. Якщо посилання на об’єкт не посилається на жоден об’єкт, то йому слід присвоїти значення null. На відміну від полів-посилань на об’єкти, локальні змінні-посилання на об’єкти не ініціалізуються значенням null при оголошенні. Для них ініціалізацію посилання слід проводити явно.

Метод – функція-член класу, яка призначена маніпулювати станом об’єкту класу.

Методи можуть бути перевантаженими. Перевантаження методів відбувається шляхом

вказування різної кількості параметрів та їх типів методам з однаковими назвами.

онструктори, методи, та поля класу можуть бути відкритими (public),закритими (private) та захищеними (protected), що визначається специфікаторомдоступу.

Специфікатор доступу public робить елемент класу загальнодоступним в межахпакету (набору класів, з яких складається програма).

Специфікатор доступу private робить елемент класу закритим (недоступним)для всіх зовнішніх відносно даного класу елементів програми (включаючи похідні класи).

Специфікатор доступу protected робить елемент класу закритим (недоступним)для всіх зовнішніх відносно даного класу елементів програми, проте цей елемент будезагальнодоступним для похідних класів.Якщо будь-який елемент класу не має специфікатора доступу, то цей елементавтоматично стає відкритим та видимим у межах пакету (не плутати з public).Всі елементи класу, що оголошені без використання ключового слова static,належать об’єкту класу. Тобто, кожен об’єкт класу містить власну копію цих елементівкласу.

Ключове слово static робить поле або метод членом класу, а не об’єкту, тобтовони є спільними для всіх об’єктів класу. Оскільки клас існує завжди, на відміну відоб’єктів, які створюються в процесі роботи програми, то статичні елементи класу доступні5навіть тоді, коли ще не створено жодного об’єкту класу.

Цей підхід використовується принаписанні методу main з якого починається виконання консольної програми, бо намомент її запуску ще не існує жодного об’єкту.Метод може генерувати виключну ситуацію.

Якщо виключна ситуація неопрацьовується у тілі методу, то вона повинна бути описана в оголошені методу післяключового слова throws.

Якщо виключна ситуація опрацьовується у тілі методу, тоцього робити не потрібно.Передача параметрів у метод відбувається по значенню шляхом копіюваннязначень реальних параметрів у формальні параметри методу. Якщо ці значення єпростими типами, то відбудеться копіювання значень. Якщо ці значення є посиланнями,то копіюватимуться не об’єкти, а посилання на об’єкти. Таким чином зміна значенняпосилання формального параметру в середині методу не вплине на значення посилання зайого межами.

Конструкторів може бути кілька. Конструктор без параметрів називається конструктором за замовчуванням. Якщо у класі не визначено жодного конструктора, то конструктор за замовчуванням генерується автоматично при компіляції (неявно). Він здійснює ініціалізацію полів об’єкту класу значеннями за замовчуванням (оскільки поля об’єкту на момент створення обов’язково мають бути ініціалізованими)

Поле (властивість) – це дані-члени класу, що призначені для зберігання стану об’єкту. Поле може бути статичним (в цьому випадку воно називається полем класу), незмінним (константне поле), простим типом чи об’єктом та мати різні рівні доступу, що визначаються специфікатором доступу. Допускається ініціалізація поля в місці оголошення.

Пакет – це механізм мови Java, що дозволяє об’єднувати класи в простори імен.Об’єднання класів в пакети дозволяє відділяти класи, що розроблені однимирозробниками, від класів, що розроблені іншими розробниками, забезпечуючи тим самимунікальність імен класів в межах програми та усуваючи можливі конфлікти імен класів.Пакети можуть бути вкладеними одні в одних, утворюючи цим самим ієрархії пакетів.Будь-який зв’язок між вкладеними пакетами відсутній. Всі стандартні пакети належатьієрархіям java і javax, наприклад, java.lang, java.util, java.net тощо.

Створення пакетів відбувається за допомогою оператора package з вказуванням назв пакету і під пакетів (за необхідності), що розділені крапкою. Оператор package вказується на початку тексту програми перед операторами import та визначенням класу.

Хід роботи

Завдання:

1. Написати та налагодити програму на мові Java, що реалізує у вигляді класу предметну

область згідно варіанту.

Програма має задовольняти наступним вимогам:

• програма має розміщуватися в пакеті Група.Прізвище.Lab2;

• клас має містити мінімум 3 поля, що є об’єктами класів, які описують складові частини предметної області;

• клас має містити кілька конструкторів та мінімум 10 методів;

• для тестування і демонстрації роботи розробленого класу розробити клас- драйвер;

• методи класу мають вести протокол своєї діяльності, що записується у файл;

• розробити механізм коректного завершення роботи з файлом (не надіятися на метод finalize());

• програма має володіти коментарями, які дозволять автоматично згенерувати

документацію до розробленого пакету.

2. Автоматично згенерувати документацію до розробленої програми.

3. Завантажити код на GitHub згідно методичних вказівок по роботі з GitHub.

4. Скласти звіт про виконану роботу з приведенням тексту програми, результату її

виконання та фрагменту згенерованої документації та завантажити його у ВНС.

5. Дати відповідь на контрольні запитання.

Варіант: 23 Електрична плитка

Лістинг програми мовою Java:

//file main.java

package Lab\_2;

public class Main {

public static void main(String[] args) {

ElectricStove obj = new ElectricStove("Bosch");

ElectricStove stove = new ElectricStove("Samsung Chef", "Induction", 3.5, 4,

"Samsung", 92.5, true, "Black");

stove.calculateEnergyConsumption(5); // розрахунок енергоспоживання за 5 годин

stove.showAllAttributes();

obj.showAllAttributes();

stove.comparePower(obj);

}

}

// file ElectricStove.java

package Lab\_2;

import java.io.FileWriter;

import java.io.IOException;

import java.io.PrintWriter;

import java.time.LocalDateTime;

import java.time.format.DateTimeFormatter;

public class ElectricStove {

private String name;

private String type;

private double power; // потужність (кВт)

private int burners; // кількість конфорок

private String manufacturer;

private double efficiency; // ККД (%)

private boolean hasOven;

private String color;

private double energyConsumed; // розрахунок енергії

// Конструктор без параметрів

public ElectricStove() {

this.name = "Custom Stove";

this.type = "unknown";

this.power = 0.0;

this.burners = 0;

this.manufacturer = "unknown";

this.efficiency = 0.0;

this.hasOven = false;

this.color = "unknown";

this.energyConsumed = 0.0;

}

// Конструктор з усіма параметрами

public ElectricStove(String name, String type, double power, int burners, String manufacturer,

double efficiency, boolean hasOven, String color) {

this.name = name;

this.type = type;

this.power = power;

this.burners = burners;

this.manufacturer = manufacturer;

this.efficiency = efficiency;

this.hasOven = hasOven;

this.color = color;

this.energyConsumed = 0.0;

}

// Конструктор з одним параметром

public ElectricStove(String name) {

this.name = name;

this.type = "unknown";

this.power = 0.0;

this.burners = 0;

this.manufacturer = "unknown";

this.efficiency = 0.0;

this.hasOven = false;

this.color = "unknown";

this.energyConsumed = 0.0;

}

// Логування дій

private void logAction(String message) {

try (PrintWriter writer = new PrintWriter(new FileWriter("Lab\_2/activity.log", true))) {

DateTimeFormatter dtf = DateTimeFormatter.ofPattern("yyyy-MM-dd HH:mm:ss");

LocalDateTime now = LocalDateTime.now();

writer.println(dtf.format(now) + " - " + message);

} catch (IOException e) {

System.err.println("Logs written attempt error: " + e.getMessage());

}

}

// Показати всі атрибути

public void showAllAttributes() {

logAction("Method showAllAttributes() called for: " + name);

System.out.println("Name: " + name);

System.out.println("Type: " + type);

System.out.println("Power (kW): " + power);

System.out.println("Burners: " + burners);

System.out.println("Manufacturer: " + manufacturer);

System.out.println("Efficiency (%): " + efficiency);

System.out.println("Has oven: " + hasOven);

System.out.println("Color: " + color);

System.out.println("Energy Consumed (kWh): " + energyConsumed);

System.out.println("\n");

}

// Розрахунок енергоспоживання

public void calculateEnergyConsumption(double hours) {

logAction("Method calculateEnergyConsumption() called for: " + name);

this.energyConsumed = this.power \* hours \* (this.efficiency / 100);

}

// Порівняння потужності

public void comparePower(ElectricStove other) {

logAction("comparePower(ElectricStove other) called");

if (this.power > other.power) {

System.out.println(this.name + " has higher power than " + other.name);

} else if (this.power < other.power) {

System.out.println(this.name + " has lower power than " + other.name);

} else {

System.out.println(this.name + " has equal power to " + other.name);

}

}

// Очистити лог-файл

public void clearLogFile() {

try {

FileWriter writer = new FileWriter("Lab\_2/activity.log", false);

writer.close();

System.out.println("Log file 'activity.log' has been cleared.");

} catch (IOException e) {

System.err.println("Logs cleaning error: " + e.getMessage());

}

}

}

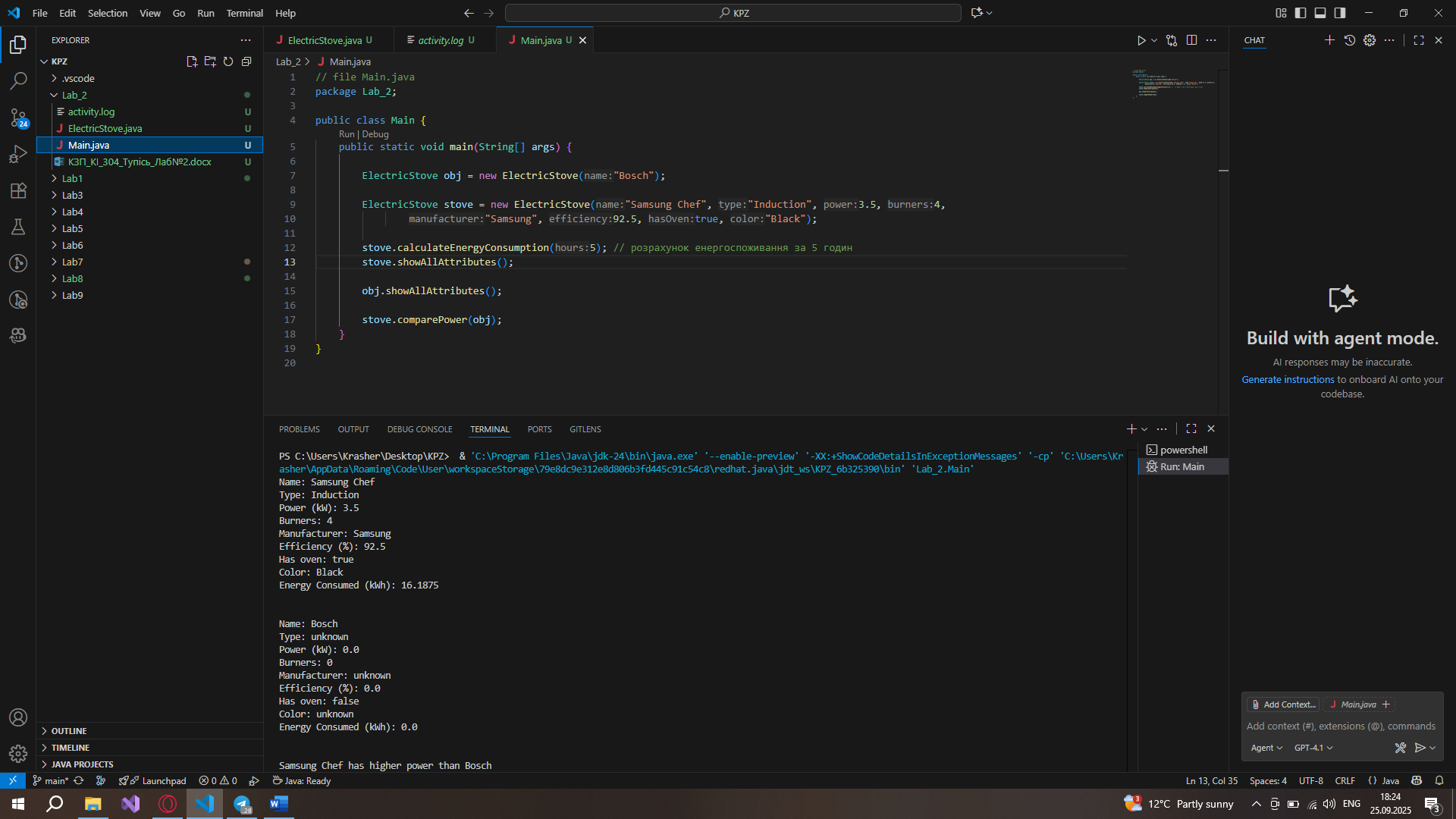


Рис.1 Приклад роботи програми



Рис.2 Git hub репозиторій з файлами програми

**Висновок:** У ході лабораторної роботи було успішно засвоєно ключові концепції об'єктно-орієнтованого програмування (ООП) в мові Java, а саме — класи та пакети.Робота з класами дозволила зрозуміти, як моделювати реальні об'єкти, інкапсулюючи їхні властивості (поля) та поведінку (методи) в єдину структуру. Це є основою для створення гнучкого, розширюваного та легко підтримуваного коду. Було отримано практичні навички створення класів, інстанціювання об'єктів та взаємодії між ними, що є фундаментом для будь-якої Java-програми.Використання пакетів продемонструвало їхню роль як ефективного механізму для організації та групування класів. Пакети допомагають уникнути конфліктів імен, забезпечують структуру проєкту та спрощують управління великою кількістю файлів. Навички імпортування класів з інших пакетів є критично важливими для використання стандартних бібліотек Java та сторонніх компонентів, що значно прискорює розробку.Таким чином, у результаті виконання роботи було досягнуто поставленої мети: набуті знання та навички роботи з класами та пакетами є необхідною базою для розробки складних програмних рішень, що ґрунтуються на принципах ООП.