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



Лістинг програми мовою Java:

// main.java

package Lab2TupisMaksym;

/\*\*

 \* Клас, що реалізує предметну область "Плитка для приготування їжі".

 \*/

public class CookingTile {

    // --- Поля класу ---

    private final String modelName;

    private boolean isOn;

    private final Logger logger;

    // --- Об'єкти класів, що описують складові частини ---

    private final Burner burner;

    private final HeatingElement heatingElement;

    private final ControlKnob controlKnob;

    // --- Конструктори ---

    /\*\*

     \* Конструктор за замовчуванням.

     \*/

    public CookingTile() {

        this.modelName = "Default Model";

        this.burner = new Burner(18.0);

        this.heatingElement = new HeatingElement("Керамічний", 700);

        this.controlKnob = new ControlKnob(9);

        this.isOn = false;

        this.logger = Logger.getInstance();

        logger.log("Створено об'єкт CookingTile (конструктор за замовчуванням): " + modelName);

    }

    /\*\*

     \* Параметризований конструктор.

     \*/

    public CookingTile(String modelName, Burner burner, HeatingElement element, ControlKnob knob) {

        this.modelName = modelName;

        this.burner = burner;

        this.heatingElement = element;

        this.controlKnob = knob;

        this.isOn = false;

        this.logger = Logger.getInstance();

        logger.log("Створено об'єкт CookingTile (параметризований конструктор): " + modelName);

    }

    /\*\*

     \* Конструктор копіювання.

     \*/

    public CookingTile(CookingTile other) {

        this.modelName = other.modelName + "\_copy";

        this.burner = new Burner(other.burner.toString().contains("18.0") ? 18.0 : 22.0); // Проста логіка для копіювання

        this.heatingElement = new HeatingElement(other.heatingElement.toString().contains("Індукційний") ? "Індукційний" : "Керамічний", 750);

        this.controlKnob = new ControlKnob(other.controlKnob.toString().contains("9") ? 9 : 12);

        this.isOn = other.isOn;

        this.logger = Logger.getInstance();

        logger.log("Створено копію об'єкта CookingTile: " + this.modelName);

    }

    // --- Методи класу ---

    public void turnOn() {

        logger.log("Спроба увімкнути плитку " + modelName);

        if (!isOn) {

            this.isOn = true;

            logger.log("Плитку " + modelName + " увімкнено.");

        } else {

            logger.log("Плитка " + modelName + " вже увімкнена.");

        }

    }

    public void turnOff() {

        logger.log("Спроба вимкнути плитку " + modelName);

        if (isOn) {

            this.isOn = false;

            this.setPowerLevel(0); // Скидаємо потужність при вимкненні

            this.burner.coolDown();

            logger.log("Плитку " + modelName + " вимкнено. Конфорка охолоджується.");

        } else {

            logger.log("Плитка " + modelName + " вже вимкнена.");

        }

    }

    public void setPowerLevel(int level) {

        logger.log("Встановлення рівня потужності " + level + " для плитки " + modelName);

        if (isOn) {

            controlKnob.setLevel(level);

            if (level > 0) {

                burner.heatUp();

                logger.log("Рівень потужності встановлено на " + level + ". Конфорка нагрівається.");

            } else {

                burner.coolDown();

                logger.log("Рівень потужності встановлено на 0. Конфорка охолоджується.");

            }

        } else {

            logger.log("Неможливо встановити потужність. Плитка " + modelName + " вимкнена.");

        }

    }

    public void increasePower() {

        logger.log("Збільшення потужності для плитки " + modelName);

        int currentLevel = controlKnob.getCurrentLevel();

        setPowerLevel(currentLevel + 1);

    }

    public void decreasePower() {

        logger.log("Зменшення потужності для плитки " + modelName);

        int currentLevel = controlKnob.getCurrentLevel();

        setPowerLevel(currentLevel - 1);

    }

    public String getStatus() {

        String status = "Статус плитки '" + modelName + "': " + (isOn ? "Увімкнена" : "Вимкнена") +

                        ". Рівень потужності: " + controlKnob.getCurrentLevel();

        logger.log("Запит статусу для " + modelName + ". Поточний статус: " + status);

        return status;

    }

    public String getModelName() {

        logger.log("Запит назви моделі: " + this.modelName);

        return this.modelName;

    }

    public String getBurnerInfo() {

        String info = this.burner.toString();

        logger.log("Запит інформації про конфорку для " + modelName + ": " + info);

        return info;

    }

    public String getHeatingElementInfo() {

        String info = this.heatingElement.toString();

        logger.log("Запит інформації про нагрівальний елемент для " + modelName + ": " + info);

        return info;

    }

    public String getControlKnobInfo() {

        String info = this.controlKnob.toString();

        logger.log("Запит інформації про ручку керування для " + modelName + ": " + info);

        return info;

    }

}

//Burner.java

package Lab2TupisMaksym;

/\*\*

 \* Клас, що описує конфорку плитки.

 \*/

public class Burner {

    private final double diameter; // в сантиметрах

    private boolean isHot;

    public Burner(double diameter) {

        this.diameter = diameter;

        this.isHot = false;

    }

    public void heatUp() {

        this.isHot = true;

    }

    public void coolDown() {

        this.isHot = false;

    }

    @Override

    public String toString() {

        return "Конфорка (діаметр: " + diameter + " см, гаряча: " + isHot + ")";

    }

}

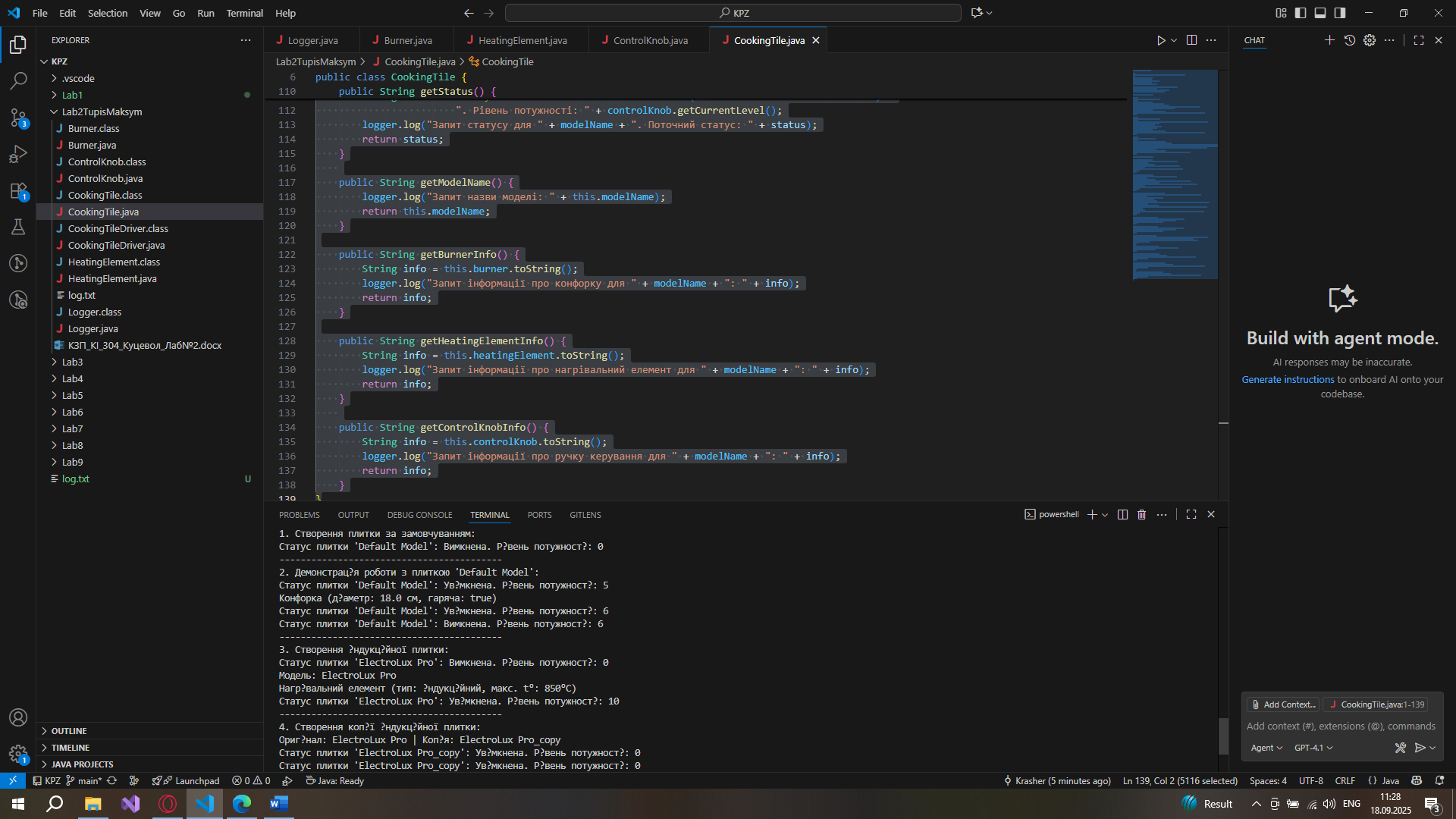


Рис.1 Приклад роботи програми

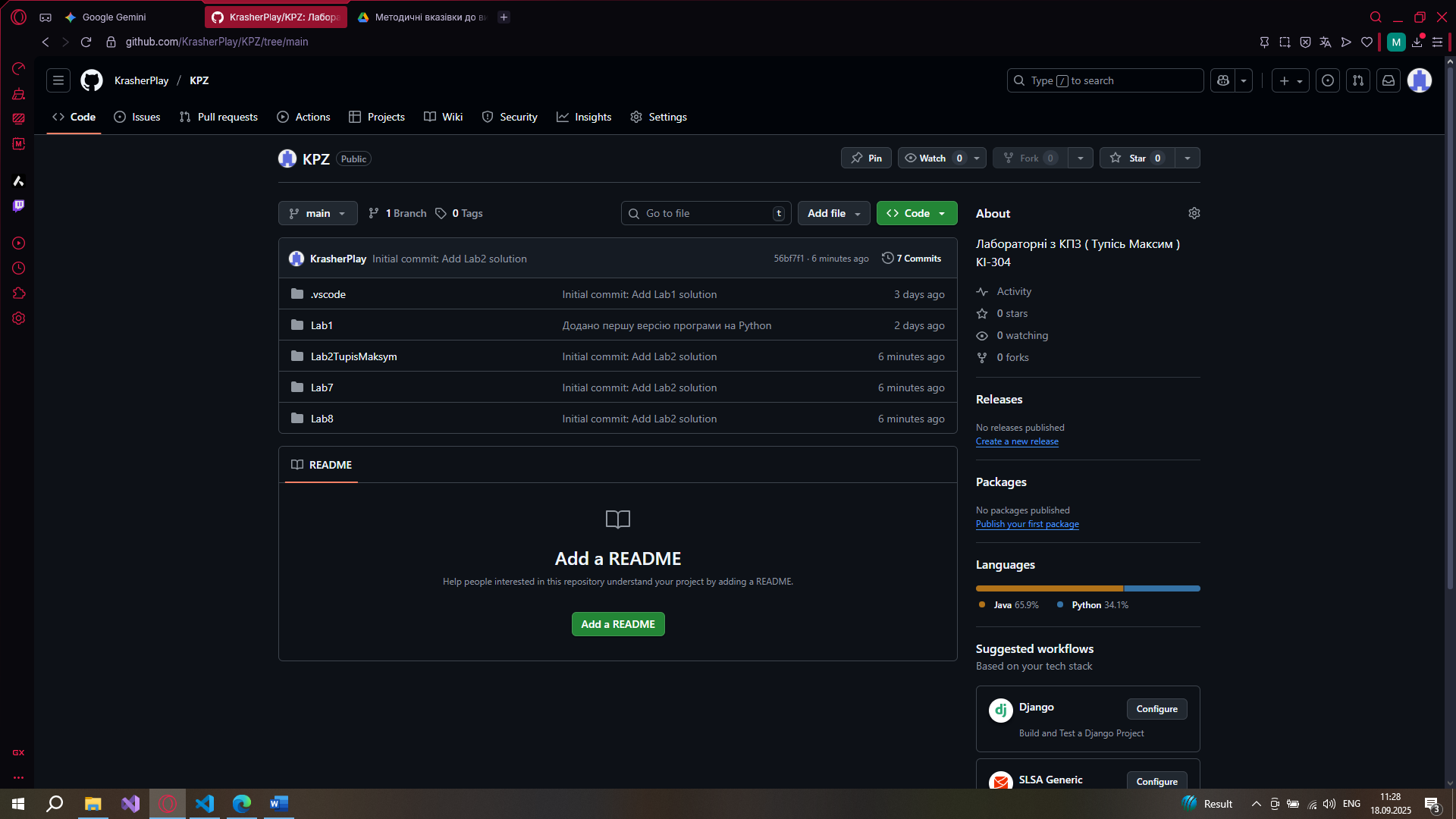


Рис.2 Git hub репозиторій з файлами програми

**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.