Міністерство освіти і науки України

Національний університет “Львівська політехніка”

Кафедра ЕОМ

Звіт

до лабораторної роботи №2

З дисципліни: «Кросплатформні засоби програмування»

На тему: «Класи та пакети»

Варіант 5

Виконала: ст. гр. КІ-305

Гринь С.М.

Прийняв:

Іванов Ю.С.

Львів 2023

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

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 5: *Машина*

Код програми

Class ***Automobile***:

```
package KI305.Hryn.Lab2;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;

/**
 * Клас, що представляє автомобіль та зберігає інформацію про нього.
 */
public class Automobile {
    //Поля класу
    private Company company;
    private Model model;
    private Price price;

    //Поле для запису протоколу
    private PrintWriter logWriter;

    /**
     * Пустий конструктор без аргументів.
     * Ініціалізує об'єкт Automobile та створює файл протоколу.
     */
    //Пустий конструктор без аргументів
    public Automobile() {
```

```
        try {
            logWriter = new PrintWriter(new
FileWriter("C:\\Users\\User\\IdeaProjects\\java_lab\\src\\KI305\\Hryn\\Lab2\\Aut
omobile.txt"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //Конструктор зі всіма аргументами
    public Automobile(Company company, Model model, Price price) {
        this.company = company;
        this.model = model;
        this.price = price;
        try {
            logWriter = new PrintWriter(new
FileWriter("C:\\Users\\User\\IdeaProjects\\java_lab\\src\\KI305\\Hryn\\Lab2\\Aut
omobile.txt"));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    //Гетери і сетери для об'єктів класу
    public Company getCompany() {
        return company;
    }

    public void setCompany(Company company) {
        this.company = company;
    }

    public Model getModel() {
        return model;
    }

    public void setModel(Model model) {
        this.model = model;
    }

    public Price getPrice() {
        return price;
    }

    public void setPrice(Price price) {
        this.price = price;
    }

    /**
     * Метод для запису дій в файл протоколу.
     *
     * @param activity Дія, яку потрібно зареєструвати в протоколі.
     */

    private void logActivity(String activity) {
        if (logWriter != null) {
            logWriter.println(activity);
            logWriter.flush();
        }
    }

    //Методи для роботи з автомобілем
    public void startEngine() {
        System.out.println("Starting the car engine.");
        logActivity("We started the car engine.");//Запис у протокол
    }
```

```
    public void brake() {
        System.out.println("Turning on the brakes.");
        logActivity("We turned on the brakes.");//Запис у протокол
    }

    public void turnLeft() {
        System.out.println("Turning to the left.");
        logActivity("We turned to the left.");//Запис у протокол
    }

    public void turnRight() {
        System.out.println("Turning to the right.");
        logActivity("We turned to the right.");//Запис у протокол
    }

    public void checkEngineStatus() {
        System.out.println("Checking the condition of the engine.");
        logActivity("We checked the condition of the engine.");//Запис у
протокол
    }

    public void turnOnHeadlights() {
        System.out.println("Turning on the headlights.");
        logActivity("We turned on the headlights.");//Запис у протокол
    }

    public void checkFuelLevel() {
        System.out.println("Checking the level of the fuel tank in the car.");
        logActivity("We checked the level of the fuel tank in the car.");//Запис
у протокол
    }

    public void lockDoors() {
        System.out.println("Locking the doors.");
        logActivity("We locked the doors.");//Запис у протокол
    }

    public void stopEngine() {
        System.out.println("Stopping the car engine.");
        logActivity("We stopped the car engine.");//Запис у протокол
        closeLogFile();//Закриваємо файл при завершенні роботи
    }

    // Метод для закриття файлу протоколу
    private void closeLogFile() {
        if (logWriter != null) {
            logWriter.close();
        }
    }
    //Метод toString()
    @Override
    public String toString() {
        return "Automobile: Company - " + company + ", Model - " + model + ",
Price - " + price;
    }

}
```

Class ***Company***:

```
package KI305.Hryn.Lab2;


/**
 * Клас, що представляє інформацію про компанію.
```

```
 */
public class Company {
    // Поля класу
    private String nameCompany; // Назва компанії
    private String infoCompany; // Інформація про компанію

    /**
     * Пустий конструктор без аргументів.
     * Ініціалізує об'єкт Company з пустими значеннями.
     */
    public Company(){

    }

    /**
     * Конструктор з усіма аргументами.
     * Ініціалізує об'єкт Company з вказаними параметрами.
     *
     * @param nameCompany Назва компанії.
     * @param infoCompany Інформація про компанію.
     */
    public Company(String nameCompany, String infoCompany){
        this.nameCompany = nameCompany;
        this.infoCompany = infoCompany;
    }

    /**
     * Перевизначений метод toString().
     *
     * @return Рядок, що представляє об'єкт Company.
     */
    @Override
    public String toString() {
        return "Company{" +
                "name='" + nameCompany + '\'' +
                ", info='" + infoCompany + '\'' +
                '}';
    }
}
```

Class ***Model***:

```
package KI305.Hryn.Lab2;

/**
 * Клас, що представляє модель автомобіля.
 */
public class Model {
```

```
    // Поле класу
    private String nameModel; // Назва моделі автомобіля

    /**
     * Пустий конструктор без аргументів.
     * Ініціалізує об'єкт Model з пустою назвою моделі.
     */
    public Model(){

    }

    /**
     * Конструктор з усіма аргументами.
     * Ініціалізує об'єкт Model з вказаною назвою моделі.
     *
     * @param nameModel Назва моделі автомобіля.
     */
    public Model(String nameModel){
        this.nameModel = nameModel;
    }

    /**
     * Перевизначений метод toString().
     *
     * @return Рядок, що представляє об'єкт Model.
     */
    @Override
    public String toString() {
        return "Model{" +
                "name:'" + nameModel + '\'' +
                '}';
    }
}
```

Class **Price**:

```
package KI305.Hryn.Lab2;


/**
 * Клас, що представляє ціну.
 */
public class Price {
    // Поле класу
    private int sum; // Сума ціни


    /**
     * Пустий конструктор без аргументів.
     * Ініціалізує об'єкт Price з нульовою сумою.
     */
    public Price() {


    }


    /**
     * Конструктор з усіма аргументами.
     * Ініціалізує об'єкт Price з вказаною сумою ціни.
```

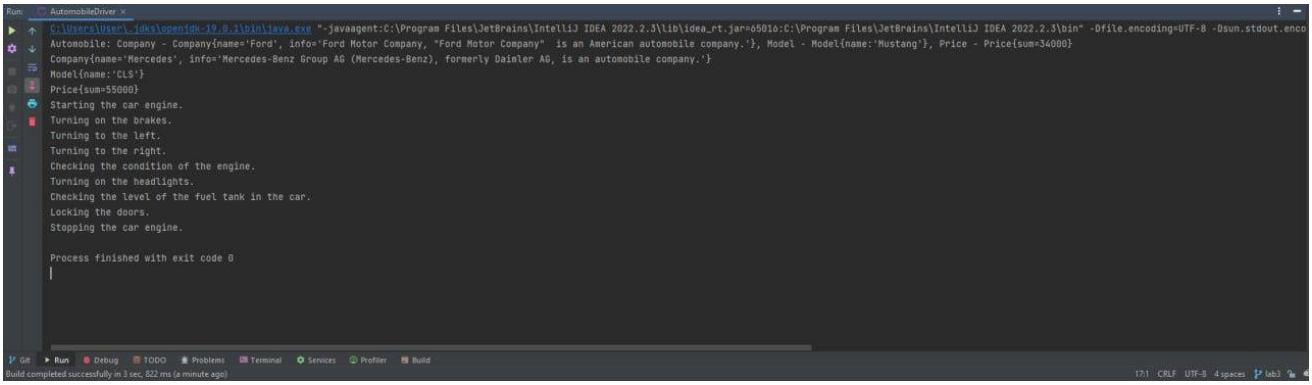
```
     *
     * @param sum Сума ціни.
     */
    public Price(int sum) {
        this.sum = sum;
    }

    /**
     * Перевизначений метод toString().
     *
     * @return Рядок, що представляє об'єкт Price.
     */
    @Override
    public String toString() {
        return "Price{" + "sum=" + sum + "}";
    }
}
```

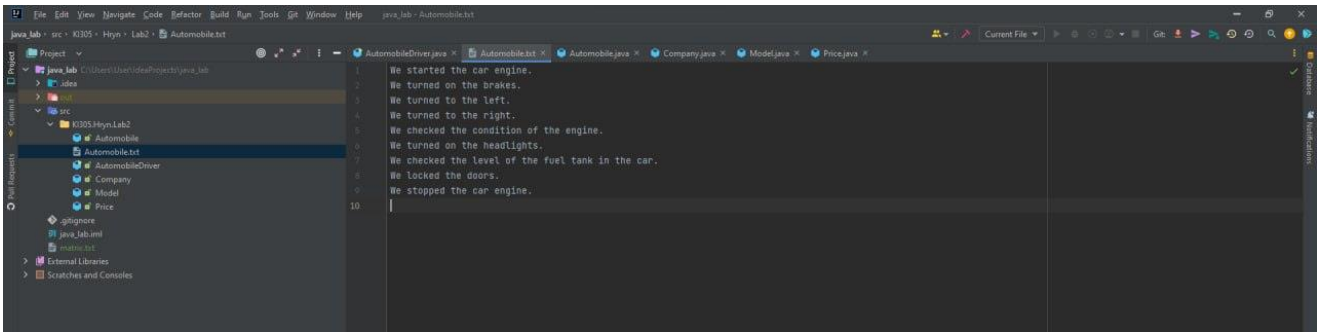
Class ***AutomobileDriver***:

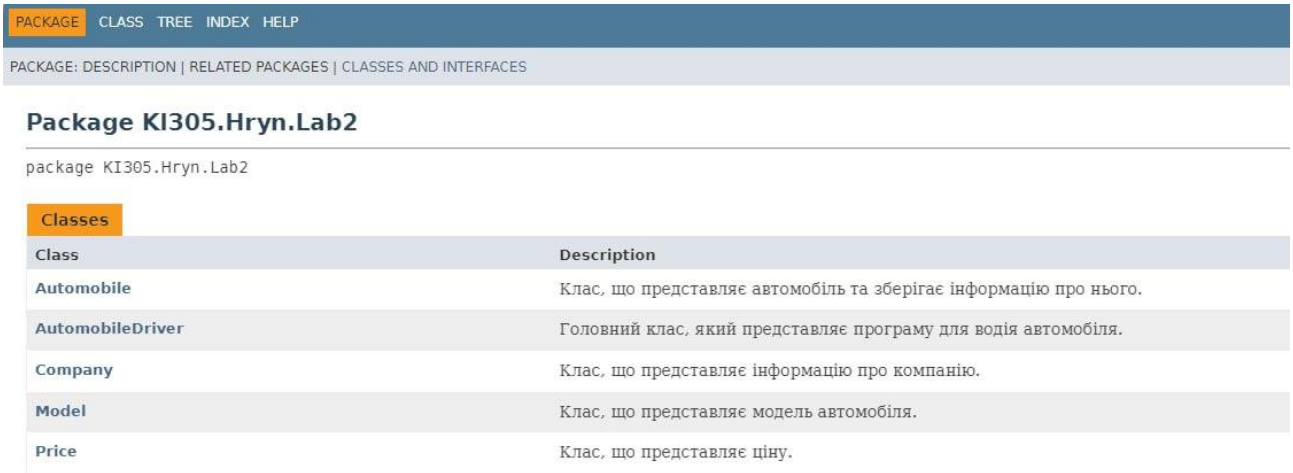
```
package KI305.Hryn.Lab2;

/**
 * Головний клас, який представляє програму для водія автомобіля.
 */
public class AutomobileDriver {
    /**
     * Головний метод програми.
     *
     * @param args Масив рядків аргументів командного рядка.
     */
    public static void main(String[] args) {

        // Ініціалізація полів
        Company company = new Company("Ford", "Ford Motor Company, \"Ford Motor
Company\"  is an American automobile company.");
        Model model = new Model("Mustang");
        Price price = new Price(34000);

        // Зміна полів за допомогою метода Сет і відображення зміненого за
допомогою метода Гет
        Automobile automobile = new Automobile(company, model, price);
        System.out.println(automobile);

        automobile.setCompany(new Company("Mercedes", "Mercedes-Benz Group AG
(Mercedes-Benz), formerly Daimler AG, is an automobile company."));
        System.out.println(automobile.getCompany());

        automobile.setModel(new Model("CLS"));
        System.out.println(automobile.getModel());

        automobile.setPrice(new Price(55000));
        System.out.println(automobile.getPrice());

        // Виклик усіх методів класу
        automobile.startEngine();
        automobile.brake();
        automobile.turnLeft();
        automobile.turnRight();
```

```
        automobile.checkEngineStatus();
        automobile.turnOnHeadlights();
        automobile.checkFuelLevel();
        automobile.lockDoors();
        automobile.stopEngine();
    }
}
```

Результат роботи програми:







PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305.Hryn.Lab2

Class Automobile

java.lang.Object
KI305.Hryn.Lab2.Automobile

public class **Automobile**
extends Object

Клас, що представляє автомобіль та зберігає інформацію про нього.

Constructor Summary

Constructors

Constructor	Description
Automobile()	Пустий конструктор без аргументів.
Automobile(**Company** company, **Model** model, **Price** price)	

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
void	**brake**()	
void	**checkEngineStatus**()	
void	**checkFuelLevel**()	
Company	**getCompany**()	
Model	**getModel**()	
Price	**getPrice**()	
void	**lockDoors**()	
void	**setCompany**(**Company** company)	
void	**setModel**(**Model** model)	
void	**setPrice**(**Price** price)	

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

void	**turnRight**()

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Details

Automobile

public Automobile()

Пустий конструктор без аргументів. Ініціалізує об'єкт Automobile та створює файл протоколу.

Automobile

public Automobile(Company company,
Model model,
Price price)

Method Details

getCompany

public Company getCompany()

setCompany

public void setCompany(Company company)

getModel

public Model getModel()

turnLeft

```
public void turnLeft()
```

turnRight

```
public void turnRight()
```

checkEngineStatus

```
public void checkEngineStatus()
```

turnOnHeadlights

```
public void turnOnHeadlights()
```

checkFuelLevel

```
public void checkFuelLevel()
```

lockDoors

```
public void lockDoors()
```

stopEngine

```
public void stopEngine()
```

toString

```
public String toString()
```

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305.Hryn.Lab2

Class AutomobileDriver

java.lang.Object
KI305.Hryn.Lab2.AutomobileDriver

```
public class AutomobileDriver
extends Object
```

Головний клас, який представляє програму для водія автомобіля.

Constructor Summary

Constructors

Constructor	Description
`AutomobileDriver()`	

Method Summary

All Methods **Static Methods** **Concrete Methods**

Modifier and Type	Method	Description
static void	`main(String[] args)`	Головний метод програми.

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, toString, wait, wait, wait

Constructor Details

AutomobileDriver

```
public AutomobileDriver()
```

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305.Hryn.Lab2

Class Company

java.lang.Object
KI305.Hryn.Lab2.Company

public class **Company**
extends Object

Клас, що представляє інформацію про компанію.

Constructor Summary

Constructors

Constructor	Description
`Company()`	Пустий конструктор без аргументів.
`Company(String nameCompany, String infoCompany)`	Конструктор з усіма аргументами.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`String`	`toString()`	Перевизначений метод toString().

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Details

Company

```
public Company()
```

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305.Hryn.Lab2

Class Model

java.lang.Object
KI305.Hryn.Lab2.Model

public class **Model**
extends Object

Клас, що представляє модель автомобіля.

Constructor Summary

Constructors

Constructor	Description
`Model()`	Пустий конструктор без аргументів.
`Model(String nameModel)`	Конструктор з усіма аргументами.

Method Summary

All Methods | Instance Methods | Concrete Methods

Modifier and Type	Method	Description
`String`	`toString()`	Перевизначений метод toString().

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructor Details

Model

```
public Model()
```

PACKAGE CLASS TREE INDEX HELP

SUMMARY: NESTED | FIELD | CONSTR | METHOD DETAIL: FIELD | CONSTR | METHOD

Package KI305.Hryn.Lab2

Class Price

java.lang.Object
KI305.Hryn.Lab2.Price

```
public class Price
extends Object
```

Клас, що представляє ціну.

Constructor Summary

Constructors

Constructor	Description
`Price()`	Пустий конструктор без аргументів.
`Price(int sum)`	Конструктор з усіма аргументами.

Method Summary

All Methods **Instance Methods** **Concrete Methods**

Modifier and Type	Method	Description
`String`	`toString()`	Перевизначений метод toString().

Methods inherited from class java.lang.Object

`clone`, `equals`, `finalize`, `getClass`, `hashCode`, `notify`, `notifyAll`, `wait`, `wait`, `wait`

Constructor Details

Price

```
public Price()
```

Контрольні запитання

1. *Синтаксис визначення класу.*

Класи

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.

Синтаксис оголошення простого класу в мові Java має наступний вигляд:

```
[public] class НазваКласу
{
	[конструктори]
	[методи]
	[поля]
}
```

Приклад оголошення загальнодоступного класу:

```
public class StartClass
{
	public StartClass()
	{
		str = "Hello";
	}
	public StartClass(String initString)
	{
		str = initString;
	}

	public void showMessage()
	{
		System.out.print(str);
	}

	private String str;
}
```

2. Синтаксис визначення методу.

Методи

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. Синтаксис оголошення методу наступний:

```
[СпецифікаторДоступу] [static] [final] Тип назваМетоду([параметри]) [throws класи]
{
	[Тіло методу]
	[return [значення]];
}
```

3.Синтаксис оголошення поля.

Поля

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. Синтаксис оголошення поля наступний:

```
[СпецифікаторДоступу] [static] [final] Тип НазваПоля [= ПочатковеЗначення];
```

Приклад оголошення поля:

```
private int i;
```

Приклад оголошення константного поля:

```
private final int i;
```

4. Як оголосити та ініціалізувати константне поле?

public class MyClass {

// Константне поле, ініціалізоване одразу

public final int MY_CONSTANT_FIELD = 42;

// Решта коду класу

}

5. Які є способи ініціалізації полів?

Ініціалізацію полів при створенні об'єкту можна здійснювати трьома способами:

- у конструкторі;
- явно при оголошені поля;
- у блоці ініціалізації (виконується перед виконанням конструктора).

Якщо поле не ініціалізується жодним з цих способів, то йому присвоюється значення за замовчуванням.

6. Синтаксис визначення конструктора.

Конструктори

Конструктор – спеціальний метод класу, який не повертає значення, має ім'я класу та призначений для початкової ініціалізації об'єктів класу. Синтаксис оголошення конструктора:

```
[СпецифікаторДоступу] НазваКласу([параметри])
{
      Тіло конструктора
}
```

7. Синтаксис оголошення пакету.

Створення пакетів

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. Синтаксис оператора `package`:

```
package НазваПакету{.НазваПідпакету};
```

8. Як підключити до програми класи, що визначені в зовнішніх пакетах?

Використання пакетів

Клас може використовувати всі класи з власного пакету і всі *загальнодоступні класи* з інших пакетів. Доступ до класів з інших пакетів можна отримати двома шляхами:

1. вказуючи повне ім'я пакету перед іменем кожного класу, наприклад,

```
java.util.Date today = new java.util.Date();
```

2. використовуючи оператор `import`, що дозволяє підключати як один клас так і всі загальнодоступні класи пакету, позбавляючи необхідності записувати імена класів з вказуванням повної назви пакету перед ними.

Оператор `import` слід розміщувати в коді програми після оператора `package` та перед оголошенням класів.

Для *підключення одного загальнодоступного класу пакету* необхідно за допомогою оператора `import` через крапку вказати повну ієрархію пакету та назву класу, який має бути імпортовано, наприклад,

```
import java.util.Date
Date today = new Date();
```

Для *підключення всех загальнодоступних класів пакету* необхідно за допомогою оператора `import` через крапку вказати повну ієрархію пакету та символ зірочка (*), наприклад,

```
import java.util.*
Date today = new Date();
```

9. В чому суть статичного імпорту пакетів?

Статичний імпорт пакетів

Починаючи з Java SE 5.0 у мову додано можливість імпортувати статичні методи і поля класів. Для цього при підключені пакету слід вжити ключове слово `static` та вказати назву пакету, або назву пакету класу та статичного методу чи поля, які ви хочете підключити:

```
import static НазваПакету{.НазваПідпакету}.НазваКласу.
    НазваСтатичногоМетодуАбоПоля;
import static НазваПакету{.НазваПідпакету}.*;
```

Наприклад,

```
// підключити тільки java.lang.Math.sqrt
import static java.lang.Math.sqrt;
// підключити всі статичні члени пакету java.lang
import static java.lang.*;
```

Статичний імпорт дозволяє не вживати явно назву класу при звертанні до статичного поля або методу класу, наприклад:

```
sqrt(pow(x, 3));        //  замість Math.sqrt(Math.pow(x, 3));

dow = d.get(DAY_OF_WEEK); //  замість dow = d.get(Calendar.DAY_OF_WEEK);
```

10. Які вимоги ставляться до файлів і каталогів при використанні пакетів?

1. **Назви пакетів**: Назви пакетів повинні бути унікальними в межах вашого проекту. Зазвичай назви пакетів збігаються зі структурою каталогів, де розміщуються відповідні файли. Наприклад, якщо у вас є пакет з назвою **`com.example.myapp`**, то його файли повинні бути розміщені в каталозі **`com/example/myapp/`**.
2. **Структура каталогів**: Файли класів, що належать до пакету, повинні розміщуватися в відповідних каталогах згідно з ієрархією пакетів. Для прикладу пакету **`com.example.myapp`**, файли класів цього пакету повинні знаходитися в каталозі **`com/example/myapp/`**. Це допомагає зберігати порядок та уникати конфліктів у назвах файлів.
3. **Оголошення пакету**: У верхній частині кожного файлу класу має бути вказано, до якого пакету він належить, за допомогою інструкції **`package`**. Наприклад:

java Copy code

```java
package com.example.myapp;

public class MyClass {
    // Реалізація класу
}
```

Висновок: я ознайомилась з процесом розробки класів та пакетів мовою Java.