**ДЕРЖАВНА СЛУЖБА УКРАЇНИ З НАДЗВИЧАЙНИХ СИТУАЦІЙ**
**ЛЬВІВСЬКИЙ ДЕРЖАВНИЙ УНІВЕРСИТЕТ БЕЗПЕКИ ЖИТТЄДІЯЛЬНОСТІ**
**Факультет цивільного захисту**
**Кафедра інформаційних технологій та систем електронних комунікацій**

ЗАТВЕРДЖУЮ
Начальник кафедри інформаційних технологій та систем електронних комунікацій,
кандиддат технічних наук, доцент
підполковник служби цивільного захисту
_____ Олександр ПРИДАТКО
«____» _____ 20 ___ року

# МЕТОДИЧНА РОЗРОБКА ДО ПРАКТИЧНИХ ЗАНЯТЬ З НАВЧАЛЬНОЇ ДИСЦИПЛІНИ

Навчальна дисципліна: Об’єктно-орієнтоване програмування
Рівень вищої освіти, курс перший(бакалаврський), 2 курс
Спеціальність (спеціалізація) :122 Комп'ютерні науки
Освітня програма: Комп'ютерні науки
Форма здобуття освіти денна

Розробник:
Старший викладач кафедри
інформаційних технологій та
систем електронних комунікацій
доктор філософії _____ Юлія НАЗАР

Методичну розробку розглянуто та затверджено на засіданні кафедри інформаційних технологій та систем електронних комунікацій,
протокол від «___» _____ 20___ №____

## Практичне заняття № 12

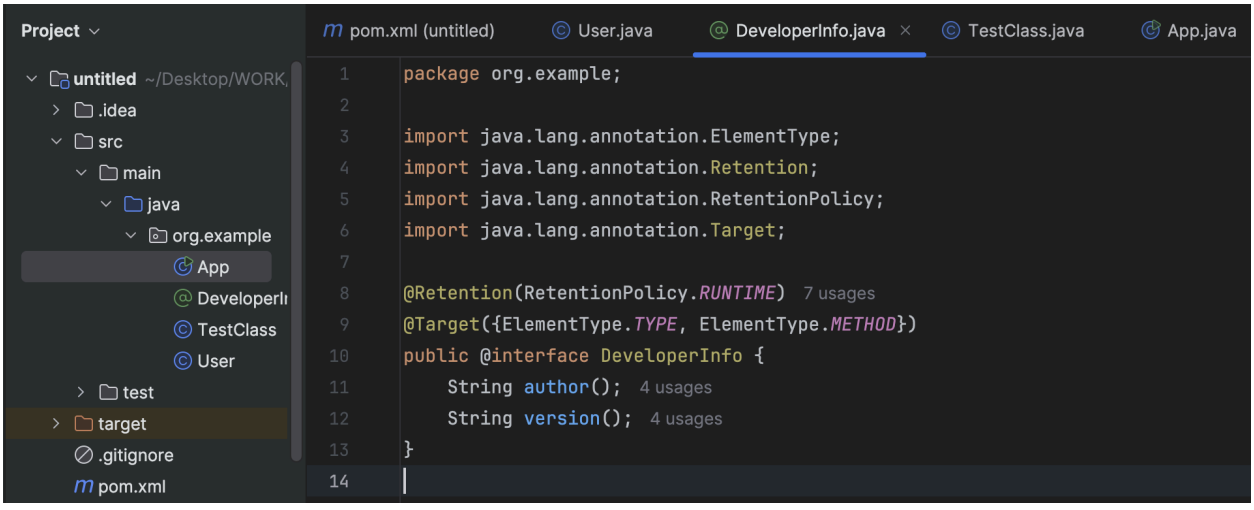
**Тема:** Анотації в Java. Reflection API
**Мета:** Ознайомитися з механізмом анотацій у Java, навчитися створювати власні анотації та використовувати Reflection API для отримання та обробки інформації про анотації під час виконання програми.
**Час:** 2 академічні години

### Завдання:

1. Створити власну анотацію відповідно до варіанту і застосувати її до класу та методу.
2. Використати Reflection API для отримання значень цієї анотації та виведення їх у консоль.

### Хід виконання роботи

1. Створення анотації, наприклад, @DeveloperInfo з полями author та version.




2. Застосування анотації до класу та методу:

pom.xml (untitled) | User.java | DeveloperInfo.java | TestClass.java | App.java

```java
package org.example;

@DeveloperInfo(author = "Yuliia Nazar", version = "1.0")  no usages
public class TestClass {

    @DeveloperInfo(author = "Yuliia Nazar", version = "1.1")  no usages
    void testMethod(){
        System.out.println("Test annotation");
    }
}
```

3. Використання Reflection API для отримання значень цієї анотації та виведення їх у консоль.

pom.xml (untitled) | User.java | DeveloperInfo.java | TestClass.java | App.java

```java
public class App {
    public static void main( String[] args ) throws NoSuchMethodException {

        Class<?> testClass = TestClass.class;
        if (testClass.isAnnotationPresent(DeveloperInfo.class)){
            DeveloperInfo developerInfo = testClass.getAnnotation(DeveloperInfo.class);
            System.out.println("Class Annotation info: Author-"+developerInfo.author() +", version " + developerInfo.version());
        }

        Method method = testClass.getDeclaredMethod( name: "testMethod");
        DeveloperInfo developerInfoMethod = method.getAnnotation(DeveloperInfo.class);
        System.out.println("Method Annotation info: Author-" +developerInfoMethod.author() + ", version " + developerInfoMethod.version());
    }
}
```

App

```
Class Annotation info: Author-Yuliia Nazar, version 1.0
Method Annotation info: Author-Yuliia Nazar, version 1.1
```

## ІНДИВІДУАЛЬНІ ЗАВДАННЯ

**Варіант 1.**
Створіть анотацію @Experimental, яка позначає методи, що ще тестуються. Виведіть у консоль попередження під час виклику таких методів.
**Варіант 2.**
Створіть анотацію @RoleAccess(role), яка визначає, хто має право викликати певний метод (ADMIN, USER).
Використайте Reflection API, щоб дозволити виклик методу лише певним ролям.

**Варіант 3.**
Створіть анотацію @MaxLength(value), яка обмежує довжину рядка.
Реалізуйте клас User, де поле username не може перевищувати задану довжину.
Використайте Reflection API для перевірки цього обмеження.
**Варіант 4.**
Створіть анотацію @MinMaxValue(min, max), яка обмежує значення числового поля.
Використовуйте Reflection API для перевірки відповідності значень.
**Варіант 5.**
Створіть анотацію @DefaultValue, яка приймає значення за замовчуванням.
Застосуйте її до полів класу та ініціалізуйте їх значеннями через Reflection API.
**Варіант 6.**
Створіть анотацію @DeprecatedMethod, щоб позначати методи, які не рекомендується використовувати.
Використайте цю анотацію в класі та виведіть у консоль попередження через Reflection API.
**Варіант 7.**
Створіть анотацію @Author, яка містить параметри name та date().
Додайте її до класу, зазначивши ім'я автора та дату створення.
Виведіть цю інформацію в консоль через Reflection API.
**Варіант 8.**
Створіть анотацію @Important, яка позначає важливі класи.
Застосуйте її до кількох класів.
Виведіть цю інформацію в консоль через Reflection API.
**Варіант 9.**
Створіть анотацію @NotNull, яка вказує, що поле не може бути null.
Використайте Reflection API для перевірки, чи всі анотовані поля класу заповнені.
**Варіант 10.**
Створіть анотацію @LogExecution, яка вказує, що потрібно логувати виклик методу.
Використайте Reflection API для автоматичного виведення повідомлення у консоль перед виконанням такого методу.