BACKEND PROGRAM

2- JUnit

ISMAIL BOUROUBA

JUnit

What is JUnit?

JUnit est un framework de test largement utilisé pour les applications Java. Il propose des annotations telles que @Test, @BeforeEach et @AfterEach pour structurer vos tests. Voici un exemple simple de test JUnit pour un service de calculatrice :

```
 import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.assertEquals;

public class CalculatorServiceTest {

    private final CalculatorService calculatorService = new
CalculatorService();

    @Test
    public void testAddition() {
        int result = calculatorService.add(3, 5);
        assertEquals(8, result, "3 + 5 should equal 8");
    }
}
```

- ## Mocking Dependencies with Mockito :

Mockito est un puissant framework de simulation pour les applications Java. Il vous permet de créer des objets simulés, de définir leur comportement et de vérifier les interactions. Pour simuler des dépendances dans vos tests unitaires, utilisez l'annotation @Mock de Mockito et l'annotation @InjectMocks pour injecter les objets simulés dans la classe en cours de test :

```java
import com.example.service.UserService;
import com.example.repository.UserRepository;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.api.extension.ExtendWith;
import org.mockito.InjectMocks;
import org.mockito.Mock;
import org.mockito.junit.jupiter.MockitoExtension;

@ExtendWith(MockitoExtension.class)
public class UserServiceTest {

    @Mock
    private UserRepository userRepository;

    @InjectMocks
    private UserService userService;

    // Write your tests here
}
```

➔ The new annotations in JUnit5 in comparison with JUnit4:

- `@TestFactory` – indique une méthode qui est une fabrique de tests pour des tests dynamiques.
- `@DisplayName` – définit un nom d'affichage personnalisé pour une classe de test ou une méthode de test.
- `@Nested` – indique que la classe annotée est une classe de test imbriquée et non statique.
- `@Tag` – déclare des balises pour filtrer les tests.
- `@ExtendWith` – enregistre des extensions personnalisées.
- `@BeforeEach` – indique que la méthode annotée sera exécutée avant chaque méthode de test (auparavant, `@Before`).
- `@AfterEach` – indique que la méthode annotée sera exécutée après chaque méthode de test (auparavant, `@After`).
- `@BeforeAll` – indique que la méthode annotée sera exécutée avant toutes les méthodes de test de la classe en cours (auparavant, `@BeforeClass`).

- `@AfterAll` – indique que la méthode annotée sera exécutée après toutes les méthodes de test de la classe en cours (auparavant, `@AfterClass`).

- `@Disabled` – désactive une classe de test ou une méthode de test (auparavant, `@Ignore`).