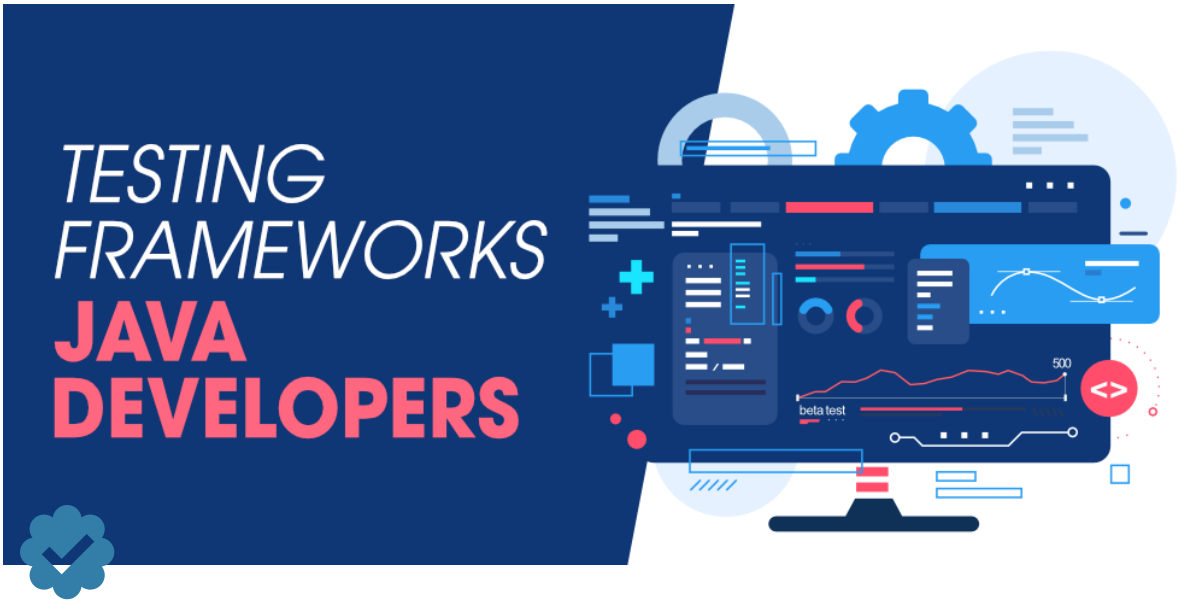


Les tests en java

💡 Test **:** Les tests en Java, comme de nombreux d'autres langages de programmation, sont essentiels pour garantir la qualité, la fiabilité et la correction de notre code. Ils nous aident à détecter et à prévenir les erreurs et les problèmes dans notre application. Voici les trois types de tests les plus communs :

👀 Test Unitaire

> **Les tests unitaires visent à tester des parties individuelles de code :**
> Ils testent des méthodes ou des fonctions individuelles de manière isolée. Ils se concentrent sur de petites unités de code spécifiques.

💭 Test d'intégration

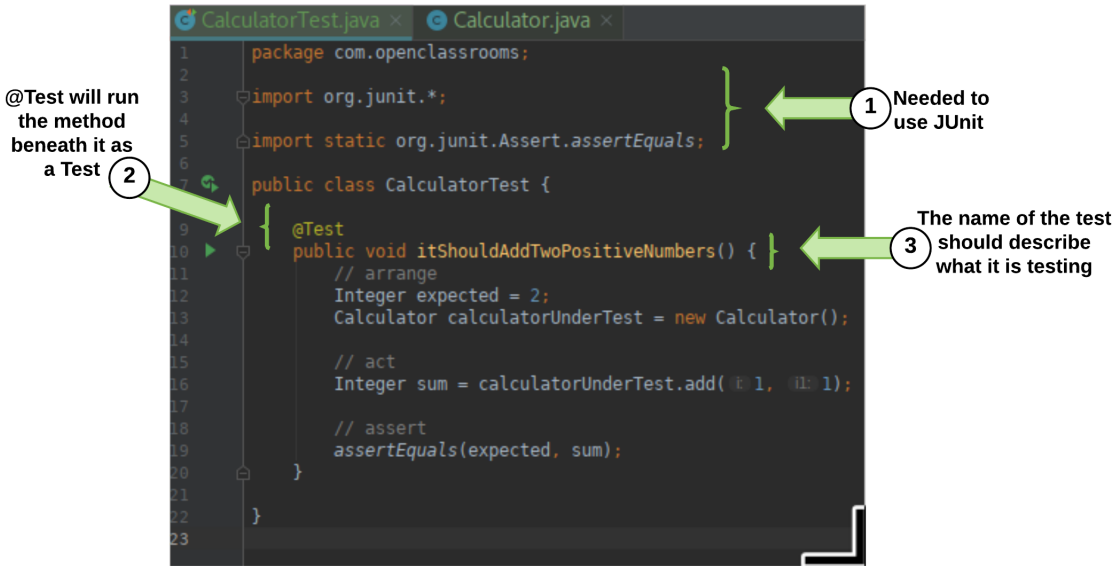
> **Les tests d’intégration vérifient l'interaction entre différentes composantes ou modules. :**
> Ils vérifient comment différentes parties du système intéragissent les unes avec les autres.

🛬 Test fonctionnel

> **Les tests fonctionnels valident que les fonctionnalités du logiciel fonctionnent comme prévu. :**
> QIls examinent le logiciel de point de vue de l’utilisateur final.

Framework de test unitaire

👉 `JUnit : est un framework pour les tests unitaires en Java. Il offre un ensemble de bibliothèques et de conventions pour écrire des tests unitaires de manière structurée et automatisée.`



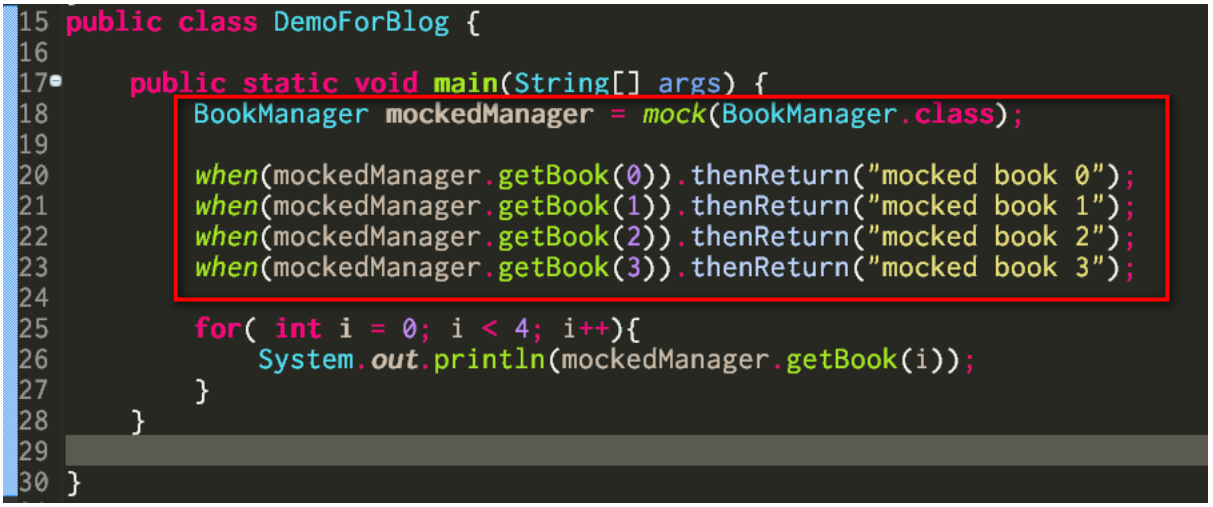
Test doubles

⚠️ les "test doubles" sont des objets ou des composants utilisés pour remplacer de vrais composants dans le but de faciliter le test d'une unité de code spécifique. Il existe plusieurs types de test doubles, notamment :

- ***Objets factices (Dummy Objects)*** : Ce sont généralement des objets passés en tant que paramètres mais qui ne sont pas utilisés dans le test. Ils sont simplement nécessaires pour satisfaire la signature de la méthode.
- ***Objets simulés (Fake Objects)*** : Ce sont des implémentations fonctionnelles, mais avec une version simplifiée, généralement non destinée à la production, de l'objet réel. Par exemple, une base de données factice en mémoire pour tester du code lié à la base de données.
- ***Stubs*** : Les stubs fournissent des réponses préconfigurées aux appels de méthode effectués pendant le test. Vous les configurez pour renvoyer des résultats spécifiques lorsqu'ils sont appelés.
- ***Mocks*** : En plus de renvoyer des résultats préconfigurés, les mocks ont également des attentes prédéfinies concernant les appels qui devraient être effectués. Vous définissez des attentes et vérifiez que le code en cours de test effectue les appels attendus.
- ***Spies*** : Les spies sont similaires aux mocks, mais conservent le comportement de l'objet réel. Ils enregistrent des informations sur la manière dont le code en cours de test interagit avec eux, mais ne font pas échouer le test si les interactions sont différentes de ce qui est attendu.

Framework de mocking

👉 `Mockito : est un framework de double test (mocking) open source en Java, il permet aux développeurs de créer des objets fictifs (ou mocks) qui simulent le comportement d’objets réels.`



```java
public class DemoForBlog {

    public static void main(String[] args) {
        BookManager mockedManager = mock(BookManager.class);

        when(mockedManager.getBook(0)).thenReturn("mocked book 0");
        when(mockedManager.getBook(1)).thenReturn("mocked book 1");
        when(mockedManager.getBook(2)).thenReturn("mocked book 2");
        when(mockedManager.getBook(3)).thenReturn("mocked book 3");

        for( int i = 0; i < 4; i++){
            System.out.println(mockedManager.getBook(i));
        }
    }

}
```