JUnit Test & Mockito & Test Doubles :

1 - les différents types de tests :

- Tests Unitaires :

Objectif : Tester des composants individuels (fonctions, méthodes, classes) du code.
Portée : Isoler et tester chaque unité de code de manière indépendante.
Outils courants : Frameworks de tests unitaires (par exemple, Jasmine, JUnit, pytest).

- Tests d'Intégration :

Objectif : Vérifier comment différentes unités interagissent les unes avec les autres.
Portée : Tester les interactions entre modules, classes, composants, etc.
Outils courants : Tests d'intégration automatisés, outils de simulation.

- Tests de Performance :

Objectif : Évaluer les performances du logiciel, y compris la vitesse, la montée en charge, la stabilité, etc.
Portée : Mesures de performances, tests de montée en charge, tests de stress.
Outils courants : JMeter, Apache ab, outils de profiling.

2 - Test Doubles :

Les "test doubles" sont des objets ou des composants utilisés lors de tests unitaires pour isoler le code que vous souhaitez tester. Ils remplacent temporairement les dépendances réelles du code afin de vous permettre de tester une unité de code de manière isolée, en contrôlant le comportement de ces dépendances. Les test doubles sont largement utilisés dans le cadre de tests unitaires pour assurer

l'indépendance, l'isolation et la répétabilité des tests. Voici les principaux types de test doubles :

Stubs :

Les stubs sont des objets qui fournissent des réponses prédéfinies à des méthodes appelées pendant le test.
Ils simulent le comportement d'une dépendance, mais leur implémentation est souvent simplifiée.
Utiles pour isoler le code de test en fournissant des réponses cohérentes et contrôlables.

Mocks :

Les mocks sont des objets qui enregistrent les interactions avec une dépendance.
Ils vous permettent de vérifier si certaines méthodes d'une dépendance ont été appelées avec les bons paramètres pendant le test.
Utiles pour s'assurer que le code testé interagit correctement avec ses dépendances.

Fakes :

Les fakes sont des implémentations plus simples de dépendances réelles, mais elles fonctionnent suffisamment bien pour les tests.
Ils peuvent inclure des bases de données en mémoire, des serveurs HTTP simplifiés, etc.
Utiles pour réduire la complexité des tests tout en simulant des dépendances réelles.

Spies :

Les spies sont des objets qui enregistrent des informations sur les appels de méthode effectués sur une dépendance, sans altérer leur comportement.
Ils permettent de vérifier ultérieurement si des méthodes ont été appelées avec les bons paramètres.
Utiles pour vérifier l'interaction sans modifier le comportement.

Dummies :

Les dummies sont des objets qui sont passés comme paramètres mais qui ne sont pas utilisés.

Ils sont nécessaires lorsque la signature d'une méthode exige des paramètres, mais que vous ne vous souciez pas des valeurs réelles.