SQLI
DIGITAL
EXPERIENCE

Les mocks

- Pré-requis :
    - Eclipse avec java 8 d'installé

Import Projet

1. Démarrer eclipse
2. Importer le projet tp-mocks
3. Ajouter la jdk si besoin

Le composant à tester

- La classe Order représente une commande avec une liste de produits.

Mes premiers Mocks

1. Ajoutez un nouveau dossier de sources nommé tests au même niveau d'arborescence que src.
2. Dans l'explorateur, faites un clic droit sur la classe Order.
3. Dans le menu contextuel, cliquez sur New – JUnit Test Case.
4. Sélectionnez le bouton radio New JUnit 4 test.
5. Changez le dossier Source folder pour tests.
6. Cliquez sur finish
7. Nous allons donc écrire un test simple qui permet de vérifier la création d'une nouvelle commande avec deux nouveaux produits. La règle est la suivante, si j'ai 2 produits dont les prix sont 3.99 et 5.00 ma commande me revient à 8.99

Mockito propose plusieurs méthodes pour instancier des Mocks. La façon la plus simple est d'utiliser la méthode static mock() :

```
Mockito.mock(Product.class);
```

Pour simuler le comportement d'un produit pour qu'il retourne un prix concret et donc tester notre méthode getTotalPrice() :

```
Mockito.when(product1.getPrice()).thenReturn(new BigDecimal("3.99"));
```

Utiliser des Mocks partiels (Spy)

- Refactoriser la classe pour utiliser

- @Mock et @RunWith(MockitoJUnitRunner.class)

Lorsque l'on utilise un Spy, il est conseillé d'utiliser la forme doReturn() | doThrow() | doAnswer() , selon la documentation de Mockito.

1. Maintenant, nous allons faire un autre test unitaire sur la méthode formatTotalPrice(). Cette méthode, vous l'aurez remarqué, fait appel à getTotalPrice() testé précédemment.
    a. Dans ce test on ne doit rien voir apparaitre sur la classe Product
    b. Vérifier lors du test qu'on appel une seule fois la méthode getTotalPrice.

Jouer avec les mocks

1. Mock user et regarder la valeur de getLogin
2. Faire un set setLogin("bob") et regarder la valeur de getLogin