Tests unitaires

Introduction à Mockito

- **DIGINAMIC** depuis avril 2017
  - Coordinateur pédagogique
  - Formateur Java EE

- **rbonnamy@diginamic.fr**

Richard BONNAMY

diginamic FORMATION
« Votre passeport pour l'emploi numérique »

Objectifs Pédagogiques

À l'issue de cette formation, vous serez en mesure de :

- ✓ Créer un mock
- ✓ Définir le comportement d'un mock
- ✓ Utiliser un mock dans le cadre d'un test unitaire avec JUnit.

Programme détaillé ou sommaire

Problématique

Mockito

Les méthodes de Mockito

Chapitre 1
Problématique

Contexte et solution

Problématique

Contexte d'utilisation

Une application utilise une DAO pour accéder à la base de données.

Cette application exploite ensuite les données dans un algorithme de calcul.

Problématique

- On souhaite tester le comportement de cette méthode dans divers cas:
  - *La base de données est vide.*
  - *Les données en base sont incohérentes: données manquantes, valeurs erronées…*

Exemple

Dans cet exemple operationDao retourne une liste d'opérations.

Comment tester toutes les branches de la méthode calculBilanMensuel ?

```
public class Comptabilite {

	private IOperationDao operationDao;

	public void calculBilanMensuel() throws ComptabiliteException {

		List<Operation> operations = operationDao.getOperations();
		if (operations.isEmpty()){
			// Algorithme 1
		}
		// Algorithme 2
	}

	public IOperationDao getOperationDao() {
		return operationDao;
	}
	public void setOperationDao(IOperationDao operationDao) {
		this.operationDao = operationDao;
	}
}
```

Solution

Comment faire ? On vide la base ?

On insère des données incohérentes ?

Bonne pratique

Remplacer la DAO par une implémentation alternative qui retourne des données vides ou incohérentes.

On appelle cela un "mock"

Chapitre 2
Mockito

Présentation et exemple

Mockito

Solution

- Mockito est une librairie qui permet de simuler le comportement d'une classe.
- La classe simulée est appelée un bouchon
- Mockito permet d'imposer un comportement aux diverses méthodes du bouchon.

Mockito

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>2.8.47</version>
</dependency>
```

Créer un mock avec Mockito

Exemple de classe à tester:

```
public class Comptabilite {

	private IOperationDao operationDao;

	public void calculBilanMensuel() throws ComptabiliteException {

		List<Operation> operations = operationDao.getOperations();
		if (operations.isEmpty()){
			throw new ComptabiliteException("La base de données est vide");
		}
		// Suite de l'algo
	}

	// GET/SET
}
```

Créer un mock avec Mockito

Exemple de test unitaire pour tester l'exception:

```
public class ComptabiliteTest {

        @Test(expected=ComptabiliteException.class)
        public void testListeOperationsVide() throws ComptabiliteException {

            ❶ IOperationDao mockedDao = Mockito.mock(IOperationDao.class);

            ❷ Mockito.when(mockedDao.getOperations()).thenReturn(new ArrayList<>());

              Comptabilite comptabilite = new Comptabilite();
            ❸ comptabilite.setOperationDao(mockedDao);

            ❹ comptabilite.calculBilanMensuel();
        }
}
```

Explications:

- ❶ : On créé un mock de IOperationDao avec la méthode static Mockito.mock .
- ❷ : On impose un comportement avec les méthodes static when et thenReturn de Mockito.
- ❸ : On appelle la méthode setOperationDao pour que la classe Comptabilite utilise le mock.
- ❹ : On appelle la méthode calculBilanMensuel() pour vérifier la génération de l'exception.

Chapitre 3
Les méthodes de Mockito

Les méthodes de base

Les méthodes de Mockito

Méthode static Mockito.when(..).doReturn(…)

- Permet de définir un comportement (doReturn) qui va se déclencher sur l'appel d'une méthode spécifique du mock.

Exemple avec une méthode executeUC:

```
when(mockedClasse.executeUC()).thenReturn(null);
```

Limitations:

- ne fonctionne que si executeUC() retourne un résultat (type de retour != void)

Les méthodes de Mockito

Méthode static Mockito.when(..).thenThrow(…)

- Permet de définir une levée d'exception (thenThrow) qui va se déclencher sur l'appel d'une méthode spécifique du mock.

Exemple avec une méthode executeUC:

```
when(mockedClasse.executeUC()).thenThrow(new RuntimeException());
```

Limitations:

- L'exception levée doit être compatible avec la signature de la méthode.
- ne fonctionne que si executeUC() retourne un résultat (type de retour != void)

Les méthodes de Mockito

Méthode static Mockito.doThrow(..).when(…)

- Permet de définir une levée d'exception (doThrow) qui va se déclencher sur l'appel d'une méthode spécifique du mock.
- Cette méthode est utilisable pour lever une exception sur une méthode void.

Exemple avec une méthode executeUC:

```
doThrow(new StorageException()).when(mockDao).deletePizza(Mockito.anyString());
```

Limitations:

- Le throw doit être compatible avec la signature de la méthode

Les méthodes de Mockito

Méthode static Mockito.doReturn(..).when(…)

- Méthode alernative à when(…).thenReturn(…)

Atelier (TP)

OBJECTIFS : apprendre à réaliser des mocks

DESCRIPTION : Dans ce TP vous allez être amené à reprendre des exemples de programmes fournis dans le TP, à les compiler et à les exécuter en lignes de commandes.