

Les tests unitaires et JUnit– TP4

Prendre en main JUnit

L'objectif du TP est de comprendre les bases du fonctionnement de l'environnement de test JUnit.

EXERCICE 1

Dans cet exercice, vous allez utiliser JUnit pour tester la classe **Calcul**, la même que celle des exemples du support de cours. Vous allez réaliser les tests donnés en exemple du support de cours et aussi les compléter.

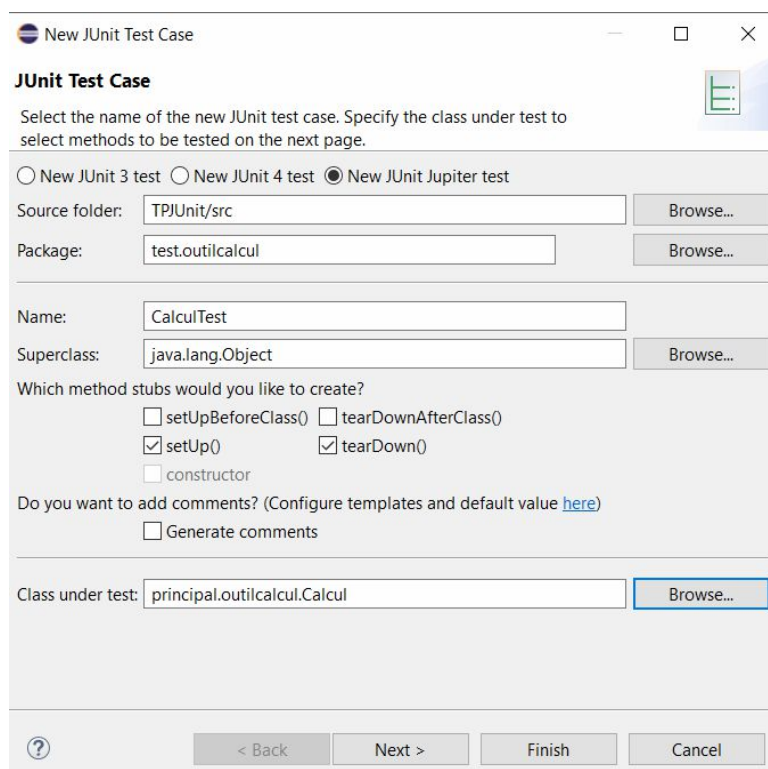
ETAPE 1 – Préparation des dossiers sous Eclipse

Tout d'abord créer un projet Eclipse dans lequel vous pourrez effectuer les opérations de ce TP (un projet nommé **TPJUnit**, par exemple). Puis récupérer sur Moodle la classe **Calcul** et la placer dans un paquetage nommé *principal*, contenant lui-même un paquetage *outilcalcul*.

Ensuite créer dans le dossier *src*, un paquetage nommé *test*, et à l'intérieur de celui-ci, un paquetage nommé *outilcalcul*. C'est dans ce paquetage qu'il faudra placer la classe de test unitaire de la classe **Calcul**

ETAPE 2 – Création de la classe de test unitaire

En suivant les indications du support de cours (section 2) créer la classe de test unitaire avec JUnit.



New JUnit Test Case

JUnit Test Case

Select the name of the new JUnit test case. Specify the class under test to select methods to be tested on the next page.

☐ New JUnit 3 test ☐ New JUnit 4 test ☒ New JUnit Jupiter test

Source folder:

Package:

Name:

Superclass:

Which method stubs would you like to create?

☐ setUpBeforeClass() ☐ tearDownAfterClass()
☒ setUp() ☒ tearDown()
☐ constructor

Do you want to add comments? (Configure templates and default value [here](#))
☐ Generate comments

Class under test:

ETAPE 3 – Version 1 de la classe de test

Compléter la classe de test qui a été générée automatiquement par JUnit. Pour ce faire, recopier le code de la version 1 conformément au support de cours.

Ensuite lancer la classe de test. Le but est d'obtenir la barre verte de JUnit indiquant que tous les tests sont passés avec succès.

ETAPE 4 – Version 2 à 4 de la classe de test

Compléter peu à peu le code de la classe de test unitaire pour obtenir celui de la version 2 du support de cours, puis celui de la version 3 et pour finir celui de la version 4. Procédez bien par étape, en comprenant bien le fonctionnement des tests que vous codez.

Pour l'assertion *assertThrows*, il faut importer les paquetages suivants :

```
import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.function.Executable;
```

ETAPE 5 – Test en erreur

Modifier la classe *Calcul* de manière à ce qu'elle comporte des erreurs de programmation. Par exemple :

- au lieu d'ajouter les opérandes, on pourrait les soustraire
- multiplier l'opérande de gauche par lui-même, au lieu de le multiplier par celui de droite
- pour la division, lorsque le diviseur est égal à 0, faire en sorte que la méthode renvoie 0

Lancer la classe de test et observer les résultats fournis par JUnit.

Remettre ensuite le code de la classe *Calcul* dans son état initial.

ETAPE 6 – Compléter la classe de test

Ajouter à la classe de test, une méthode pour tester la méthode *toString* de la classe *Calcul*. L'objectif est d'effectuer plusieurs tests. La méthode *toString* sera donc appliquée aux objets de type *Calcul* initialisés à partir des tableaux *VAL_GAUCHE* et *VAL_DROITE*.

EXERCICE 2

Le but de l'exercice est de refaire le TP 2 sur les tests unitaires, mais cette fois en utilisant JUnit. Il s'agit donc de tester la méthode *prixAPayer* de la classe *TarifSpectacle*.

ETAPE 1 – Test avec des données d'entrée valides

Effectuer d'abord les mêmes tests que dans l'étape 2 du TP 2. Il faut donc tester le résultat de la méthode *prixAPayer* dans le cas où les valeurs des paramètres sont correctes. Vous pouvez récupérer les tableaux constants du TP 2 comme jeux de test.

Essayer aussi votre classe de test avec la classe *TarifSpectacleBis* dans laquelle la méthode *prixAPayer* comporte des erreurs.

ETAPE 2 – Test avec des données d’entrée invalides

On souhaite maintenant tester le comportement de la méthode *prixAPayer* lorsque l’une des données d’entrée, le code spectacle ou le code catégorie, est incorrecte. Le comportement attendu est la levée de l’exception *IllegalArgumentException*.

Refaire les tests tels que décrits dans l’étape 4 du TP 2.

Essayer aussi votre classe de test avec la classe *TarifSpectacleErr* dans laquelle la méthode *prixAPayer* comporte des erreurs.

EXERCICE 3

Le but de l’exercice est de refaire le TP 3 sur les tests unitaires en utilisant JUnit. Il s’agit donc de tester les méthodes de la classe *OutilHoraire*.