# Conception Logicielle - TD 2

‡ ‡ ‡

## Jeu des personnages – V1.5 : Tests unitaires & Tests d'intégration

Nous souhaitons maintenant augmenter le niveau de vérification de cette première version de notre jeu. Pour ce faire nous ajouterons dans un premier temps les tests unitaires correspondants aux différentes classes produites. Puis nous enrichirons ces derniers grâce à l'ajout des test d'intégration.

## I - Tests unitaires

Avant toute chose, nous allons configurer notre projet pour pouvoir utiliser le framework Junit5.

*Exercice I.1*

### Configuration JUnit5

1. Ouvrir le fichier `pom.xml` et compléter la section des dépendances par le code ci-dessous.

```
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-api</artifactId>
        <version>5.6.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-engine</artifactId>
        <version>5.6.2</version>
        <scope>test</scope>
    </dependency>
    <dependency>
        <groupId>org.junit.jupiter</groupId>
        <artifactId>junit-jupiter-params</artifactId>
        <version>5.6.2</version>
        <scope>test</scope>
    </dependency>
```

Listing 1 – Dépendances pour `Junit 5` dans le fichier `pom.xml`

2. Depuis Eclipse, ouvrir le fichier `TestApp.java` est exécuter le en réalisant un clic-droit sur le code de ce fichier puis `run as ...` puis `Junit tests`.
   Le résultat du test devrait s'afficher à droite vous donnant un résumé de l'exécution de celui-ci.
3. *couverture de code :* en se basant sur l'exécution des différents tests unitaires [1], il est également possible de mesurer le taux de couverture de code. Cette métrique est assez intéressante puisque permettant très rapidement d'identifier les zones de code qui ne sont pas couverte par l'exécution des tests.
   Comme précédemment, faire un clic-droit sur le code du fichier `TestApp.java` puis `coverage as ...` puis `Junit tests`.
   Cette fois ci, en plus des résultats des tests, vous obtiendrez une coloration du code source des l'ensemble des classes selon trois couleurs :
   - ↬ rouge : le code n'a jamais été exécuté
   - ↬ jaune : le code a été partiellement exécuté
   - ↬ vert : le code a été totalement exécuté.

   Vous remarquerez également, dans la partie basse de l'éditeur, un résumé du taux de couverture (pourcentage) pour chacune des classes de votre projet.

*Exercice I.2*

### Tests unitaires du Jeu des personnages

Forts de notre nouvelle configuration, commençons l'écriture des tests de notre projet.

↬ Dans la section `maven` prévue à cet effet, ajouter l'ensemble des classes de tests nécessaires au test unitaire de votre projet.
On veillera à écrire les classes de tests en respectant la hiérarchie des packages du code source.

1. et d'intégration une fois que l'on maitrisera ce concept.

↬ Sans outrepasser les règles du test unitaire, améliorer vos tests (et votre code si nécessaire), de façon à obtenir le plus fort taux de couverture de code.

# II - Tests d'intégration

Comme remarqué à l'issu de la section précédente, les seuls tests unitaires ne suffisent pas à la vérification de la totalité de notre code. C'est ici qu'interviennent les tests d'intégration.
Comme présenté en cours, nous nous appuierons sur le framework *Mockito* pour la réalisation de ceux-ci. Voyons tout d'abord les modifications à apporter à la configuration de notre projet.

*Exercice II.1*

## Configuration Mockito

1. Dans le fichier `pom.xml`, dans la section des dépendances, ajouter les lignes de configuration suivantes (à la suite de celles correspondant au framework JUnit5 :

```
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-core</artifactId>
    <version>4.0.0</version>
    <scope>test</scope>
</dependency>
<dependency>
    <groupId>org.mockito</groupId>
    <artifactId>mockito-junit-jupiter</artifactId>
    <version>4.0.0</version>
    <scope>test</scope>
</dependency>
```

Listing 2 – Dépendances pour `Mockito` dans le fichier `pom.xml`

2. Réaliser un `update` de votre projet afin de vous assurer du bon chargement de ces dépendances.

*Exercice II.2*

## Tests d'intégration du jeu des personnages

1. Compléter l'ensemble de vos tests unitaires par les tests d'intégration nécessaires.
2. Obtenir un taux de couverture de 100% pour l'ensemble des classes de votre projet.

*À suivre. . .*
‡ ‡ ‡