Pratique

Création du projet

Créez un nouveau projet Maven sous IntelliJ.
Vous verrez dans le cours *Industrialisation du logiciel* ce qu'est un projet Maven. Contentez-vous de créer le projet sans trop réfléchir au fonctionnement de cet outil.

Pour vous aider dans cette démarche, vous pouvez suivre le tutorial présent sur le site d'IntelliJ :
https://www.jetbrains.com/help/idea/maven-support.html

Ajout des dépendances

Nous allons utiliser Java 11, JUnit 5 et AssertJ afin de pratiquer du Test Driven Developement.
Pour ce faire, ouvrez le fichier **pom.xml** et ajoutez les lignes suivantes avant la fermeture de la balise </project> :

```
<dependencies>
   <dependency>
      <groupId>org.junit.jupiter</groupId>
      <artifactId>junit-jupiter-api</artifactId>
      <version>5.3.2</version>
      <scope>test</scope>
   </dependency>
   <dependency>
      <groupId>org.assertj</groupId>
      <artifactId>assertj-core</artifactId>
      <version>3.13.2</version>
      <scope>test</scope>
   </dependency>
</dependencies>

<build>
   <plugins>
      <plugin>
         <groupId>org.apache.maven.plugins</groupId>
         <artifactId>maven-compiler-plugin</artifactId>
         <version>3.8.0</version>
         <configuration>
            <release>11</release>
         </configuration>
      </plugin>
   </plugins>
</build>
```

IntelliJ vous demandera ensuite de réimporter les changements du projet Maven. Importez les changements et vous aurez un environnement fonctionnel pour la mise en place de tests unitaires.

Partie 1 – Introduction au TDD et à JUnit

Dans cet exercice, nous allons réaliser le kata « FizzBuzz » (http://codingdojo.org/kata/FizzBuzz/), exercice simple et bien connu lors des entretiens d'embauche.

Contraintes

- Le code doit être produit en ***Pair Programming,*** vous coderez à deux sur une même machine via Teams
- Le code doit être écrit en utilisant du ***Test First Programming (Test Driven Developement),*** aucun code ne doit être écrit avant que le test le soit
- Une des personnes du binôme écrit le test unitaire, puis l'autre écrit le code de production et le prochain test. Et on continue jusqu'à résolution complète du problème
- La couverture de tests doit être de 100% sur votre code de production

Règles du jeu

- Un nombre est un « Fizz » si c'est un multiple de 5
- Un nombre est un « Buzz » si c'est un multiple de 7
- Un nombre est un « FizzBuzz » si c'est un multiple de 5 et de 7
- Dans les autres cas, la réponse est le nombre donnée (6 → 6)
- Un nombre négatif est invalide (et doit remonter une Exception)
- Un nombre est un « FizzBuzz » s'il contient le nombre « 66 »

A réaliser

- Ecrire une fonction ou méthode (votre choix) qui prend en paramètre un nombre et qui retourne son résultat
- Ecrire une fonction ou méthode (votre choix) qui prend en paramètre deux nombres afin de calculer une liste de résultats. Le premier correspond au nombre de départ de la séquence, le second correspond au nombre de résultats à générer