R5.A8.D7 : Qualité de Développement
Feuille TD-TP n° 2

Test-Driven Development

Objectifs :

1.- S'exercer sur la technique de développement TDD sur des exercices simples
2.- Révisions – Approfondissement du langage Java : List, ArrayList, VarArgs

Sujet :

Cette feuille de TD-TP comporte quelques exercices algorithmiques simples qui devront être développés selon l'approche TDD.
Le focus est donc sur le respect de l'approche TDD pour la production du programme.

Ressources à votre disposition :

- L'archive **`junit5-jupiter-starter-gradle.zip`**

  C'est un projet 'Modèle' minimal pour le développement en Java avec IntelliJ et gradle. La fonction main() de son unique clase Main affiche « Hello world ».

  Il devra être configuré pour les développements demandés.

Préparation du travail

1.- Création du dossier consacré aux TDs et TPs de cette ressource (R5.A.08 – R5.D.07)

Créer un dossier **`tdtp2`** dans le dossier de votre espace réseau destiné à la ressource **`r5.A08.D07`**

Par la suite, penser à valider chaque étape par une compilation et une sauvegarde de l'étape sur vos dépôts.

Exercice 1 – Produire la liste des diviseurs premiers d'un nombre entier

Étant donné un nombre entier > 0, écrire la méthode **`generate()`** d'une classe **`FacteursPpremiers`** qui génère la liste des diviseurs premiers de ce nombre.

Exemples d'appels : `FacteursPremiers.generate(1)` → liste vide

`FacteursPremiers.generate(2)` → {2}

`FacteursPremiers.generate(6)` → {2, 3}

`FacteursPremiers.generate(8)` → {2, 2, 2}

Travail à faire

1. Scénarios représentatifs du fonctionnement de la méthode à développer

Sur votre feuille de TD, écrire la liste complète des scénarios servant à identifier le comportement de complet de la méthode à développer. Ils sont indispensables pour l'écriture des tests.
C'est l'étape 0 : Think !

2. Configuration du projet

Créer et configurer le projet

a. Télécharger l'archive `junit5-jupiter-starter-gradle.zip` disponible sur eLearn. Décompresser l'archive.
b. Déposer le dossier décompressé dans `r5.A08.D07\tdtp2`. Supprimer l'archive `.zip`
c. Changer le nom du dossier/projet → `FacteursPremiers`
d. Lancer IntelliJ, ouvrir le projet `FacteursPremiers`
e. Compiler, exécuter `main()`
f. Configurer le projet (fichier build.gradle) pour l'utilisation de la bibliothèque `jAssert` [1]. Ne pas oublier de mettre à jour le fichier buid.gradle.

3. Créer le package et la classe de test et préparer le versionnement

a. Dans src/main/java, créer un package java nommé `facteursPremiers`[2]
b. Déplacer la classe `Main` dans le package `facteursPremiers`
c. Dans le package, créer la classe de test `FacteursPremiersTest`.
d. Dans `FacteursPremiersTest.java`, ajouter les imports des bibliothèques JUnit et JAssert[3] e. Compiler, exécuter `main()`

Préparer la gestion de versions

f. À la racine du dossier du projet, créer un dépôt local (git)
g. Engager (commit) le projet minimal sur le dépôt local

4. Développer la méthode generate()

La suite de l'exercice consistera à développer la méthode generer() selon la démarche TDD, c'est-à-dire en suivant un certain nombre de cycles (Test fails, Test passes, Refactor). ***Pensez à commiter chaque cycle pour garder dans votre dépôt la trace de cette démarche.***

Analyser les exemples d'appel de generate() fournis dans le sujet, ils vous guideront pour l'écriture de la signature de la méthode.

[1] Etendre les **dépendances** du fichier `build.gradle` avec la dernière version (3.24.2) de assert-core : `https://mvnrepository.com/artifact/org.assertj/assertj-core`

[2] Pour rappel, la structure du code et noms des packages en Java est la suivante :
- Le code de l'application se trouve dans le dossier src/main/java
- Le code des tests se trouve dans le dossier src/test/java
- La pratique de nommage des **packages** est la suivante :
  `com.nomEntreprise.nomPackage` ou bien `com.nomProgrammeur.nomPackage` Par exemple : `com.pantxi.calculator`

[3] Les méthodes à importer appartiennent à la classe `Assertions`. Elles sont dans le package `org.assertj.core.api.Assertions` : https://www.javadoc.io/static/org.assertj/assertj-core/3.26.3/org/assertj/core/api/Assertions.html Si cela est nécessaire, se reporter à la feuille de tdtp1.

Exercice 2 – Déplacements d'un personnage

On souhaite créer un jeu d'action contenant des personnages. On souhaite pouvoir faire tourner les personnages dans le sens des aiguilles d'une montre (nord→ est → sud → ouest → nord →…).

L'orientation initiale des personnages doit toujours être le NORD.

Écrire la méthode tourner(int fois) d'une classe Personnage permettant :

- de faire changer l'orientation d'un personnage à raison de quarts de tours.
- de retourner la nouvelle orientation du personnage

Exemple, si monPersonnage est orienté vers le NORD, monPersonnage.tourner(1) retourne EST Écrire

le minimum d'implémentation nécessaire.

Travail à faire

1. Scénarios représentatifs du fonctionnement de la méthode à développer

Sur votre feuille de TD la liste complète des scénarios servant à identifier le comportement de la méthode à développer. Ils sont indispensables pour l'écriture des tests.

C'est l'étape 0 : Think !

2. Configuration du projet

Idem que pour l'exercice 1, en adaptant au projet courant.

3. Créer le package et la classe de test et préparer le versionnement

Idem que pour l'exercice 1, en adaptant au projet courant.

4. Développer la méthode tourner()

La suite de l'exercice consistera à développer la méthode tourner() selon la démarche TDD, c'est-à-dire en suivant un certain nombre de cycles (Test fails, Test passes, Refactor). ***Pensez à commiter chaque cycle pour garder dans votre dépôt la trace de cette démarche.***

Exercice 3 – Conversion en chiffres romains d'un nombre entier (>0) écrit en chiffres arabes

Étant donné un nombre entier compris entre 1 et 50, écrire la méthode **`convert(int nbr)`** d'une classe **`ArabicRomanNumerals`** qui retourne le nombre nbr écrit en chiffres romains.

Exemples d'appels : `ArabicRomanNumerals.convert(1)` → I

`ArabicRomanNumerals.convert(3)` → III

`ArabicRomanNumerals.convert(4)` → IV

`ArabicRomanNumerals.convert(10)` → X

`ArabicRomanNumerals.convert(39)` → XXXIX

Présentation *(Conversion de Chiffres Romains sur dCode.fr* [https://www.dcode.fr/chiffres-romains]*)*

Que sont les chiffres romains ?

Les *chiffres romains* sont le nom donné au système de numération utilisé dans l'antiquité romaine (notamment du temps de César), lu de gauche à droite il utilise 7 lettres dont les valeurs s'ajoutent ou se soustraient en fonction de leur position.

Quelles sont les lettres pour écrire en chiffres romains ?

La numérotation romaine utilise 7 lettres correspondant à 7 nombres. Les *chiffres romains* de 1 à 1000 sont :

I	V	X	L	C	D	M
1	5	10	50	100	500	1000

Comment lire/écrire en chiffres romains ? La numérotation romaine utilise 4 règles :

(1) L1L2 : Toute lettre L2 placée à la droite d'une autre lettre L1 et L2≤L1 s'ajoute à L1 Exemple :

VI = 5 + 1 = 6	XX = 10 + 10 = 20

(2) L1L2 : Toute lettre L1 placée immédiatement à la gauche d'une autre lettre L2>L1 se retranche de L2. Exemple :

XC = 100 - 10 = 90	ID = 500 – 1 = 499

(3) Tout symbole (lettre) est répété au maximum 3 fois consécutivement.

Quelques exemples

1970 en *chiffres romains* MCMLXX 1971 en *chiffres romains* MCMLXXI

1972 en *chiffres romains* MCMLXXII 1973 en *chiffres romains* MCMLXXIII 1974 en *chiffres romains* MCMLXXIV 1975 en *chiffres romains* MCMLXXV

2016 en *chiffres romains* MMXVI 2017 en *chiffres romains* MMXVII

2018 en *chiffres romains* MMXVIII 2019 en *chiffres romains* MMXIX

2020 en *chiffres romains* MMXX 2021 en *chiffres romains* MMXXI

2022 en *chiffres romains* MMXXII 2023 en *chiffres romains* MMXXIII

2024 en *chiffres romains* MMXXIV 2025 en *chiffres romains* MMXXV

Travail à faire

5. Scénarios représentatifs du fonctionnement de la méthode à développer

- Quel est, avec ce système, le plus grand nombre entier convertible en chiffres romains ?
- Écrire quelques conversions et comparer vos résultats avec votre voisin.
- Une fois familiarisé avec la notation et les règles, écrire sur votre feuille de TD la liste complète des scénarios servant à identifier le comportement de la méthode à développer. Ils sont indispensables pour l'écriture des tests.

C'est l'étape 0 : Think !

6. Configuration du projet

Idem que pour l'exercice 1, en adaptant au projet courant.

7. Créer le package et la classe de test et préparer le versionnement

Idem que pour l'exercice 1, en adaptant au projet courant.

8. Développer la méthode convert()

La suite de l'exercice consistera à développer la méthode convert() selon la démarche TDD, c'est-à-dire en suivant un certain nombre de cycles (Test fails, Test passes, Refactor). ***Pensez à commiter chaque cycle pour garder dans votre dépôt la trace de cette démarche.***

Analyser les exemples d'appel de convert() fournis dans le sujet, ils vous guideront pour l'écriture de la signature de la méthode.