Feuille de travaux pratiques n° 1
Assertions en Java

1 Introduction

En Java, les assertions sont utilisées pour détecter les erreurs qui pourraient passer inaperçues. Les assertions contiennent des expressions booléennes qui assurent la cohérence de l'état d'un programme à des points spécifiques.

Habituellement, les développeurs confirmés utilisent les assertions :

— Au début d'une méthode pour vérifier si l'état de l'objet à l'entrée de la méthode est celui attendu. Ceci car une forte proportion d'erreurs est liée à des malentendus concernant les interfaces des méthodes.

— À la fin d'une procédure complexe, pour vérifier que le résultat est cohérent. Ce type d'assertion est parfois appelé un auto-test.

A partir de la version 1.4 les assertions sont devenues un élément du langage Java. Dans sa forme la plus simple, une assertion évalue une expression booléenne et si celle-ci est fausse, lève l'exception `AssertionError`.

```
BankAccount acct = null;

// ...
// Get a BankAccount object
// ...

// Check to ensure we have one
assert acct != null;
```

Dans cet exemple, l'assertion assure que la variable acct n'est pas nulle. Dans le cas contraire, une exception est levée. Ainsi, les instructions qui suivent l'assertion sont sûres que la variable n'est pas nulle.

Un inconvénient du langage Java c'est que les assertions ne sont pas habilitées par défaut. Lorsqu'on exécute un programme, aucune assertion n'est vérifiée. Il faut les habiliter de façon explicite :

```
java -ea MainClass // ou
java  -enableassertions
```

Utilisez les assertions en Java pour résoudre les exercices suivants

Exercice 1.1

Développez une classe qui contient une pile d'entiers d'au plus 10 éléments. Cette classe doit implémenter deux méthodes : push(Integer), pour ajouter un élément sur la pile et pop(): Integer pour retirer un élément de la pile. L'implémentation doit assurer qu'aucun élément ne sera retiré si la pile est vide et qu'aucun objet ne sera ajouté si la pile est pleine.
Après l'implémentation, testez les situations suivantes :

1. ajout et retrait d'un élément.
2. ajout d'un élément à une liste pleine.
3. retrait d'un élément d'une liste vide.

Exercice 1.2

Développez une classe contenant deux méthodes de conversion de température, de Fahrenheit à Celsius et vice-versa. Ces méthodes n'acceptent pas des valeurs nulles ni des températures inférieures à -273,15°C ou -459.67°F.
Les formules de conversion sont : $[^\circ F] = [^\circ C] * 9/5 + 32$ et $[^\circ C] = ([^\circ F] - 32) * 5/9$
Après l'implémentation, testez les situations suivantes :

1. si 0°C est bien converti en 32°F et si 32 °F est converti en 0°C.
2. conversion d'une température inférieure à -273,15 °C et à -459.67°F.
3. conversion d'une température à exactement -273,15 °C et -459.67°F.

Exercice 1.3

Modifiez le code des classes précédentes de manière à assurer que ces classes ne pourront être utilisées que lorsque les assertions sont habilitées.

Exercice 1.4

AssertJ est une bibliothèque Java qui simplifie l'écriture d'assertions. Son code est disponible sur l'URL suivante : `https://joel-costigliola.github.io/assertj/`.
Remplacez vos assertions par des assertions AssertJ.