Les Exceptions et les Tests

Le langage java inclut un mécanisme de gestion d'erreurs appelé les exceptions. Ce mécanisme permet de propager une erreur dans la pile d'appel sans utiliser le traditionnel retour de méthode.

Une exception est un objet. C'est une instance d'une classe qui hérite de la classe **java.lang.Throwable** par exemple : **IllegalArgumentException** (voir la documentation de l'API pour les différentes exceptions existantes)

Par défaut, toutes les exceptions sont capturées par la machine virtuelle java qui affiche un message (appel à la méthode **printStackTrace**()) et stoppe l'exécution.

Ce travail utilise les tests des classes réalisant l'abstraction jauge.

Chaque tandem recopie l'ensemble des classes jauge dans son répertoire de travail. Les tandems vont modifier la classe de test **JaugeTest** et une des classes "implémentant" l'interface **Jauge**.

Table des matières

1 Lever une exception

D'après les tests effectués dans la classe **JaugeTest**, il y deux cas d'instanciation qui aboutissent à un état incohérent.

Dans la classe **JaugeTest**, retirer les deux tests **testLimiteVigieMaxInferieurVigieMin**() et **testMaxEgaleMin**() et ajouter le test suivant :

```
private void testCreationNonValide() {
  Jauge inverse = creerJauge(78, 13, 0);

  Jauge egale = creerJauge(-45, -45, -45);
}
```

Dans le cas d'une instanciation avec des arguments invalides, il est obligatoire d'indiquer une erreur plutôt que de ne rien faire ou d'essayer de mettre l'objet dans un état cohérent. L'instanciation doit échouer. Ici seul le mécanisme des exceptions est utilisable.

Lever l'exception **IllegalArgumentException** dans le constructeur de votre classe.

Avec le test ci-dessus, vérifier si les exceptions sont bien levées.

⇒ *Expliquer pourquoi une seule exception est levée et non les deux.*

2 Capturer une exception

Au lieu d'effectuer une vérification visuelle, nous allons écrire le test pour vérifier si les exceptons sont bien levée dans le code du test. Pour cela, il faut capturer l'exception dans les tests.

Le traitement à réaliser est le suivant :

— Si l'exception est capturée, c'est à dire que l'exception a été levée. Le test ne doit rien afficher puisque c'est le comportement voulu.
— Si l'exception n'est pas capturée, c'est à dire que l'exception n'a pas été levée. Le test doit afficher un message qui montre qu'une assertion a échoué : utiliser l'instruction suivante **assert false**: **message**;.

⇒ *Pourquoi faut-il deux blocs **try/catch** pour s'assurer que l'exception est bien levée dans tous les cas d'instanciation invalide.*

Vérifier le comportement du test si aucune exception n'est levée.

⇒ *Quelle est la valeur des variables* `inverse`, `egale` *dans la partie **catch** ? Vérifiez ces valeurs grâce à une assertion.*

⇒ *Comment déclarer les variables pour les utiliser à la fois dans la clause* `try` *et la clause* `catch` *?*

Remarque : Le code écrit dans ce test ne correspond pas à une gestion d'erreur.

3 Exception contrôlée

Ajoutons la méthode suivante à la classe **JaugeTest** :

```
private void testExceptionControlee() {
        throw new NullPointerException("Attention");
}
```

⇒ *Donner la classe de base de cette exception ?*

Ne pas oublier d'ajouter l'appel dans la méthode **run**().

Compiler et exécuter. Expliquer le résultat.

A la lecture du code de la méhode **run**(), il est impossible d'avoir une information sur les exceptions progagées par cette méthode.

Remplaçons l'exception dans le code précedent par une instance de la classe **ClassNotFoundException**. La compilation provoque une erreur.

⇒ *Pourquoi cette exception est-elle contrôlée et pas la précédente ?*

Pour les exceptions contrôlées, le compilateur oblige à renseigner la clause **throws** qui spécifie quelles exceptions risquent d'être propagées par une méthode.

Ajouter la clause **throws ClassNotFoundException** dans le prototype de la méthode **testExceptionControllee**(). Compiler.

⇒ *Comment corriger les autres erreurs sans capturer l'exception ?*

⇒ Exécuter les tests. *Expliquer le résultat.*

Petits exercices de manipulation :

— Capturer l'exception **ClassNotFoundException** dans la méthode **run**(), sans modifier les clauses `throws` des méthodes. Compiler.
— Mettre en commentaire la levée de cette exception dans la méthode **testExceptionControllee**(). Compiler.

⇒ Conclusion : *Quel contrôle effectue le compilateur ?*

⇒ **Boutez vos neurones ;-)**

Est-il possible de préciser une exception non contrôlée dans une clause **throws** ?

Dans l'A.P.I. y-a-t-il plus d'exception contrôlée ou non contrôlée ?

Dans la méthode **run**() comment transformer l'exception contrôlée en une exception non contrôlée ?

Pourquoi la classe **AssertionError** peut-elle être utilisée dans une instruction **throw** ? Peut-elle être capturée ? Est-elle contrôlée ?

D'après la documentation de l'A.P.I, que représente la classe **Error** ?