Classe Inversée 1 - Les exceptions

Evaluation

1 Terrain à explorer !

Nous allons implémenter une simulation de traversée de terrain miné pour illustrer le mécanisme des exceptions.

1.1 Les mines

En vous basant sur le canevas vu au cours et en CI, veuillez implémenter trois classes pour les trois exceptions suivantes :

- une exception vérfiée appelée `ExplosiveMineException` ;
- une exception vérifiée appelée `IncendiaryMineException` ;
- une exception non-vérifiée appelée `UndetectedMineException`.

Nous vous demandons de ne pas créer de hiérarchie, vos classes d'exception se situeront au même niveau que la classe `EvalOne` (pas d'utilisation de package dédié).

1.2 La traversée du terrain miné

Complétez la classe `EvalOne` présentée ci dessous.

```
1  public class EvalOne {
2
3          [...]
4          public static void main(String[] args) {
5                  int compteur = 1;
6                  [...] // appeler ici autant que nécessaire la méthode traverser()
7                  System.out.println(compteur + " tentative(s) a(ont) été nécessaire(s)
                   ↪  pour traverser !");
8          }
9
10         [...]
11         public static void deminer()
12         {
13                 [...]
14         }
15
16         [...]
17         public static void creuser() {
18                 int value = (int) (Math.random() * 10);
19                 //génère un nombre aléatoire entre 1 et 10
20                 [...]
```

```
        }

        [...]
        public static boolean traverser() {
                [...]
        }
}

/**
* Selon la norme java, les exceptions non-vérifiées et les exceptions vérifiées peuvent
* recevoir un traitement différent en termes de spécification et de gestion (de
↪  l'exception).
* En supposant qu'un de vos collègues ait suivi la norme java pour la spécification et
* la gestion de l'exception (non-vérifiée), quelles différences trouverait-on
* dans son code et ses spécifications, comparé aux vôtres (qui respectent
* les bonnes pratiques vues dans le cours sur ce point)?
*
* - 1ère différence :
* - 2ème différence :
*
* (Pensez différence entre exceptions vérifiées et non vérifiées d'un point
* de vue documentation/spécification et pour le compilateur)
*/
```

Cette classe contient 4 méthodes. Pour chacune des méthodes nous vous demandons de :

- **respecter les bonnes pratiques vues au cours** ;
- écrire les spécifications ;
- modifier si nécessaire la signature ;
- terminer l'implémentation ;
- rajouter des commentaires pour expliquer votre code (à l'intérieur de la méthode), si vous le jugez nécessaire.

La méthode `main()` est l'entrée du programme. Elle appelle autant de fois que nécessaire la méthode `traverser()` et indique le nombre de tentatives qui ont été nécessaires pour traverser le terrain miné.

La méthode `creuser()` est une méthode qui lance aléatoirement une exception vérifiée `ExplosiveMineException` ou `IncendiaryMineException` ; ou encore une exception non vérifiée `UndetectedMineException`. Le code présent dans classe assigne une valeur aléatoire entre 1 et 10 à la variable *value*, vous utiliserez cette dernière pour le lancement de vos exceptions (Ex. : si entre 1 et 3 alors, si entre 4 et 7 alors...).

La méthode `deminer()` est une méthode qui appelle la méthode `creuser()`, elle neutralise (capture) toute exception `ExplosiveMineException` ou `IncendiaryMineException` qui surviendrait. La méthode indique dans le terminal quelle mine (Exception) a été neutralisée (attrapée) ou si aucune mine n'a été trouvée.

La méthode `traverser()` indique dans le terminal qu'une tentative de traversée a lieu. Ensuite, elle appelle la méthode `déminer()` 10 fois. Lors de ces 10 appels à `déminer()`, une Execption de type `UndetectedMineException` peut survenir, l'information sera indiquée dans le terminal (Ex. : "Boum !") et la méthode `traverser()` retournera alors False ; sinon elle retournera True à la fin des 10 appels. Attention à ne pas laisser transparaitre ces détails d'implémentation dans votre spécification !

1.3 Bonnes et mauvaises pratiques

Veuillez indiquer les deux différences entre bonne pratique et mauvaise pratique (au niveau de code appelant) pour la gestion des exceptions non vérifiées. **Indiquer les 2 différences en complétant le bloc de commentraire en dessous du code de la classe**.

2 Consignes

- ne pas utiliser de package (4 fichiers au sein d'un dossier) ;
- remettez vos sources et uniquement vos sources ;
- compressez vos fichiers/votre dossier **dans un fichier zip (sans mot de passe), pas de *.rar, *.tar ou autre** ;
- soumettre votre solution sur webCampus dans le temps imparti ;
- il n'est pas nécessaire de mentionner votre nom dans les fichiers remis.

Remarque : tout code qui serait remis dans un format de compression autre que *.zip **ne sera pas évalué**.