

PROJET 1

Exercice 1 : La classe Configuration

1.

L'annotation `@Override` exprime l'intention du programmeur à redéfinir une méthode de la super-classe. Elle permet donc au compilateur de comprendre et contrôler qu'il s'agit ici d'une redéfinition de la méthode dont il hérite.

Elle apparaît ici car la méthode `toString()` existe déjà au sein de la classe `Objet`, toutes les autres classes héritent donc de cette méthode par défaut. Dans le cas de l'héritage, on fait appel à l'annotation `@Override`.

`@Override` n'est pas obligatoire, elle permettra juste au compilateur de contrôler que la signature de la méthode `toString()` à bien respecter celle dont elle hérite. Dans le cas échéant, le compilateur ne contrôlera pas et ne signalera donc pas en cas de problème.

2.

Les attributs ne devraient pas être déclarés « public » car il ne faut pas que l'utilisateur puisse directement modifier l'attribut. De plus, ceux sont des attributs de la classe `Contribution` et non d'une autre. Ils ne devraient pas être accessibles par les autres classes soeurs ou filles à celle-ci.

Pour mieux faire les choses, il faudrait que leur droit d'accès soit « private ».

3.

Il est possible d'écrire `c1.equals(c2)`.

Si aucune des valeurs n'a été changée entre temps, les deux attributs seront égaux et le résultat sera donc le bon.

Exercice 2 : Analyse des arguments de la ligne de commande : version naïve

1.

Voir au sein de la classe de Test => `ConfigurationTest`

2.

Le résultat sera : « alpha=0.85, epsilon=0.001, indice=100, mode=CREUSE »

3.

On peut en déduire que le programme n'est pas robuste.

4.

Il ne nous est pas possible de déduire `NumberFormatException` étant donné qu'il s'agit d'une exception non vérifiée (sous-type de `RuntimeException`). Le compilateur ne reconnaissant et n'interceptant seulement au moment de l'exécution, n'ayant eu aucune gestion des exceptions dans ce code, il est impossible de s'attendre à l'interception de cette exception. Un développeur s'y connaissant pourra savoir que les méthodes `parseInt` et `parseDouble` ont une chance de déclencher cette exception.

5.

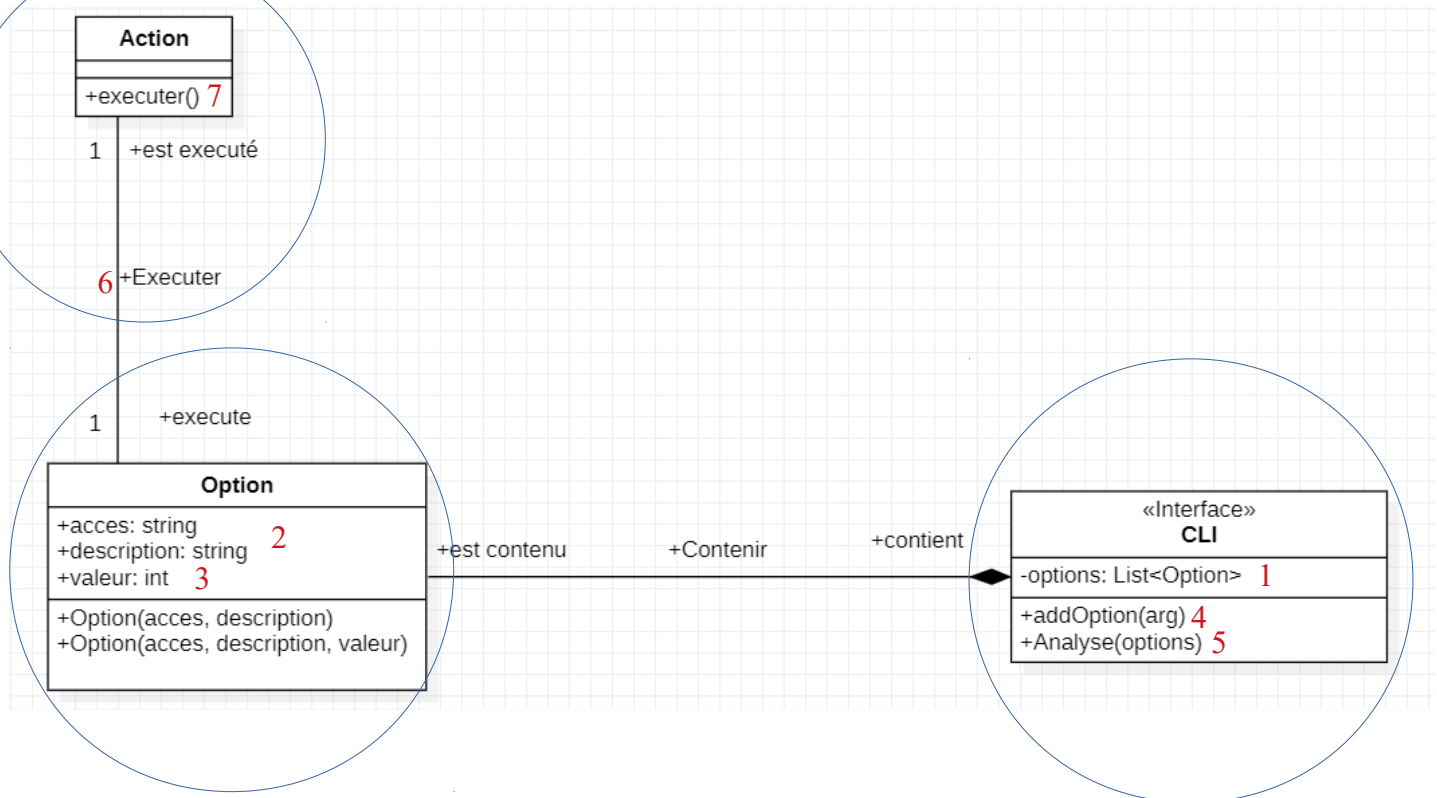
Voir au sein de la classe => CLIClassique

6.

Il manque un break sur l'élément default du dans le try.

Ainsi que le constructeur de la classe, ici seul un constructeur par défaut est utilisé.

Exercice 3 : Version réutilisable



J'ai dans un premier temps pensé à une liste d'option utilisant l'interface collection List mais je pense qu'il serait plus intéressant d'utiliser Set ou SortedSet.

2.

Voir au sein de la classe => PageRank

3.

Afin de stocker les options d'une CLI, je pense qu'il serait intéressant d'utiliser l'interface SET ou SORTEDSET de l'API Collection. Étant donné que dans le cas ici présent il serait normal de ne pas voir deux éléments identiques dans l'ensemble. Ces méthodes semblent être le plus efficace.

4.

...

Exercice 4 :

1.

On aura dans un premier temps une fenêtre, la JFrame que l'on positionnera en BorderLayout.

Dans la partie Nord, nous avons :

- 1 JPanel de type FlowLayout (pour être précis FlowLayout.CENTER)
- 2 JButton

Dans la partie Centrale, nous avons :

- 1 JPanel de type GridLayout(3,2)
- 3 JButton
- 3 JTextField
- 3 JLabel

Dans la partie Sud, nous avons :

- 1 JPanel de type GridLayout(1,1)
- 1 JTextField

2.

Pour pouvoir ajouter -C lors d'un clique sur le bouton, il suffirait d'appeler la méthode addActionListener au bouton concerné. Puis en implémentant la méthode actionPerformed de la classe ActionListener, demander comme action l'affichage d'un « - C ».

3.

Voir la classe => Main.