Cours 2 : Java 8 et Clean Code

Mardi 10 Novembre 2020

Antoine Flotte (aflotte@excilys.com)

Excilys
Développeurs de passion

SOMMAIRE

Java 8

- Optional permet d'encapsuler un objet potentiellement null, et de l'informer dans la signature de la méthode
- Optional.ofNullable(myObject)
- Optional.empty()
- optional.isPresent()
- optional.get()

Permet de faire du code plus concis et plus lisible

() -> System.out.println(“ok”)

myObject -> System.out.println(myObject)

(myObject) -> System.out.println(myObject)

(Type myObject) -> System.out.println(myObject)

(myObject1,myObject2)-> System.out.println(myObject1)

(Type1 myObject1,Type2 myObject2) -> System.out.println(myObject1)

Paramètre d’entrée

```
(myObject) -> System.out.println(myObject)
(myObject) -> {System.out.println(myObject);}
(myObject) ->myObject
(myObject) ->{return myObject;}
(myObject) ->{System.out.println(myObject);return myObject;}
```

Une interface fonctionnelle est une interface avec une seule méthode abstraite.

Exemple d'interface fonctionnelle :

Consumer<T> qui prend un argument de type T et ne retourne rien avec sa fonction accept

Comparator<T> qui prend en entrée deux arguments de type T et retourne un entier avec sa fonction compare

Exemple d'utilisation :

```
resultList.sort((FilmPojo film,FilmPojo film2) -> {return
film.getDuration()-film2.getDuration();});
```

Opération intermédiaire

Opération terminale

Permet d'utiliser les Collections de façon plus claire et optimale

```
resultList.stream().forEach(film ->
System.out.println(film.getDuration()));
```

Try with Ressources

On a un nouveau type de try, le try with ressources.

```
try (Ressource res1 = new Ressource();
      Ressource res2= new Ressource()){
}
```

Les ressources sont doivent étendre AutoCloseable, une interface disant que la class possède une fonction close()

La fonction close est appelé quand on sort du try

For Each

```
for ( String value : list ) {
    ...
}
```

Clean Code

Une fonction = 10 lignes max

Javadoc sur toutes les fonctions sauf getter setter

Encapsulation

Noms des fonctions et des variables cohérents

PAS DE COMMENTAIRES

On évite les répétitions, on utilise des sous fonctions, on utilise des interfaces.

Une fonction a une responsabilité unique.

On utilise les interfaces, on instancie avec les implementations.

Des outils :

- SonarCube
- Maven : CheckStyle

CheckStyle

Pour générer le checkstyle :

Ajouter dans le pom :

```
<reporting>
    <plugins>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-site-plugin</artifactId>
            <version>3.7.1</version>
        </plugin>
        <plugin>
            <groupId>org.apache.maven.plugins</groupId>
            <artifactId>maven-checkstyle-plugin</artifactId>
            <version>3.0.0</version>
        </plugin>
    </plugins>
</reporting>
```

Puis dans le terminal faire :

mvn site

Et enfin dans le dossier target aller dans site puis ouvrir checkstyle.html