



Java 8 et Clean Code

Vendredi 5 Mars 2021

Excilys
Développeurs de passion

SOMMAIRE

- I. Ajout Java 8
- II. Clean Code

Java 8

- Optional permet d'encapsuler un objet potentiellement null, et de l'**informer** dans la signature de la méthode

```
public Optional<Client> findById(int id);
```

Le résultat de findById peut potentiellement être null, il faudra checker cette possibilité quand on appelle la fonction

- `Optional.ofNullable(myObject)` pour obtenir un `Optional` qui contient `myObject`
- `Optional.empty()` pour obtenir un `Optional` vide
- `optional.isPresent()` pour savoir si le `Optional` possède un objet
- `optional.get()` pour obtenir l'objet (il faut avoir vérifié qu'il est présent avant)

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 automatiquement quand on sort du try

```
try (  
    Connection connection = DriverManager.getConnection();  
    PreparedStatement preparedStatement =  
connection.prepareStatement(FIND CLIENTS QUERY);  
    ResultSet resultSet = preparedStatement.executeQuery();  
    ) {  
    ...  
} catch (SQLException e) {  
    throw new DaoException(e.getMessage());  
}
```

Clean Code


```
List<Client> clients = new ArrayList<>();  
for ( Client client : clients) {  
    System.out.println(client);  
}
```

Une fonction = 10 lignes max

Javadoc sur toutes les fonctions sauf getter setter

Encapsulation

Noms des fonctions et des variables cohérents

PAS DE COMMENTAIRES

```
/**
 * retourne l'ensemble de client de la base de donnée
 * @return une liste qui contient tout les clients présent en base
 * @throws DaoException en cas d'erreur lors de la connexion à la base de
donnée ou dans la requête
 */
public List<Client> findAll() throws DaoException {

}
```

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.

```
List<Client> clients = new ArrayList<>();
```

Des outils :

- SonarCube
- Maven : 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