Design pattern I

Ce premier TP sur les designs patterns va vous permettre de mieux comprendre le fonctionnement du pattern Singleton et Abstract Factory en les implémentant vous-même. Le but est de les implémenter dans des classes de votre projet. D'abord il s'agit de déterminer où il est judicieux de les utiliser et ensuite de les implémenter correctement. Pour pouvoir les utiliser de manière judicieuse voici quelques exemples qui pourraient vous aider:

Pour le Singleton: Le singleton sert à n'avoir qu'une seule instance pour un objet.

- Dans une application qui utilise une base de données, l'objet BD devra être un singleton car on veut avoir qu'une instance de la même BD.
- Un service qui va faire des appels à une API est aussi un singleton car on veut toujours que ce soit la même instance de service qui est appelé par les différentes classes.

Pour l'Abstract Factory: L'abstract factory sert à créer une interface commune pour la création d'objets différents, qui partagent les mêmes fonctions.

- Dans une application qui a plusieurs types d'utilisateur · e · s (étudiant et professeur par exemple), on veut pouvoir créer des étudiant · e · s et des professeur · e · s de la même manière.
- Amazon souhaite livrer leur marchandise en utilisant plusieurs moyen de transport (drone, camion, avion) et en pouvant en ajouter à tout moment (ex: bateau).

A rendre

Avant le prochain cours, une archive contenant votre code source java pour les deux designs pattern ainsi qu'un bout de texte justifiant votre choix.

Quelques ressources pour vous aider

- https://www.baeldung.com/java-abstract-factory-pattern
- https://ddst.medium.com/design-pattern-101-singleton-and-factory-pattern-f0908a56d044
- https://refactoring.guru/design-patterns/abstract-factory
- https://refactoring.guru/design-patterns/singleton

Il y en a bien sûr plein d'autres...