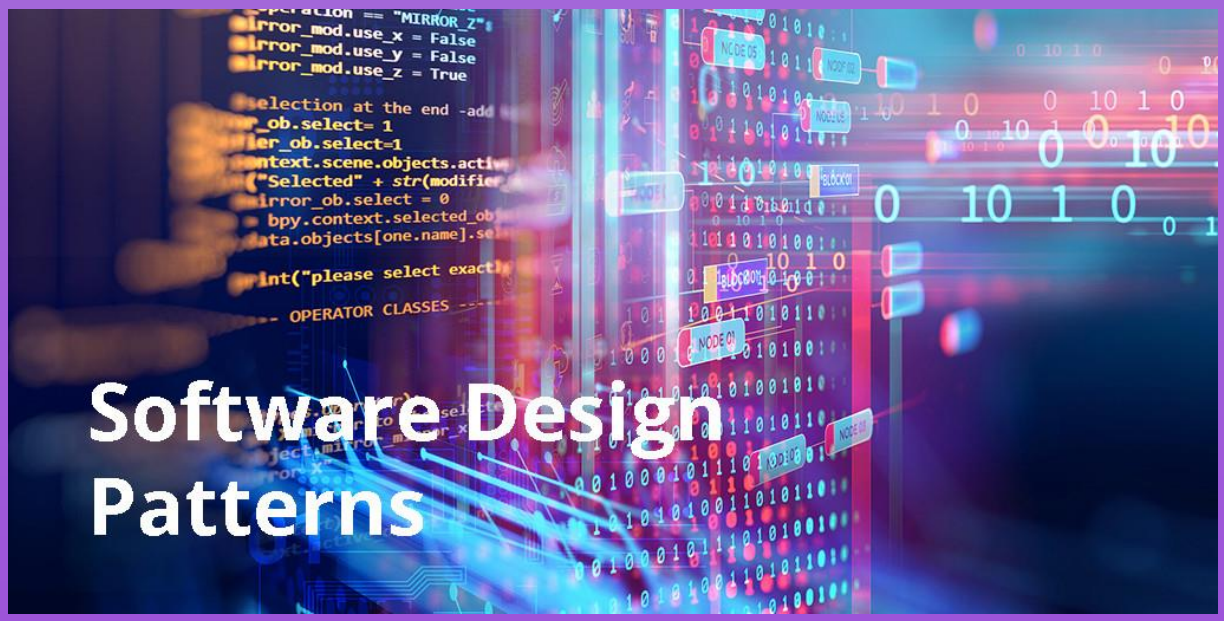


# DETAIL DES DP UTILISE

**Explication des Design Patterns utilisés**

Chef de projet : Reine SONKIN FOZI

Stephane Aristide KOLOKO KALEU

Uziel SIMOU NGHUEU-SI

Yvan Olivarex ZANGUE TSOMEJIO

Yves-Alexis KUETE DOUNMO

# DESIGN PATTERN



## ABSTRACT FACTORY: (DESIGN PATTERN CRÉATIF)

Le patron Abstract Factory nous permettra d'instancier des familles de produits mutuellement dépendants sans avoir besoin de spécifier leur type concret. Il fournit une interface pour créer des familles d'objets liés ou interdépendants sans avoir à spécifier au moment de leur création la classe concrète à utiliser.

## OBSERVATOR: (DESIGN PATTERN COMPORTEMENTAL)

On l'utilise pour limiter le couplage entre les modules aux seuls phénomènes à observer. Aussi, il permet une gestion simplifiée d'observateurs multiples sur un même observable. Ces derniers effectuent l'action adéquate en fonction des informations qui parviennent depuis des modules qu'ils observent.

## SINGLETON

Il permet de s'assurer qu'une classe puisse posséder qu'une seule et unique instance sous peine de générer des erreurs.

## ITERATOR : (DESIGN PATTERN COMPORTEMENTAL)

Le modèle Iterator est un modèle de conception très courant utilisé dans les environnements de programmation Java et .Net. Ce patron est utilisé pour fournir un moyen d'accéder aux éléments d'un objet de collection de manière séquentielle sans avoir besoin de connaître sa représentation sous-jacente.

## MVC : (DESIGN PATTERN ARCHITECTURAL)

Le modèle architectural Modèle-Vue-Contrôleur (MVC) sépare une application en trois composants principaux : le modèle, la vue et le contrôleur.

Un contrôleur est chargé de contrôler la manière dont un utilisateur interagit avec une application MVC. Il contient la logique de contrôle du flux pour une application MVC. Le contrôleur détermine la réponse à renvoyer lorsque l'utilisateur fait une demande au navigateur.

Les deux actions de contrôleur exposées par la classe HomeController, Index() et About(), renvoient toutes deux une vue. Une vue contient les balises et le contenu, qui sont envoyés au navigateur.

Un modèle MVC contient toute votre logique métier qui n'est pas contenue dans une vue ou un contrôleur. Le modèle doit contenir toute la logique métier de votre application, la logique de validation et la logique d'accès à la base de données.