# Justification des design patterns utilisés

Table des matières

[Justification des design patterns utilisés 1](#_Toc148993947)

[I. Introduction 2](#_Toc148993948)

[II. Design Patterns utilisés 2](#_Toc148993949)

[1. Factory 2](#_Toc148993950)

[2. Observer 2](#_Toc148993951)

[3. Prototype 2](#_Toc148993952)

[III. Modèle conceptuel de données 3](#_Toc148993953)

[1. Schéma 3](#_Toc148993954)

[2. Description 3](#_Toc148993955)

## Introduction

Projet : Gestion des cours des élèves (Ajout, modification, suppression de cours dans leur parcours respectif) + envoie d’une notification par courriel lors d’une update du parcours

## Design Patterns utilisés

### Factory

Lors de l’ajout, modification ou suppression d’un cours cela n’affectera aucune classe à part le parcours.

Un parcours possède 5 cours : Java, PHP, Base de données, UX design, gestion de projet. Le lendemain, le directeur de l’école annonce qu’un nouvel expert en JavaScript arrive et qu’il enseignera sa matière à la classe de bachelor 3 conception d’application. Le parcours était prévu pour uniquement 5 cours et changer sa taille reviendrait à changer toute la structure du code.

Le pattern Factory propose de remplacer les appels directs au constructeur de l’objet, il est donc maintenant possible de redéfinir la méthode Factory dans la sous-classe et de changer la classe des produits crées par la méthode.

### Observer

Détection de la mise à jour d’un parcours + envoie d’une notification par courriel lors d’une update du parcours (Avec si c’est suppression de ce cours ci, ajout de ce cours la …)

Nous nous retrouvons dans une situation problématique. Un élève ayant choisi un parcours développement-cyber car il aimait les langages de programmations mais surtout la cyber sécurité apprend que les matières liées à la cyber sécurité ont été supprimé de son parcours. En réalité, ils ont été déplacés vers un autre parcours celui de cyber sécurité. Il est impossible pour l’école de changer la liste des élèves à partir d’une certaine date. L’élève n’ayant pas vérifié tous les jours l’actualité de son parcours se retrouve à suivre des cours qu’il n’apprécie pas.

L’observer propose un système de souscription à la classe diffuseur (celle qui enverra les notifications lors de la modification de la classe sujet) et cela permettra aux objets que l’on souhaite de recevoir ou non cette notification de mise à jour de la classe. Lorsque que l’évènement de modification arrive, il fait le tour des souscripteurs et appelle la méthode de notification des objets.

### Prototype

On aimerait « créer » des élèves avec un identifiant, un nom, un prénom. Seulement, l’identifiant, le nom et le prénom de l’élève sont privés et ne peuvent pas sortir de la classe. Il est nécessaire d’arriver à la même structure de conception d’un élève en gardant leurs informations privées.

Cette méthode permet de cloner les objets qui vont être copiés. Une interface commune est déclarée pour tous les objets qui vont être clonés. L’interface permet de cloner les objets sans coupler le code à la classe de cet objet. Lors du clonage, les valeurs des attributs de l’objet de d’origine se reporte dans le nouveau (clone de l’objet d’origine).

## Modèle conceptuel de données

### Schéma

Une image contenant texte, diagramme, Plan, Dessin technique

Description générée automatiquement

### Description

* Les classes élèves et professeurs héritent de la classe personne qui contient plusieurs informations génériques (id, nom, prénom) ;
* La classe professeur donne un seul et unique cours ;
* La classe cours est donnée par un seul et unique professeur ;
* La classe parcours contient un ou plusieurs cours ;
* La classe élève suit un seul et unique parcours ;
* La classe classe possède zéro ou plusieurs élèves ;