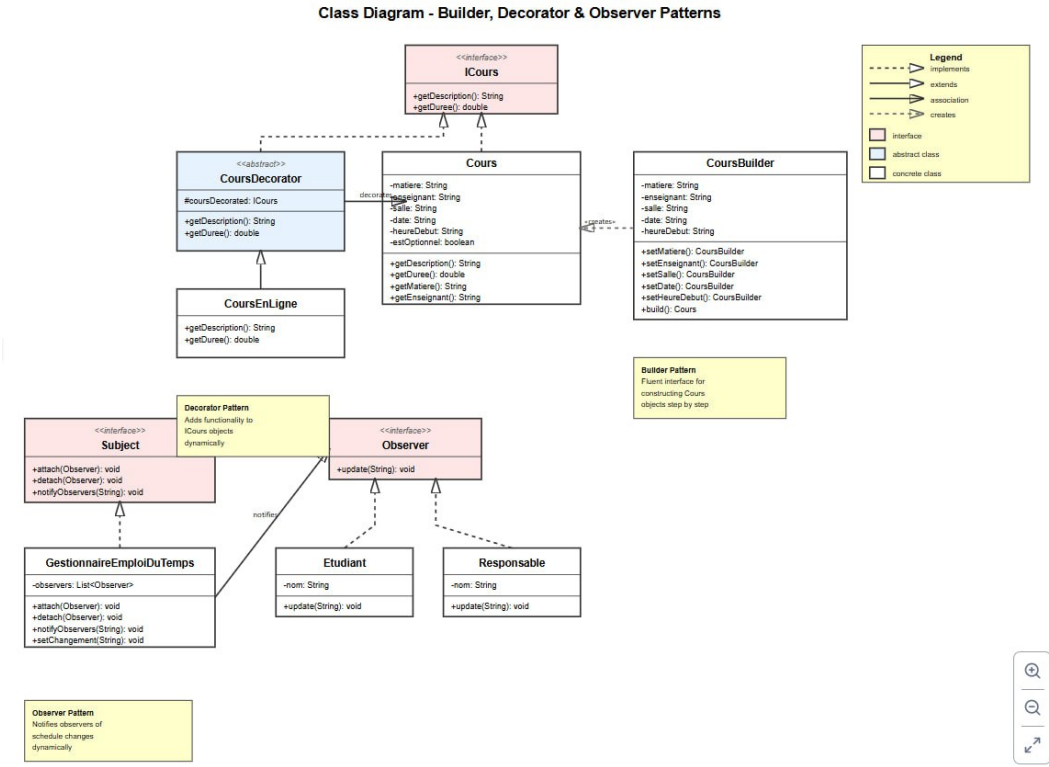
TP 2 Génie Logiciel : Design Patterns
Gestion de l'Emploi du Temps

Chegrani Seif Islem – SIAD –

20 November 2025

1 Diagramme de Classe Final

Le diagramme suivant montre l'implémentation des trois design patterns : Builder, Observer et Decorator.



2 Principes non respectés

Mon code viole deux principes du modèle SOLID :

2.1 DIP – Dependency Inversion Principle

Le `GestionnaireEmploiDuTemps` dépend directement de classes concrètes comme `Etudiant` ou `Responsable`. Ces classes ne passent pas toutes par une interface `Observer`, ce qui rend le gestionnaire dépendant des implémentations au lieu d'une abstraction.

Solution : s'assurer que tous les observateurs implémentent bien l'interface `Observer`.

2.2 OCP – Open/Closed Principle

Pour ajouter un nouvel observateur (par exemple un enseignant ou une application mobile), il faudrait modifier le code du gestionnaire. Il n'est donc pas réellement ouvert à l'extension sans modification.

Solution : utiliser exclusivement l'interface `Observer` afin de rendre le gestionnaire extensible sans changer son code.

Conclusion

Le code enfreint donc les principes **DIP** et **OCP** car les observateurs ne reposent pas tous sur une abstraction commune, et le gestionnaire n'est pas suffisamment extensible sans modification.