

TP 2 Génie Logiciel – Design Patterns

Bonus – KELAIAIA Ayoub

Groupe RSD

Novembre 2025

Bonus 1 – Diagramme de classes UML final

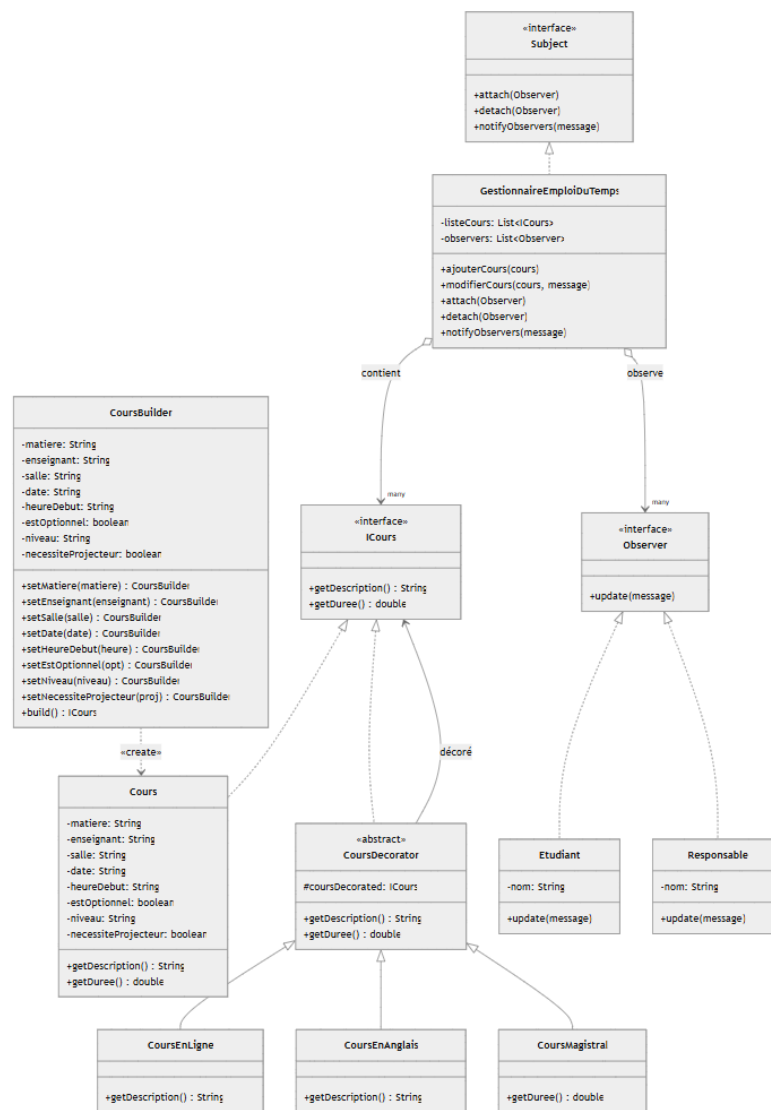


FIGURE 1 – Diagramme de classes UML complet intégrant les patterns Builder, Decorator et Observer

Bonus 2 – Respect des principes SOLID et GRASP

Conclusion : L'architecture respecte l'ensemble des principes SOLID et les principes GRASP de faible couplage et forte cohésion. Le code est propre, extensible et maintenable.

| Principe | Respecté ? | Justification |
|--------------------------------------|------------|--|
| Single Responsibility (SRP) | Oui | Chaque classe n'a qu'une seule responsabilité : CoursBuilder construit, Cours représente un cours, CoursDecorator ajoute des fonctionnalités, GestionnaireEmploiDuTemps gère les cours et notifie, Etudiant/Responsable reçoivent les notifications. |
| Open/Closed (OCP) | Oui | La classe Cours est fermée à la modification mais ouverte à l'extension via le pattern Decorator (on peut ajouter « En ligne », « En anglais », « Magistral »... sans toucher à Cours). |
| Liskov Substitution (LSP) | Oui | Tous les objets implémentant ICours (Cours et ses décorateurs) sont substituables sans casser le programme. |
| Interface Segregation (ISP) | Oui | Interfaces très petites : ICours (2 méthodes), Observer (1 méthode), Subject (3 méthodes). |
| Dependency Inversion (DIP) | Oui | GestionnaireEmploiDuTemps dépend de l'abstraction Observer, pas des classes concrètes Etudiant/Responsable. Le Decorator dépend aussi uniquement de l'interface ICours. |
| Low Coupling / High Cohesion (GRASP) | Oui | Les patterns Builder, Observer et Decorator réduisent fortement le couplage. Les observateurs ne connaissent pas le sujet concrètement, les décorateurs sont indépendants. |

TABLE 1 – Respect des principes SOLID + GRASP