1 Diagramme de classes

Le diagramme complet est présenté figure 1.

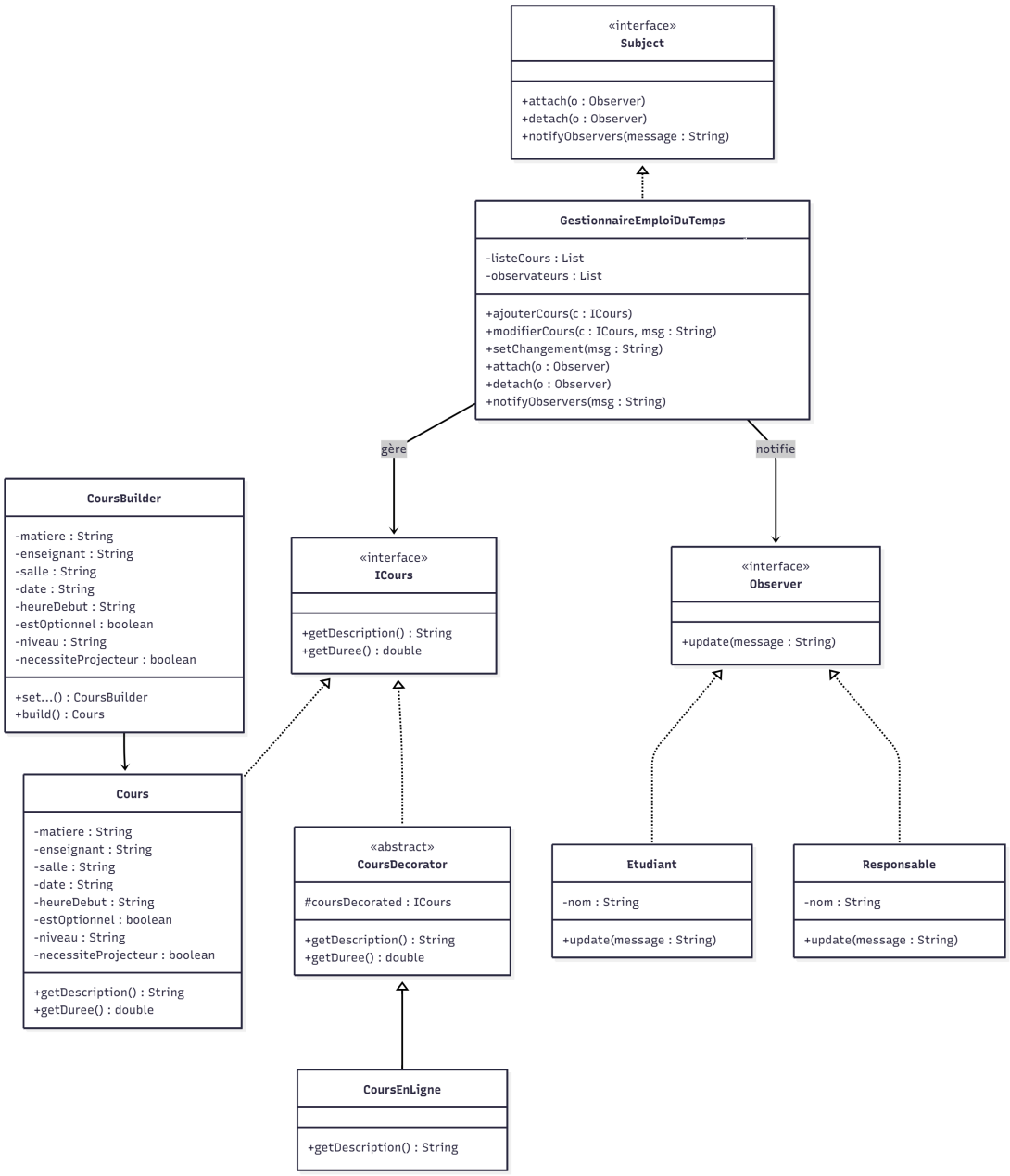

FIGURE 1 – Diagramme de classes du module de gestion des cours

2 Analyse des principes SOLID

2.1 Single Responsibility Principle (SRP)

Principe : chaque classe doit avoir une unique responsabilité.
Application :

- `CoursBuilder` ne gère que la construction des objets `Cours`.
- `CoursEnLigne` ajoute uniquement la mention “En ligne” sans toucher à la logique des cours.
- `GestionnaireEmploiDuTemps` coordonne les cours et les notifications.

Les responsabilités étant bien séparées, SRP est respecté.

2.2 Open/Closed Principle (OCP)

Principe : les entités doivent être ouvertes à l'extension mais fermées à la modification.
Application : l'utilisation des interfaces `ICours`, `Observer` et `Subject` permet d'ajouter de nouveaux décorateurs ou observateurs sans modifier les classes existantes. OCP est donc respecté.

2.3 Liskov Substitution Principle (LSP)

Principe : les classes dérivées doivent pouvoir remplacer leurs classes mères sans altérer le comportement attendu.
Application : `CoursEnLigne` reste un `ICours`, et `Etudiant`/`Responsable` restent des `Observer`. Leur substitution ne casse pas les clients, LSP est respecté.

2.4 Interface Segregation Principle (ISP)

Principe : mieux vaut plusieurs petites interfaces spécialisées qu'une grosse interface générique.
Application : le système définit trois interfaces ciblées (`ICours`, `Observer`, `Subject`) ; aucune classe n'est forcée d'implémenter des méthodes inutiles. ISP est respecté.

2.5 Dependency Inversion Principle (DIP)

Principe : les modules de haut niveau doivent dépendre d'abstractions et non de détails.
Application :

- `GestionnaireEmploiDuTemps` dépend des interfaces `Observer` et `ICours` au lieu de classes concrètes.
- Les décorateurs interagissent via `ICours`, ce qui limite le couplage.

DIP est respecté dans l'implémentation actuelle.