Analyse des Principes de Conception SOLID

1 Analyse des Principes SOLID

1.1 Principes respectés

Single Responsibility Principle (SRP)

- `Cours` : gestion des données du cours.
- `CoursBuilder` : construction d'objets `Cours`.
- `Etudiant` et `Responsable` : réception des notifications.
- `GestionnaireEmploiDuTemps` : gestion du planning et des notifications (voir remarque).

Open/Closed Principle (OCP)

- Les décorateurs (`CoursEnLigne`, `CoursEnAnglais`, `CoursMagistral`) permettent d'étendre le comportement sans modifier la classe `Cours`.
- Ajout d'observateurs possible en implémentant simplement l'interface `Observer`.

Liskov Substitution Principle (LSP)

- Tous les décorateurs substituent correctement `ICours`.
- `Etudiant` et `Responsable` sont substituables en tant qu'observateurs.

Interface Segregation Principle (ISP)

- Interfaces minimalistes : `ICours`, `Observer`, `Subject`.

Dependency Inversion Principle (DIP)

- Respecté dans les décorateurs, qui dépendent de l'interface `ICours`.
- Respecté dans l'observer : dépendance envers `Observer` et non des classes concrètes.
- Limite : `CoursBuilder` dépend directement de la classe `Cours`.

1.2 Violations et améliorations proposées

Violation du SRP : double responsabilité `GestionnaireEmploiDuTemps` gère à la fois les cours et les observateurs.

Solution recommandée :

```
class NotificationManager implements Subject { ... }

class GestionnaireEmploiDuTemps {
```

```
    private NotificationManager manager;
}
```

Violation du DIP dans `CoursBuilder` La méthode `build()` crée directement un `Cours`. **Solution optionnelle :**

```
public ICours build() { return new Cours(...); }
```

Absence de validation dans `CoursBuilder` Ajout recommandé :

— vérification des champs obligatoires,

— levée d'exceptions en cas de données invalides.

Constructeur trop chargé dans `Cours` Présence de nombreux paramètres ; ce problème est déjà compensé par l'utilisation du Builder.

2 Tableau récapitulatif

Principe	**État**	**Remarque**
SRP	Partiellement respecté	Deux responsabilités dans `GestionnaireEmploiDuTemps`.
OCP	Respecté	Extensible via décorateurs et observateurs.
LSP	Respecté	Substitution correcte.
ISP	Respecté	Interfaces minimalistes.
DIP	Partiellement respecté	`CoursBuilder` dépend de `Cours`.

3 Autres principes

DRY Pas de duplication significative ; réutilisation naturelle via les décorateurs.

KISS Conception claire, simple et facile à maintenir.

COI Les décorateurs utilisent la composition plutôt que l'héritage, ce qui améliore la modularité.