Ecole Militaire Polytechnique
Département d'Informatique
Spécialités : IASD

TP n°3 : Observer,Décorateur et Builder

Module : Génie Logiciel

Réalisé par : AZZOUZ Mhamed Alaa Eddine
Année : Année Universitaire 2025–2026

Résumé Exécutif

Ce rapport présente une analyse détaillée du respect des principes SOLID dans l'implémentation du système de gestion d'emploi du temps. L'analyse révèle plusieurs violations importantes nécessitant une refactorisation.

1 Introduction

Le code analysé implémente un système de gestion d'emploi du temps utilisant plusieurs patterns de conception (Builder, Decorator, Observer). Cette évaluation vise à identifier les violations des principes SOLID.

2 Méthodologie d'Évaluation

Chaque principe a été évalué selon les critères suivants :
— **Respecté** : Conforme au principe
— **Partiel** : Respect partiel avec améliorations possibles
— **Violé** : Violation claire du principe

3 Analyse Détaillée des Violations

3.1 Principe de Responsabilité Unique (SRP)

VIOLÉ

Classe	Violation
GestionnaireEmploiDuTemps	— Gestion des cours (`listeCours`) — Implémentation du pattern Observer (`listeners`) — Logique de notification — Logique métier d'ajout/modification

Code Problematic

```
public class GestionnaireEmploiDuTemps implements Subject {
    private List<ICours> listeCours = new ArrayList<>();
    final ArrayList<Observer> listeners = new ArrayList<>();
    //  Trois responsabilit s diff rentes
}
```

3.2 Principe Ouvert/Fermé (OCP)

PARTIELLEMENT VIOLÉ

Élément	Problème
CoursDecorator	— Implémentation concrète dans une classe abstraite — Ne force pas la redéfinition des méthodes — Risque de comportement par défaut inapproprié

Code Problematic

```
public abstract class CoursDecorator implements ICours {
    public String getDescription(){
        return coursDecorated.getDescription(); // Impl mentation concr te
    };
    //  Devrait etre abstrait pour forcer l'extension
}
```

3.3 Principe de Substitution de Liskov (LSP)

RISQUE ÉLEVÉ

Aspect	Problème Potentiel
Décorateurs	— Comportement non garanti lors de la substitution — Les décorateurs pourraient altérer le comportement attendu — Manque de contrat clair pour les extensions

3.4 Principe de Ségrégation des Interfaces (ISP)

VIOLÉ

Interface	Problème
ICours	— Interface trop minimaliste (2 méthodes) — Ne reflète pas les capacités réelles de la classe — Oblige les clients à utiliser des méthodes hors interface — Violation de la cohérence de l'interface

Code Problematic

```
public interface ICours {
    String getDescription();
    double getDuree();
    //  Seulement 2 m thodes alors que la classe en a beaucoup plus
}

public class Cours implements ICours {
    public String getMatiere() { return matiere; } // Hors interface!
    public String getEnseignant() { return enseignant; } // Hors interface!
}
```

3.5 Principe d'Inversion des Dépendances (DIP)

RESPECTÉ

Aspect	Évaluation Positive
Dépendances	— Bon usage des interfaces ICours, Subject, Observer — Injection des dépendances via constructeurs — Découplage correct entre les composants

Principe	Statut	Impact
SRP	VIOLÉ	Forte couplage, difficulté de maintenance
OCP	PARTIEL	Extensibilité limitée
LSP	RISQUE	Comportement imprévisible possible
ISP	VIOLÉ	Interface incohérente
DIP	RESPECTÉ	Bon découplage des composants

TABLE 1 – Synthèse des violations des principes SOLID

4 Tableau Synthétique des Violations

5 Conclusion

Le système présente une **architecture globalement solide** avec une bonne utilisation des patterns de conception, mais souffre de **violations importantes des principes SOLID**.