Travaux Pratiques - Contrôle Continu

Design Patterns en Java

Deux Sujets au Choix

Enseignant : Haythem Ghazouani
Groupe : ING-4-A-GLSI-OO
Module : Design Patterns

Date de remise du rapport : 02 Septembre 2025
Séance de validation : Septembre 2025 (date à préciser)
Modalité : Envoi par email à `haythemghz@yahoo.fr`

PROCESSUS D'ÉVALUATION EN DEUX ÉTAPES

ÉTAPE 1 : Remise du rapport PDF par email avant le 02 Septembre 2025 à 23h59
ÉTAPE 2 : Séance de validation (soutenance technique) en Septembre 2025

- Présentation orale de votre projet (10-15 minutes)
- Explication détaillée de votre code et des patterns utilisés
- Questions techniques de l'enseignant
- Démonstration en direct de votre application

La note finale combinera le rapport écrit ET la soutenance technique

Instructions Générales

VOUS DEVEZ CHOISIR UN SEUL SUJET PARMI LES DEUX PROPOSÉS

- **Sujet A :** Patterns Command et Composite - Éditeur Graphique
- **Sujet B :** Patterns State et Decorator - Jeu Super Mario

Indiquez clairement votre choix dans votre rapport !

Objectifs Pédagogiques Communs

- Maîtriser l'implémentation de patterns de conception avancés
- Concevoir un système orienté objet cohérent et extensible
- Réaliser un diagramme UML complet et précis
- Développer une application Java respectant les bonnes pratiques
- Savoir présenter et défendre ses choix techniques

1 SUJET A : Patterns COMMAND et COMPOSITE

1.1 Contexte - Système de Gestion d'Éditeur Graphique

Vous devez concevoir et implémenter un système de gestion pour un éditeur graphique simplifié. Cet éditeur permet de créer, manipuler et organiser des formes géométriques ainsi que des groupes de formes en utilisant les patterns **Command** et **Composite**.

1.2 Pattern Composite - Gestion Hiérarchique

Le pattern Composite doit être utilisé pour représenter la hiérarchie des formes :

1. **Component** : Interface commune `GraphicElement`
   - `draw()` : Dessiner l'élément
   - `move(int dx, int dy)` : Déplacer l'élément
   - `resize(double factor)` : Redimensionner
   - `setColor(Color color)` : Changer la couleur
   - `getBounds()` : Obtenir les dimensions
2. **Leaf** : Formes géométriques simples
   - `Circle` : Cercle avec centre et rayon
   - `Rectangle` : Rectangle avec position et dimensions
   - `Triangle` : Triangle avec trois points
   - `Line` : Ligne avec point de départ et d'arrivée
3. **Composite** : Groupe de formes
   - `ShapeGroup` : Contient d'autres formes ou groupes
   - Opérations appliquées à tous les éléments contenus
   - Support de la structure arborescente

1.3 Pattern Command - Système d'Annulation

Le pattern Command doit implémenter les opérations avec Undo/Redo :

1. **Command Interface**
   - `execute()` : Exécuter la commande
   - `undo()` : Annuler la commande
   - `getDescription()` : Description de l'opération
2. **Commandes Concrètes**
   - `CreateShapeCommand` : Création d'une forme
   - `DeleteShapeCommand` : Suppression d'une forme
   - `MoveShapeCommand` : Déplacement d'une forme
   - `ResizeShapeCommand` : Redimensionnement
   - `ChangeColorCommand` : Changement de couleur
   - `GroupShapesCommand` : Regroupement de formes
   - `UngroupShapesCommand` : Dégroupement
3. **Invoker**
   - `CommandManager` : Gestionnaire des commandes
   - Historique des commandes (Stack pour Undo/Redo)
   - Limitation de l'historique (20 commandes max)

1.4 Fonctionnalités Requises - Sujet A

1. **Gestion des formes :**
   - Création de formes avec propriétés (position, taille, couleur)
   - Modification des propriétés avec historique
   - Suppression avec possibilité d'annulation
2. **Gestion des groupes :**
   - Regroupement de formes sélectionnées
   - Dégroupement avec restauration individuelle
   - Opérations sur groupe (déplacement, couleur, etc.)
   - Groupes imbriqués (groupes de groupes)
3. **Système d'annulation :**
   - Undo : annulation de la dernière opération
   - Redo : rétablissement d'une opération annulée
   - Affichage de l'historique des commandes
   - Nettoyage automatique de l'historique
4. **Fonctionnalités avancées :**
   - Sauvegarde/chargement de compositions
   - Sélection multiple de formes
   - Calcul automatique des dimensions de groupe

2 SUJET B : Patterns STATE et DECORATOR

2.1 Contexte - Système de Jeu Super Mario

Vous devez implémenter un système de jeu Super Mario utilisant le **pattern State** pour gérer les états de Mario et le **pattern Decorator** pour le système de power-ups combinables.

2.2 Pattern State - États de Mario

Le pattern State gère les différents états de Mario :

1. **State Interface** : `MarioState`
   - `move()` : Déplacement selon l'état
   - `jump()` : Saut avec hauteur variable
   - `attack()` : Attaque spécifique à l'état
   - `takeDamage()` : Réaction aux dégâts
   - `collectPowerUp(PowerUp)` : Ramasser un power-up
2. **États Concrets**
   - `SmallMario` : État de base, vulnérable
   - `SuperMario` : Taille doublée, résiste à un coup
   - `FireMario` : Lance des boules de feu
   - `IceMario` : Lance des boules de glace
   - `InvincibleMario` : État temporaire invulnérable
3. **Transitions d'États**
   - Gestion automatique des transitions
   - Sauvegarde de l'état précédent pour Invincible
   - Règles de transition selon les power-ups

2.3 Pattern Decorator - Power-ups Combinables

Le pattern Decorator permet de combiner plusieurs améliorations :

1. **Component** : `Mario` (classe de base)
   - Capacités de base : courir, sauter
   - Statistiques : vitesse, force, points de vie
2. **Decorator Abstract** : `PowerUpDecorator`
   - Référence vers un objet Mario
   - Délégation des méthodes de base
   - Ajout de nouvelles capacités
3. **Décorateurs Concrets**
   - `SuperPowerDecorator` : +1 PV, force +20%
   - `FirePowerDecorator` : Attaque à distance
   - `IcePowerDecorator` : Gel des ennemis
   - `SpeedBoostDecorator` : Vitesse +50% (temporaire)
   - `ShieldDecorator` : Bouclier protecteur
   - `FlyingDecorator` : Capacité de vol
   - `DoubleJumpDecorator` : Saut supplémentaire

2.4 Intégration des Deux Patterns

1. **Coordination State-Decorator**
   - Les états influencent les décorateurs disponibles
   - Les décorateurs peuvent déclencher des changements d'état
   - Gestion des conflits entre états et décorateurs
2. **Système de Ressources**
   - Énergie pour certaines capacités
   - Durée limitée pour certains power-ups
   - Régénération automatique

2.5 Fonctionnalités Requises - Sujet B

1. **Gestion des états :**
   - Transitions automatiques selon les événements
   - Comportements différents par état
   - Gestion des états temporaires
2. **Système de power-ups :**
   - Application/suppression dynamique de décorateurs
   - Combinaisons multiples de power-ups
   - Calcul des statistiques cumulées
3. **Gameplay :**
   - Système de combat contre ennemis
   - Collecte de power-ups et objets
   - Score et progression
4. **Interface utilisateur :**
   - Affichage de l'état actuel de Mario
   - Liste des power-ups actifs
   - Statistiques en temps réel

3 Spécifications Techniques Communes

3.1 Architecture Générale

— Utilisation correcte des patterns choisis
— Respect des principes SOLID
— Séparation claire des responsabilités
— Code extensible et maintenable

3.2 Gestion des Exceptions

— Exceptions personnalisées appropriées
— Gestion robuste des erreurs
— Messages d'erreur informatifs

3.3 Tests et Démonstration

— Classe de test principale obligatoire
— Scénarios de démonstration complets
— Logs détaillés des opérations

4 Livrables Attendus

4.1 Rapport Écrit (50% de la note finale)

1. **Diagramme de classes UML** incluant :
   — Toutes les classes avec attributs et méthodes
   — Relations et dépendances clairement indiquées
   — Représentation explicite des patterns utilisés
   — Légende et annotations explicatives
2. **Justification du choix** de sujet (1/2 page)
3. **Explication détaillée** des patterns implémentés (1-2 pages)
4. **Guide d'utilisation** avec exemples (1 page)
5. **Code source Java** complet et bien formaté
6. **Captures d'écran** de l'exécution

4.2 Soutenance Technique (50% de la note finale)

1. **Présentation orale** (10-15 minutes) :
   — Explication du choix de sujet et de l'approche
   — Présentation du diagramme UML
   — Démonstration des patterns implémentés
2. **Démonstration pratique** :
   — Exécution en direct de l'application

- Présentation des fonctionnalités principales
- Tests de cas d'usage spécifiques

3. **Questions techniques** :
   - Explication détaillée du code
   - Justification des choix de conception
   - Discussion sur les améliorations possibles

5 Critères d'Évaluation

5.1 Rapport Écrit (50%)

Critère	Points	Description
Diagramme UML	4/10	Complétude, précision, respect des conventions
Patterns principaux	3/10	Implémentation correcte des 2 patterns
Fonctionnalités	2/10	Respect du cahier des charges
Qualité du code	1/10	Structure, lisibilité, bonnes pratiques

5.2 Soutenance Technique (50%)

Critère	Points	Description
Présentation orale	2/10	Clarté, structure, maîtrise du sujet
Démonstration	3/10	Fonctionnement, robustesse, cas d'usage
Maîtrise technique	3/10	Compréhension du code, justifications
Réponses aux questions	2/10	Pertinence, précision, réactivité

6 Consignes de Remise

MODALITÉS DE REMISE DU RAPPORT

- **Format :** Un seul fichier PDF contenant :
  1. Page de garde avec nom, prénom, groupe et **SUJET CHOISI**
  2. Justification du choix de sujet
  3. Diagramme UML (haute résolution, lisible)
  4. Documentation technique complète
  5. Code source Java complet et bien formaté
  6. Captures d'écran de l'exécution
- **Nom du fichier :** `TP_DesignPatterns_SUJET_NOM_Prenom.pdf`
  (où SUJET = EDITOR ou MARIO)
- **Email :** `haythemghz@yahoo.fr`
- **Objet :** `[ING-4-A-GLSI-OO] TP Design Patterns [SUJET] - NOM Prénom`
- **Date limite :** 02 Septembre 2025 à 23h59

PRÉPARATION À LA SOUTENANCE

Pour la séance de validation, vous devez :

- Apporter votre ordinateur portable avec le projet configuré
- Préparer une présentation de 10-15 minutes maximum
- Être capable d'expliquer chaque partie de votre code
- Prévoir des scénarios de démonstration variés
- Anticiper les questions techniques sur les patterns utilisés

La date exacte de la soutenance sera communiquée ultérieurement

7 Conseils Spécifiques

7.1 Pour le Sujet A (Command + Composite)

- Commencez par implémenter la hiérarchie Composite
- Testez chaque commande individuellement avant l'intégration
- Gérez soigneusement la mémoire dans l'historique des commandes
- Pensez aux cas limites (undo sur pile vide, etc.)
- Préparez des démonstrations visuelles pour la soutenance

7.2 Pour le Sujet B (State + Decorator)

- Définissez clairement les interactions entre états et décorateurs
- Implémentez d'abord les états simples, puis les décorateurs
- Testez les combinaisons complexes de power-ups
- Gérez les ressources et durées limitées avec des timers
- Préparez des scénarios de jeu variés pour la démonstration

7.3 Conseils pour la Soutenance

- Entraînez-vous à présenter votre projet à voix haute
- Préparez des réponses aux questions classiques sur les patterns
- Testez votre application sur plusieurs scénarios avant la soutenance
- Soyez prêt à modifier le code en direct si demandé
- Gardez une attitude professionnelle et confiante

Choisissez le sujet qui vous motive le plus et préparez-vous bien pour la soutenance !

Bonne chance à tous !