Travail élaboré par : TECHNOVA

Encadré par D.HAJER BERHOUMA

Objectif

Objectifs du patrons de structure

Fournir une interface unifiée et simplifiée à un système complexe, en centralisant l'accès aux fonctionnalités. Ce patron réduit le couplage en masquant la complexité interne, facilite l'usage du sous-système et agit comme point d'entrée unique pour le client.

Problèmes et solutions

Problème 1 : Complexité excessive du système

Description : Le système comporte un grand nombre de classes avec des interactions complexes. Le client doit comprendre tous les détails pour effectuer une action simple

Solution apportée : La façade encapsule cette complexité en fournissant des méthodes simples qui regroupent les appels aux différentes classes internes.

Problème 2 : Couplage fort entre client et sous-système

Description : Le client dépend directement de plusieurs classes du sous-système. Cela rend le code difficile à maintenir ou à modifier (changement dans une classe interne = impact sur le client).

Solution apportée : La façade agit comme un bouclier entre le client et les composants internes, ce qui réduit le couplage. Le client dépend uniquement de la façade.

Problème 3 : Difficulté de réutilisation

Description : Il est difficile de réutiliser certaines fonctionnalités du système, car elles nécessitent l'appel coordonné de plusieurs classes internes.

Solution apportée : La façade centralise et regroupe ces appels dans des méthodes réutilisables, prêtes à l'emploi.

Problème 4 : Trop de points d'entrée dans le sous-système

Description : Le client doit interagir avec plusieurs objets pour accomplir une tâche. Cela augmente le risque d'erreurs et rend l'API moins intuitive.

Solution apportée : La façade fournit un point d'entrée unique pour accéder aux fonctionnalités nécessaires, améliorant ainsi la cohérence et la lisibilité du code.

Problème 5 : Difficile de limiter l'accès à certaines parties du système

Description : Le client peut accéder à toutes les classes internes, même celles qui ne devraient pas être utilisées directement.

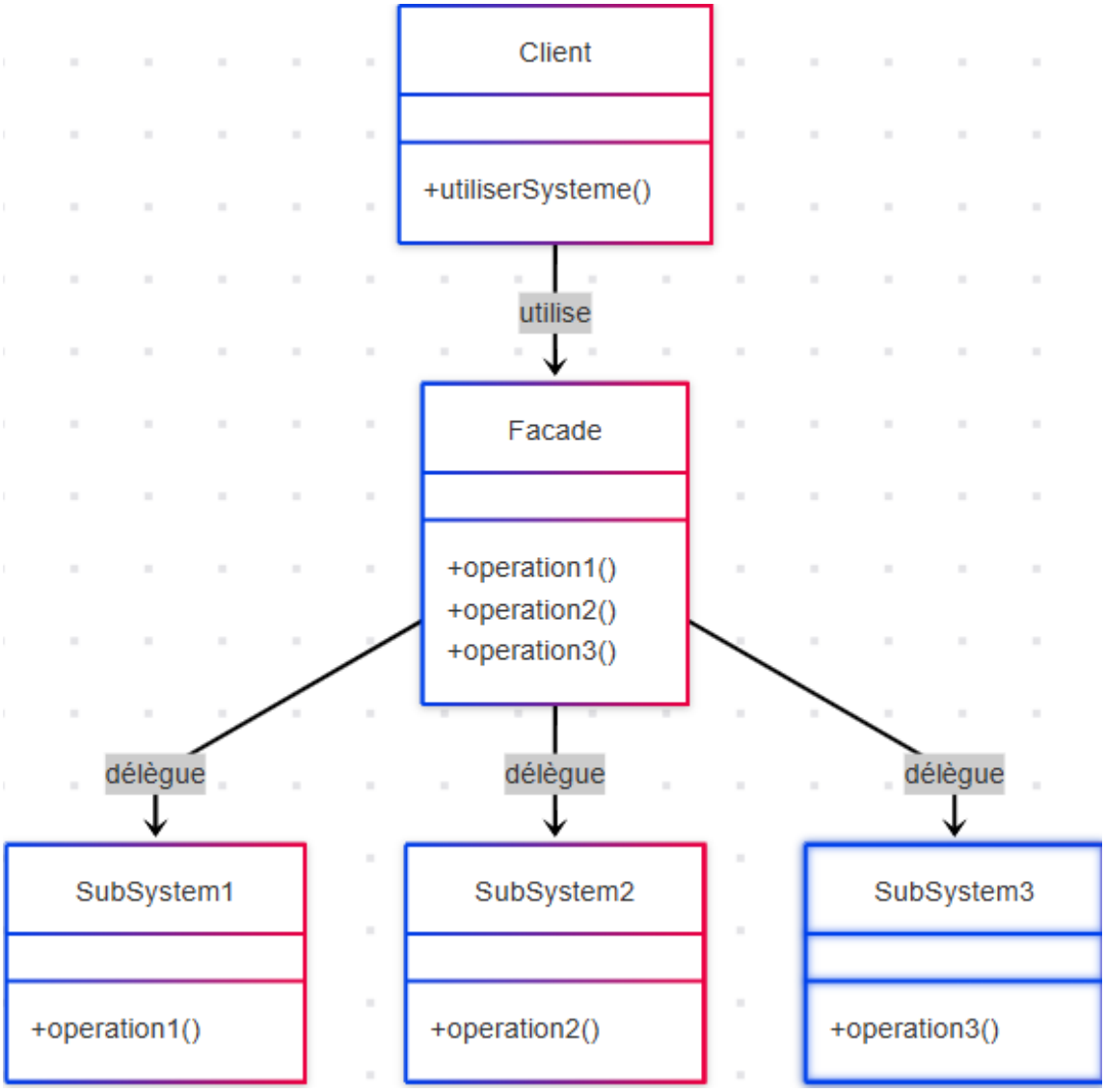
Solution apportée : La façade masque les détails internes du sous-système et contrôle l'accès, ne laissant visibles que les éléments nécessaires.

Problème 6 : Code du client difficile à tester

Description : Comme le client interagit avec plusieurs objets directement, les tests nécessitent beaucoup de configuration et de mocks.

Solution apportée : En passant par une façade, il devient plus simple de mock la façade pour les tests, car elle isole le client des détails techniques.

1. Diagramme du patron Facade En generale



exemple generale Diagramme du patron Facade

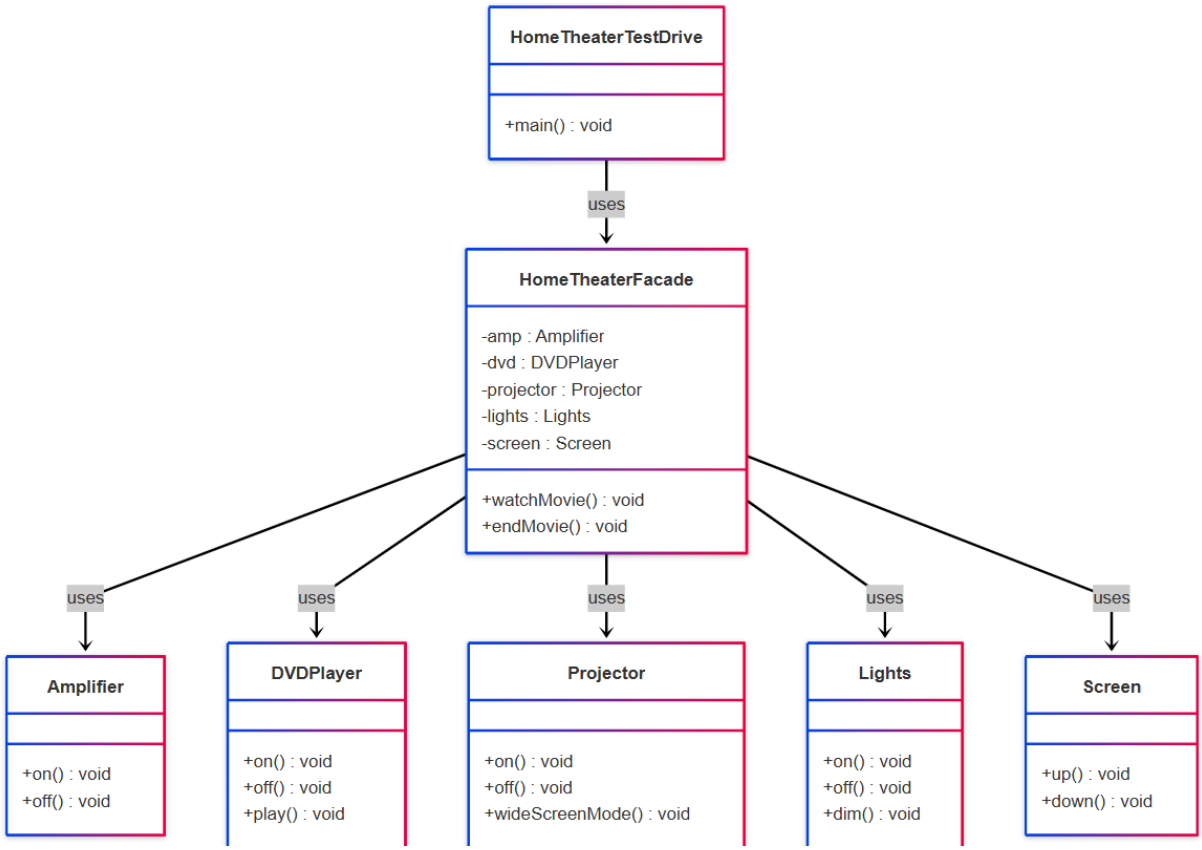


Diagramme du patron Facade pour civismart


TableauDeBordVille
+principal(args: String[])
utilise la façade
FaçadeVilleIntelligente
+creerUtilisateur()
+collecterDechets()
+traiterDocuments()
+allumerLumieres()
+eteindreLumieres()
+traiterTaxes()
+gererReclamation()
délègue la création d'utilisateur
délègue la collecte des déchets
délègue le traitement des documents
délègue la gestion de l'éclairage
délègue le traitement des taxes
délègue la gestion des réclamations
GestionUtilisateurs
+creerUtilisateur()
GestionDechets
+collecterDechets()
GestionDocuments
+traiterDocuments()
GestionEclairage
+allumerLumieres()
+eteindreLumieres()
GestionTaxes
+traiterTaxes()
GestionReclamations
+gererReclamation()