

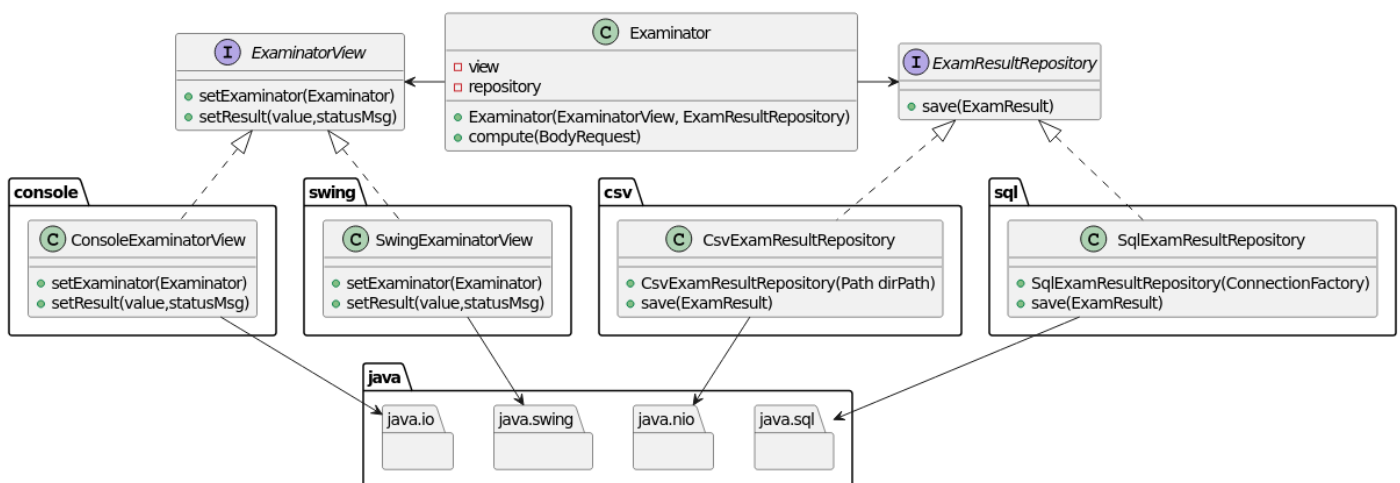
# MVP (Modèle Vue Présentateur)

## Patron architectural

Un patron architectural est une solution générale et réutilisable à un problème architectural, comme les patrons de conceptions mais ont une portée plus large. Dans le cas de celui que l'on va voir ici, on remarquera qu'il est lui même composé des patrons [Façade](#) et [Adaptateurs](#) vus précédemment.

## MVP

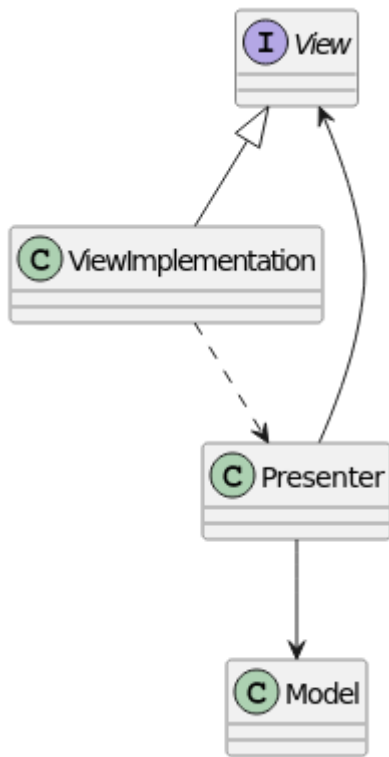
Voici ce que l'on obtient quand on combine les deux derniers patterns (façade et adaptateurs) :



Ici les classes `*ExaminatorView` et `*ResultRepository` sont des adaptateurs, tandis que les classes de `java` sont les services.

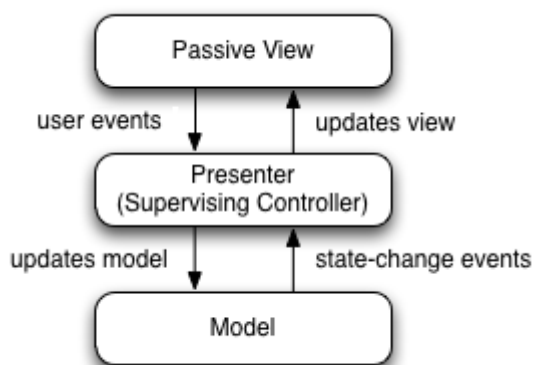
Mais on peut aussi y voir la façade car la classe `Examinator` agit comme une façade pour le reste du système.

En vérité, cette structure correspond au patron architectural "MVP" (Modèle Vue Présentateur)



Dans l'exemple précédent :

- Examiner est le Présentateur (il n'interagit qu'avec la vue et le modèle).
- ExamResultRepository est le Modèle qui s'occupe de tout ce qui est de la gestion des données (oui dans le MVP le modèle est simplement une classe et pas une interface, mais cela dépend simplement des besoins de l'application)
- ExaminerView est l'interface des vues
- ConsoleExaminerView et SwingExaminerView sont les classes concrètes de la vue.



## Caractéristiques du MVP

- Une vue ne connaît que son présentateur
- Le présentateur supervise les vues via une interface
- Le présentateur met à jour le modèle du domaine puis la vue
- Le patron de l'observateur peut être envisagé pour affaiblir le couplage

---

Revision #1

Created 3 October 2023 21:04:59 by SnowCode

Updated 3 October 2023 21:07:06 by SnowCode