

# INFO-H-200 : Programmation orientée objet

## TP 9 - Design patterns

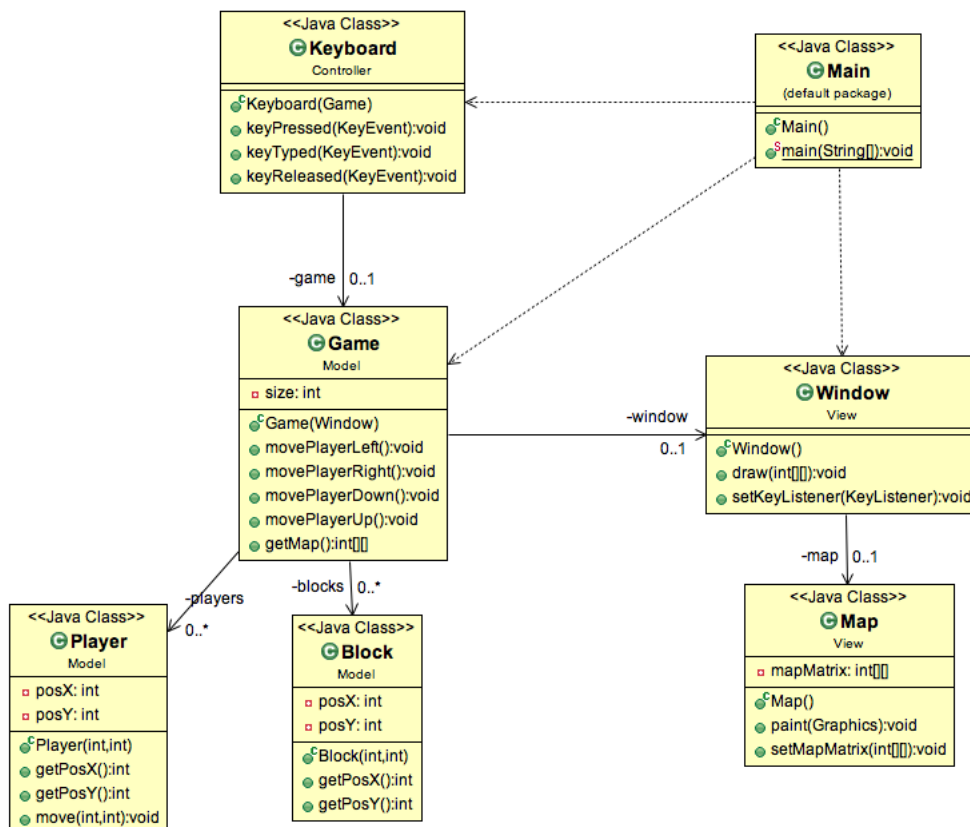
Professeur : Hugues Bersini

<http://cs.ulb.ac.be/public/teaching/infoh200>

Année académique 2015-2016

### Exercice 9.1 : Modèle-Vue-Contrôleur

Implémenter un jeu de plateau simple composé de blocs et d'un unique joueur pouvant se déplacer au sein du plateau de jeu. La structure attendue est décrite par le schéma ci-dessous. Seule la classe Main a connaissance de l'ensemble des composants du programme. Le jeu (modèle) notifie donc la vue en cas de changement et transmet ici une description de l'état du jeu sous la forme d'une matrice à window (vue). Le clavier (controller) quant à lui notifie le jeu des actions du joueur physique.

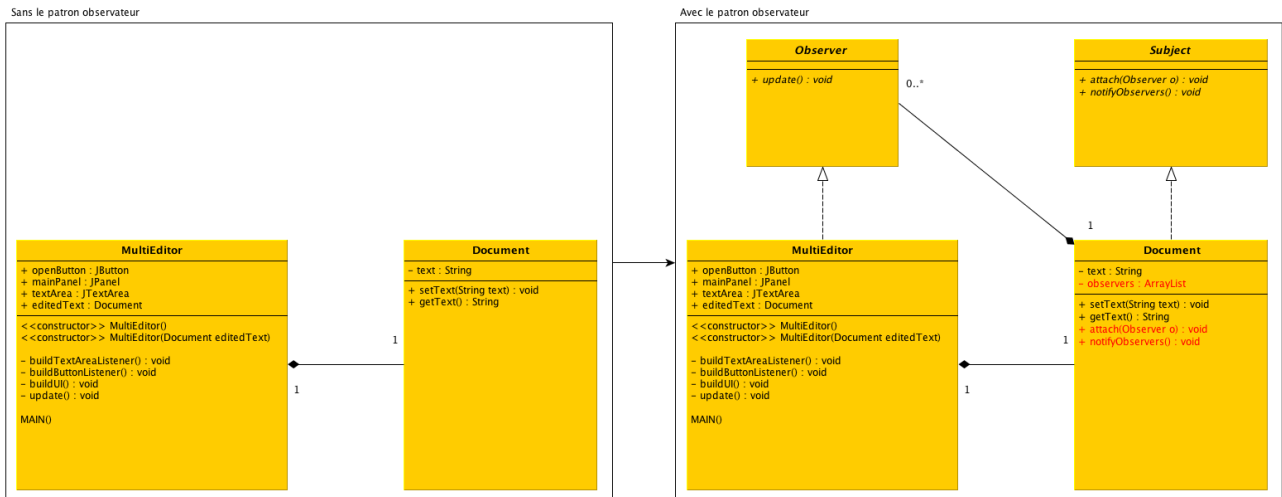


## Exercice 9.2 : Observateur

Appliquer l'*Observer Pattern* pour une application d'édition de texte multi-fenêtre. Quand un texte est modifié dans l'une des fenêtres, cette modification est répercutée à l'ensemble des autres fenêtres.

Pour cela, implémenter les classes suivantes, les interfaces `Subject` et `Observer` et les *listeners* nécessaires.

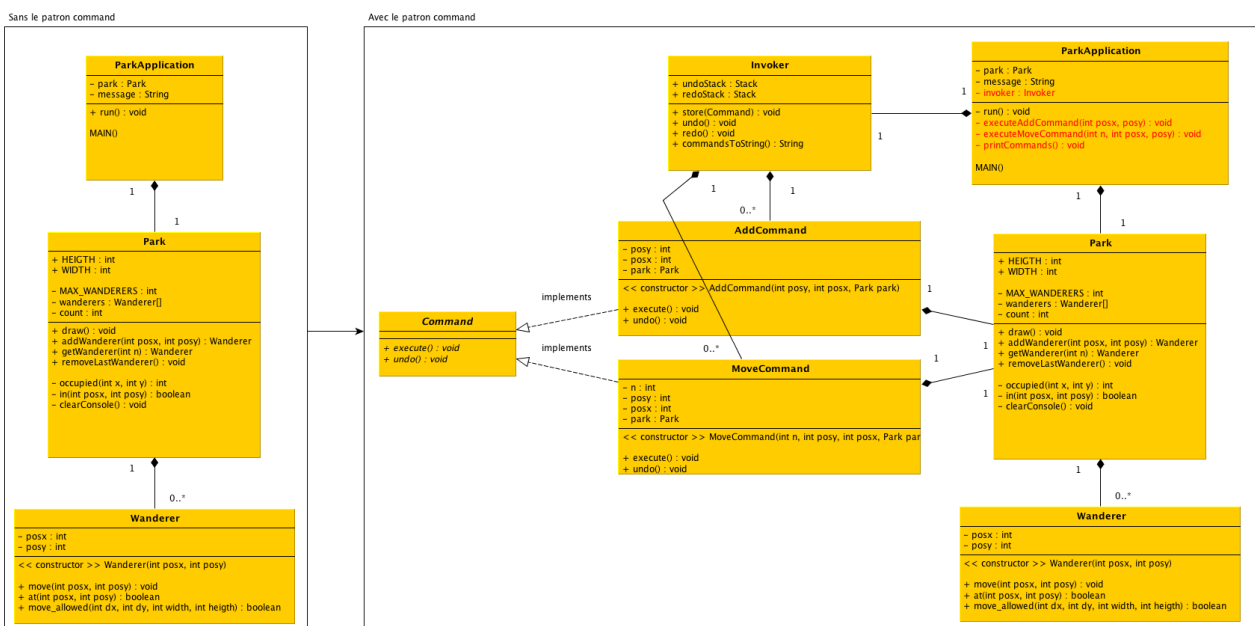
- `Document` : contient le texte et qui est le sujet observé.
- `MultiEditor` : fenêtre permettant l'édition du texte et qui observe le texte.



## Exercice 9.3 : Commande

Voici le code d'une application console permettant de faire circuler un des promeneurs et d'ajouter de nouveaux promeneurs dans un parc (représenté par une matrice de  $20 \times 20$ ).

1. Essayer cette application. Pour ajouter un promeneur, il faut écrire `add x y` où  $x$  et  $y$  est la position du promeneur. Pour bouger un promeneur, il faut écrire `move n x y` où  $n$  est le numéro du promeneur.
2. Analyser le code de l'application.
3. Modifier ce code pour permettre des "undos" multiples.
4. Modifier ce code pour permettre des "redos" multiples.
5. Ajouter une commande pour retirer un promeneur.



INFO-H-200 - Programmation orientée objet  
TP 9  
Corrections

---

**Voir les fichiers fournis**