Projet UML – C++

Jeux d'échecs et de dames

D'après A. Gademer

1 Objectifs

L'objectif de ce projet est de mettre en œuvre en C++ les mécanismes d'héritage, de polymorphisme et d'encapsulation sur l'exemple des jeux d'échecs et de dames.

2 Présentation

Nous désirons faire une application C++ qui permet de jouer de manière simplifiée aux échecs ou aux dames.

Pour ce faire, nous allons créer un certain nombre de classes décrites par la figure 1.

Le programme principal (fonction main) initialise le jeu et demande en boucle les coups des joueurs, redemandant au besoin si un joueur lui spécifie un coup non valide.

Nous commencerons par les classes génériques : les classes Position, Plateau et Piece.

- La classe Position correspond aux coordonnées sur l'échiquier sous la forme d'une chaîne alphanumérique (ex : D4).
- La classe Plateau correspond à une classe abstraite décrivant n'importe quel plateau de jeu. Elle sera spécifiée plus tard en Echiquier et Dame.
- La classe Piece correspond à la classe abstraite décrivant n'importe quelle pièce. Elle pourra être spécifiée en Pion, Cavalier, et pour le jeu d'échec, ou en PionDame et DameDame pour le jeux de dame.

On trouvera ensuite toutes les classes correspondant au jeu d'échec classique.

- La classe JeuxEchec, qui initialise le jeu et demande en boucle les coups des joueurs, redemandant au besoin si un joueur lui spécifie un coup non valide.
- La classe Echiquier qui hérite de Plateau
- Toutes les sous-classes de Piece : Pion, Cavalier, Fou, Tour, Reine, Roi. On remarquera que comme une Dame se déplace à la fois comme une Tour et comme un Fou, on peut très facilement créer la Dame par héritage multiple depuis le Fou et la Tour.
- On trouvera enfin toutes les classes correspondant au jeu de dame.
- La classe JeuxDame, qui initialise le jeu et demande en boucle les coups des joueurs, redemandant au besoin si un joueur lui spécifie un coup non valide.
- La classe Damier qui hérite de Plateau
- Toutes les sous-classes de Piece : PionDame, ReineDame.

Par la suite, nous allons décrire le fonctionnement pour le jeu d'échec, et vous définirez vous même comme fonctionne le jeu de dame.

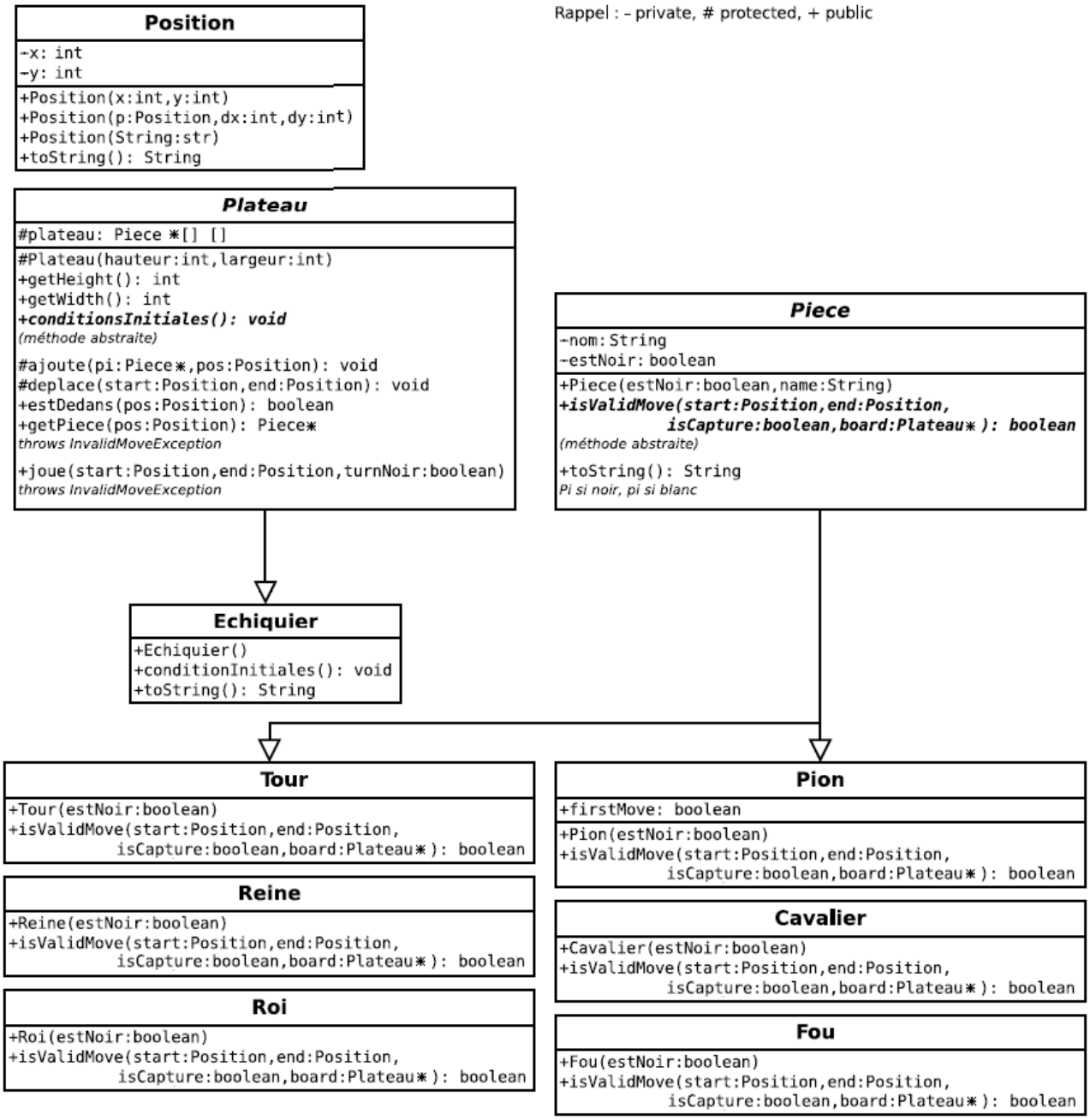

Figure 1 : Diagramme UML

Les objets passés en paramètre aux méthodes le sont en général par référence.

Les objets abstraits passés en paramètre aux méthodes le sont **OBLIGATOIREMENT** par pointeur.

3 Quelques explications

3.1 Position

Les champs x et y seront constants. Une fois la position créée, on ne peut la modifier. Il n'y aura donc que des get, pas de set.

On peut créer une position :

- à partir de deux entiers
- à partir d'une autre position et de deux valeurs de translation
- à partir d'une String de la forme d'une lettre suivit d'un chifre (ex : "D4").

La méthode toString() renvoie une String représentant les coordonnées sous forme d'une lettre suivie d'un chifre (ex : "A3").

3.2 Plateau

La méthode conditionsInitiales() est déclarée abstraite et n'est pas définie dans cette classe étant donné qu'on ne sait pas à quel jeu on joue à ce niveau.

Les méthodes Plateau(int,int), ajoute(Piece *,Position) et deplace(Position,Position) sont déclarées protected et ne sont visibles qu'aux sous-classes de Plateau.

La méthode getPiece(Position) renvoie le pointeur sur la pièce correspondante (null le cas échéant) ou jette une exception de type InvalidMoveException si les coordonnées sont hors du plateau.

La méthode joue(Position,Position,boolean) renvoie une exception de type InvalidMoveException si :

- la case de départ est vide
- la case de départ contient une pièce de l'autre couleur
- la case d'arrivée contient une case de ma couleur
- le mouvement de la pièce ne permet pas d'arriver sur la case d'arrivée

Sinon, elle déplace la pièce indiquée.

3.3 Piece

Les champs de la classe Piece sont constants. Une fois la pièce créée, on ne peut la modifier.Il n'y aura donc que des get, pas de set.

La méthode isValidMove(Position, Position, boolean, Plateau *) est abstraite.

Elle dépend des règles de mouvements et ne peut donc être définie ici.

La méthode toString() renvoie les deux premières lettres du nom, dont la première en majuscule si la pièce appartient aux noirs.

3.4 Relation entre Position, Piece et Plateau

Les instances des classes dérivées de Piece (les pions, tours, etc. placés sur le plateau) ne connaissent pas leur propre position sur le plateau.

Le seul à savoir la position des pièces est le plateau lui-même (par le biais du tableau de Piece à double entrée).

Lorsque le jeu appelle, par exemple, la méthode

joue (new P o s i t i o n (" D6 ") , new P o s i t i o n (" D4 ") , false)

de l'objet Échiquier instancié , c'est ce même objet qui a la connaissance des positions et qui peut retrouver quelle est la pièce en D6 (si il y en a une et si elle est de la "bonne" couleur) en regardant dans la case 3,5 de son tableau.

Une fois la pièce retrouvée, l'échiquier peut demander à la pièce si la position d'arrivée est valide.

Dans ce cas, l'échiquier déplace la pièce sur la case d'arrivée. Sinon, il renvoie une exception.

3.5 Echiquier

La classe Echiquier hérite de la classe Plateau. Nous n'avons donc pas beaucoup de choses à redéfinir.

Le constructeur Echiquier instancie un échiquier de 8 par 8.

La méthode conditionsInitiales place toutes les pièces selon les règles des échecs classiques (cf. Fig 2).

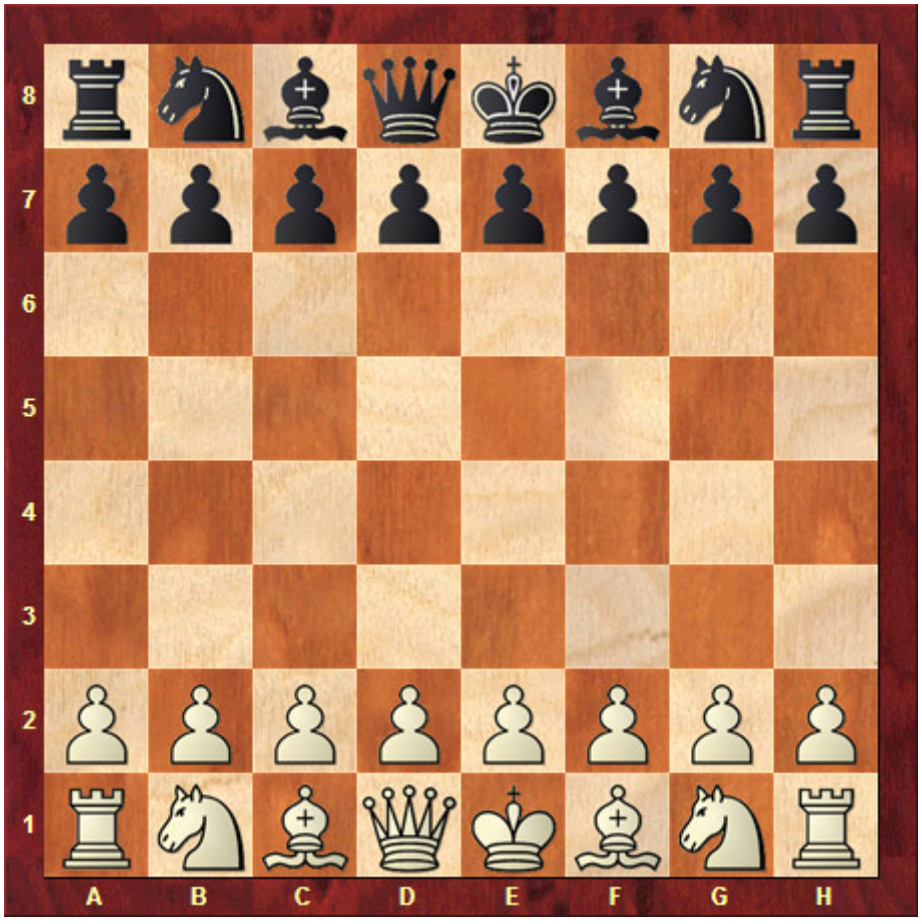

Figure 2 : Disposition de départ

La méthode toString() génère une chaîne de caractères comprenant toutes les cases de l'échiquier avec les pièces disposées dessus.

3.6 Pion

La classe pion se distingue des autres par un champ supplémentaire qui permet de distinguer si c'est le premier mouvement de la pièce ou non.

On considérera que la première demande valide de mouvement sera le premier mouvement de la pièce (c'est une approximation raisonnable).

La méthode isValidMove(Position, Position, boolean, Plateau) devra distinguer les cas du premier mouvement mais aussi celle de la capture ou non d'une pièce adverse.

3.7 Cavalier, Fou. . .

Ces classes se ressemblent beaucoup sauf pour l'implémentation de la méthode isValidMove(Position, Position, boolean, Plateau) qui doit correspondre aux règles de mouvement du jeu d'échec. La Reine peut se faire très facilement en héritant à la fois de la Tour et du Fou (héritage en diamant).

On fera attention :

- au respect du mouvement
- à l'interdiction (ou non) de sauter des pièces (alliées et ennemies).

C'est pour cette dernière raison que vous avez accès au plateau dans cette méthode.

Astuces : La plupart des règles peuvent se déduire des valeurs de distX et disY correspondant aux distances sur les deux axes entre end et start.

En retrouvant dx et dy comme étant la direction (-1 ou 1) selon ces deux axes, on peut généralement parcourir le chemin que ferait la pièce en mouvement et vérifier ainsi l'absence d'obstacle.

On remarquera que la présence d'une pièce adverse sur la dernière case du mouvement n'est pas un obstacle.

3.8 JeuxEchec

Vous instancierez le nouvel échiquier puis dans une boucle :

- vous récupérerez une String de la part de l'utilisateur
- si celle-ci vaut "quit" vous quitterez le programme.
- si celle-ci est de la forme "D6 D4" vous effectuerez l'action et vous redemanderez une chaîne tant que l'action provoque une erreur.
- vous faites de même pour l'autre joueur.

3.9 JeuxDame

- Vous instancierez le nouveau damier puis dans une boucle :
- vous récupérerez une String de la part de l'utilisateur
- si celle-ci vaut "quit" vous quitterez le programme.
- si celle-ci est de la forme "D6 D4" vous effectuerez l'action et vous redemanderez une chaîne tant que l'action provoque une erreur.
- vous faites de même pour l'autre joueur.