

1. Jeu d'échecs (1/2)

Héritage, Classe abstraite, Redéfinition de méthode, Opérateurs d'affectation et de comparaison, Opérateur de sortie.

Le programme qu'il est demandé d'écrire permettra de simuler une partie d'échecs. La simulation ne sera que partielle car seules certaines règles du jeu seront prises en compte.

1. Une partie d'échecs se joue sur un échiquier, qui peut être vu comme une grille de 8 colonnes sur 8 lignes = 64 cases. Sur chacune de ces cases pourra être placée une pièce. Écrire une classe position permettant de stocker une position sur un échiquier. Il pourra être pertinent de définir un type coordonnee équivalant à un type entier signé, une position étant composée de deux coordonnee. position sera munie d'un constructeur à deux coordonnee, pas de constructeur par défaut.

2. Une position devra pouvoir être copiée (pour créer un nouvel objet par recopie d'un objet existant ou par un appel à l'opérateur d'affectation).

Écrire des accesseurs nommés x et y et des mutateurs setx et sety.

Une position devra pouvoir être comparée par == et != avec une autre position, .

Une position devra pouvoir être envoyée sur un flux de sortie sous la forme « (x,y) ».

Définir une méthode estvalide retournant true si la position est valide sur un échiquier, false sinon.

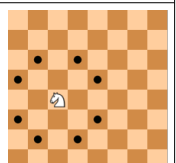
3. Une pièce d'un échiquier dispose d'une couleur (parmi deux couleurs possibles : blanc et noir, définir un type énuméré couleur pour cela) et d'une position. Une pièce peut être de différents types : pion, cavalier, dame, roi. À chacun de ces types correspond un ensemble de déplacements possibles (par exemple, le pion peut avancer d'une case, alors que la dame peut avancer de plusieurs cases à chaque déplacement), un symbole (une lettre utilisée pour l'affichage) et une valeur, qui est un entier représentant l'importance de la pièce. Noter que la valeur, le symbole et les déplacements dépendent **uniquement du type de la pièce** (tous les pions ont la même valeur, tous les pions peuvent se déplacer de la même façon, etc.). Les valeurs, symboles et déplacements sont décrits dans le tableau ci-dessous.

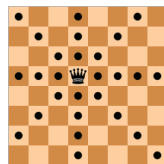
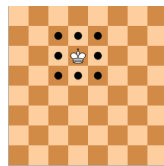
Définir une classe piece ainsi que des sous-classes de piece pour chacun des quatre types de pièces.

Munir ces classes de possibilités d'accéder à la valeur (par une méthode valeur), à la couleur (cou), au symbole (symbole), et à la position (pos) actuelle de la pièce.

Écrire une méthode toString qui retournera une chaîne du type tcxy, où t est le symbole du type de pièce, c la couleur (B ou N), x la colonne, y la ligne. Par exemple CN42 représentera le cavalier noir en position 4,2.

Pièce	Valeur	Symbole	Déplacements possibles
Pion	1	P	Pions blancs : une case vers le haut. Pions noirs : une case vers le bas. Lors du premier déplacement, une ou deux cases dans la direction correspondant à la couleur. Dans un premier temps, vous ne prendrez pas en compte cette particularité du premier déplacement d'un pion.
Cavalier	3	C	Une case dans une direction, deux cases dans une direction perpendiculaire à la première direction.



Dame	9	D	Déplacement en diagonale, sur les colonnes ou sur les lignes.	
Roi	20	R	Déplacement d'une seule case en diagonale, sur les colonnes ou sur les lignes.	

4. Munir ces classes d'une méthode déplacementspossibles retournant l'ensemble des positions accessibles pour la pièce, cela dépend évidemment du type de pièce et de la position courante de la pièce.
5. Définir la méthode accepterposition prenant comme paramètre une position et retournant un booléen valant true si la position passée en paramètre peut être atteinte en fonction de la position actuelle de la pièce, false sinon.
6. Définir la méthode déplacer prenant comme paramètre une position destination et déplaçant la pièce à la position destination (si la position est acceptable) et retournant true si le déplacement a été effectué, false s'il est impossible.

2. Bibliothèque

Héritage, Classe abstraite, Redéfinition de méthode, Polymorphisme, Opérateur de sortie, Constructeur virtuel.

Une bibliothèque offre à ses usagers la possibilité d'emprunter des documents. Selon la nature de ces documents, les conditions de prêt sont différentes. La bibliothèque dispose de vidéos, de livres et de périodiques. Les périodiques ne peuvent pas être empruntés. Les vidéos peuvent être empruntées. Certains livres peuvent être empruntés, mais d'autres non. Les informations à représenter concernent le titre et le nom de l'auteur. En plus, pour les périodiques, il est demandé de stocker le numéro. Pour les vidéos, il est demandé de représenter le type de support (DVD, Blu-Ray, Blu-Ray3D). Enfin, le nombre de pages doit être représenté pour les documents écrits, car il permet de calculer le coût du document (ce coût est imputé à la personne ayant emprunté le document en cas de perte, ou à la personne consultant le document en cas de détérioration) : Le coût des documents écrits est de 0,50 € la page, alors que le coût d'une vidéo est le même pour toutes les vidéos : 70 €.

1. Définir des classes pour représenter les différents types de documents, la possibilité de les emprunter, et leur coût, en factorisant au mieux le code. Tester les classes en écrivant une fonction main appelant les différentes méthodes afin de tester leur fonctionnement.
2. Écrire un opérateur de sortie sur flux fonctionnant sur les différents types de document et affichant le Type (livre, périodique, vidéo), Titre, Nom de l'auteur, Empruntable (ou non empruntable), Coût, suivi des informations supplémentaires spécifiques à certains types de documents : nombre de pages, type de support vidéo, numéro du périodique.
3. Écrire une classe bibliotheque contenant un ensemble de documents. On ne déclarera dans cette classe qu'un seul conteneur pour manipuler tous les documents. Dans cet exercice, on utilisera les pointeurs bruts : les std::unique_ptr et std::shared_ptr seront utilisés dans les exercices suivants. Attention, votre code ne devra pas comporter de fuites mémoire.
4. Écrire une (des) méthode(s) permettant d'ajouter un document à la bibliothèque.
5. Écrire une (des) méthode(s) permettant d'accéder aux documents depuis l'extérieur de la classe, par exemple pour les parcourir afin de les afficher par l'opérateur de sortie sur flux, mais sans pouvoir les modifier, et sans pouvoir « casser » la classe bibliotheque.
6. Écrire une méthode prenant comme paramètre un type de support vidéo et retournant le nombre de vidéos de la bibliothèque qui sont de ce type.

7. Écrire une méthode permettant de faire une copie d'une bibliothèque. Attention : la copie de la bibliothèque contiendra des copies des documents de la bibliothèque originale (et non des pointeurs sur les mêmes instances). Le code de cette méthode devra gérer d'autres types de documents (qui n'ont pas encore été définis) sans aucune modification.