
Classe, objet, interface, héritage, polymorphisme

Ce TD vise à construire progressivement un jeu d'échecs interactif pour redécouvrir la programmation objet en C++. Vous devez dans un premier temps récupérer les fichiers situés dans le répertoire

Bibliotheque/STAGE_PROG_SIL/Echecs

Exercice 1 : Prise en main

Regardez le code de `Echecs` et exécutez-le.

Exercice 2 : Constructeurs

Faire un nouveau constructeur qui prend en paramètres `x,y` et la couleur. Instanciez un nouvel objet `p2` et affichez le.

Exercice 3 : Accès aux attributs / méthodes

Essayez d'accéder à l'attribut `m_x` directement dans le `main` en écrivant `p1.m_x = 2;` . Peut-on toujours compiler ? Mettez-le en section `public`. Que se passe-t-il alors ?

Exercice 4 : Méthodes

Ecrivez une nouvelle méthode `isBlack` qui retourne l'inverse de `isWhite`.

Exercice 5 : Utilisation d'autres modules.

On veut intégrer une méthode d'affichage dans la classe `Piece`, appelée `affiche`. Constatez qu'il faut alors include `<iostream>` dans `Piece.cxx`. Pourquoi ? Dans le programme principal, appelez plutôt la méthode `affiche`.

Exercice 6 : Constructeurs ; destructeur ; durée de vie d'un objet

Dans chaque constructeur de `Piece`, écrivez une ligne du genre `cout << "Une piece creee" << endl;`

Ecrivez une méthode de prototype `~Piece()` dans `Piece`. Faites que le corps de cette méthode affiche "Une piece detruite". Qu'en déduisez-vous sur la durée des objets `p1` et `p2` ? On dit que ces objets ont été alloués *statiquement*.

Exercice 7 : Test de l'instanciation des tableaux

Dans le programme principal, déclarez un tableau de 4 pièces. Que constatez-vous ? Combien d'instances sont créées ? Est-ce similaire à JAVA ? Quel est le constructeur appelé ?

Exercice 8 : Classe Joueur

Ecrivez maintenant une classe `Joueur`. Cette classe représente un joueur d'échecs, qui joue les blancs ou les noirs. Il possède 16 pièces à des positions bien déterminées (coordonnée `y` : 1 ou 2 pour le blanc, 7 ou 8 pour le noir, coordonnée `x` de 1 à 8 pour les deux). On ignorera pour le moment que les pièces sont différenciées. On proposera un constructeur de `Joueur` qui aura un comportement différent selon que le joueur est blanc ou noir. On testera cette classe en instanciant deux joueurs dans le programme principal. On écrira aussi une méthode `affiche` qui liste les pièces du joueur.

Exercice 9 : Passage de paramètres

Ecrivez une fonction ou méthode qui teste si deux pièces sont au même endroit. Vérifiez-la en comparant deux pièces quelconques des joueurs. Y a-t-il de nouvelles instances de `Piece` créées ?

Réécrivez la fonction ou méthode précédente avec un passage par référence.

Exercice 10 : Pointeurs ; classe `Echiquier`

Récupérez les fichiers `Echiquier.h` et `Echiquier.cxx`. Pourquoi l'attribut `m_cases` ne mémorise-t-il que des pointeurs et pas des pièces directement ? Ecrivez proprement le constructeur pour que l'échiquier soit vide au début. Complétez les méthodes spécifiées. Testez en instanciant un échiquier et en l'affichant.

Ecrivez ensuite une méthode de `Joueur` qui place automatiquement toutes ses pièces sur un échiquier donné en paramètre.

Exercice 11 : Héritage

Définissez deux classes `JoueurBlanc` et `JoueurNoir` qui héritent de `Joueur` et dont les constructeurs initialisent correctement les pièces. Mettez à jour `Joueur` en éliminant le constructeur `Joueur(bool)`. Mettez du texte dans le constructeur par défaut de `Joueur` et dans les destructeurs de ces classes pour observer dans quel ordre sont appelés les constructeurs à l'instanciation et les destructeurs à la fin du programme.

Au lieu d'instancier deux joueurs, instanciez un `JoueurBlanc` et un `JoueurNoir`. Vérifiez que vous pouvez toujours les afficher avec `affiche`.

Exercice 12 : Polymorphisme ; méthodes virtuelles

On va pouvoir modéliser maintenant chaque pièce du jeu d'échecs avec son comportement propre. On écrira donc une classe par type de pièce : `Roi`, `Reine`, `Fou`, `Cavalier`, `Tour`, `Pion`. Ces classes spécialiseront une méthode virtuelle de pièce de prototype :

```
bool mouvementValide( Echiquier & e, int x, int y );
```

Attention, certaines pièces ont des mouvements autorisés différents selon noir ou blanc. Pourquoi passer un échiquier en paramètre ? Vous pourrez aussi spécialiser l'affichage des pièces ainsi : Blanc : `RQFCTP`, Noir : `rqfctp`. Dans les joueurs, instanciez correctement les nouvelles pièces. Vérifiez ensuite vos méthodes avec une interface simple de saisie au clavier.

Exercice 13 : Echecs

Terminez le jeu d'échecs. On notera qu'il faudra gérer la zone où un roi est mis en échec. On pourra sans doute étoffer la classe `Echiquier` pour mémoriser les zones où les rois ne peuvent aller. On ignorera le roque et la prise en passant.

Un extension possible du jeu est l'initialisation d'un jeu à partir d'un fichier.