(M2103) Test 2 de M2103

Le 8 juin 2019– durée : 2h

© *Marc Laporte marc.laporte@univ-amu.fr*
Aix Marseille Université
I.U.T. d'Aix en Provence - Département Informatique

Institut Universitaire de Technologie
Aix-Marseille Université
Département Informatique, Aix

Preambule

Documents autorisés : une feuille simple A4 manuscrite recto / verso.
Remarques :

- Lire attentivement tout le sujet, avant de commencer. Le barème, donné à titre indicatif, est sur 24 points, et 0 de bonus ;
- Écriture au crayon de papier interdite.

Pour rappel (information), l'exécution du petit programme suivant :

```
// Example program
#include <iostream>
#include <vector>

using namespace std;
class CX
{
  public :
    virtual void operator () (void) { cout << "CX" << endl; }

    virtual ~CX (void) {}

};// CX

class CY : public CX
{
  public :
    virtual void operator () (void) { cout << "CY" << endl; }

    virtual ~CY (void) {}

};// CY

void f (CX * x) { (*x) (); }

int main()
{
    CX * x = new CX;
    CY * y = new CY;
    vector <CX*> v {x, y};

    f (v [0]);
    f (v [1]);

    return 0;
}
```

est l'affichage sur l'écran de :

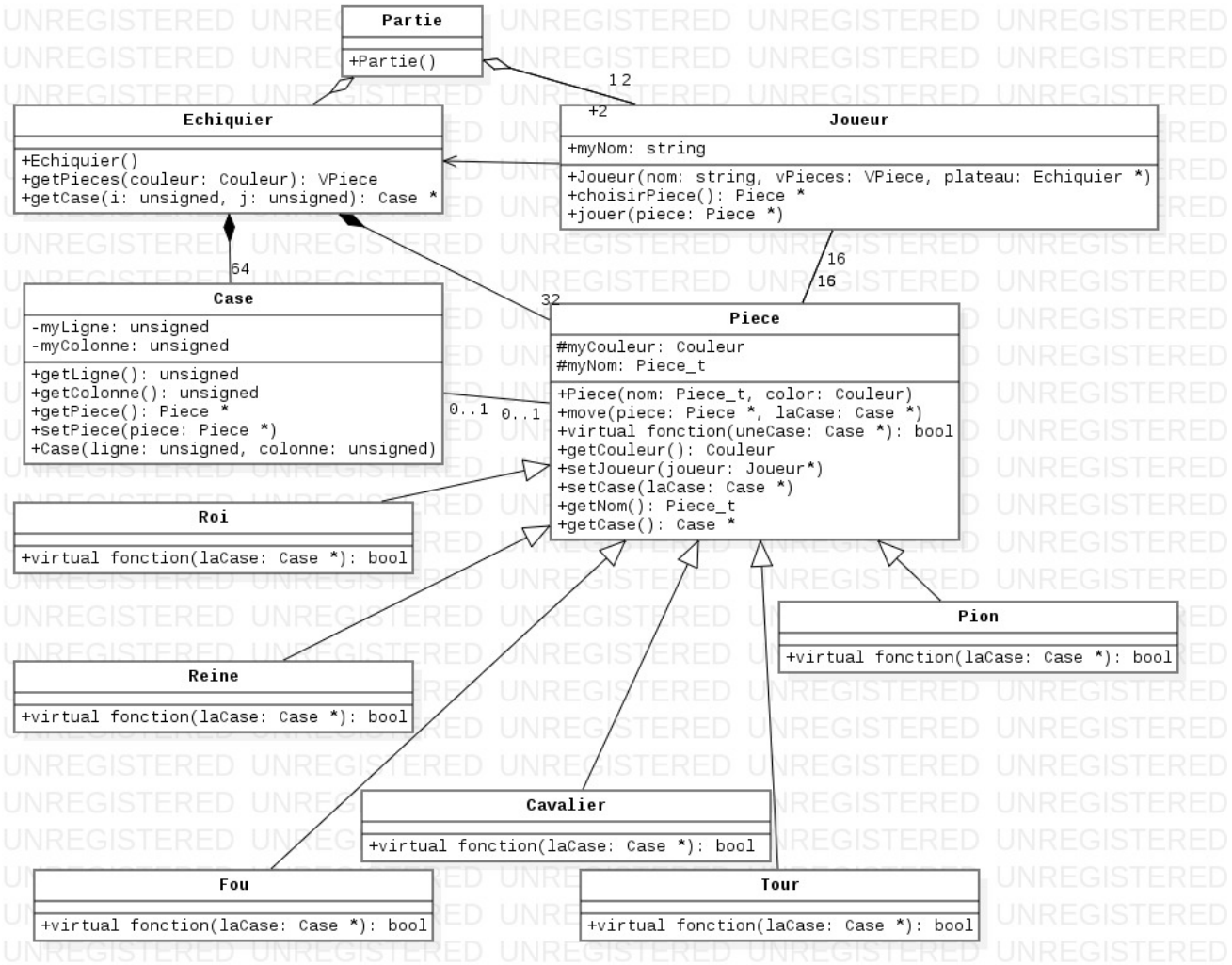
```
CX
CY
```

C'est du polymorphisme.

Avec les définitions suivantes dont vous disposez :

```
#define classdef typedef
classdef vector<Piece *> VPiece;
enum Couleur {KBlanc, KNoir};
enum Piece_t {KAucun, KRoi, KReine, KFou, KCavalier, KTour, KPion};
```

Le but de cet exercice est de programmer en `C++` un petit peu d'une partie d'échec, en partie modélisée par le diagramme de classes `UML` suivant :



Exercice 1.

Question 1.1 (1 point) :

Déclarer la classe `Partie` telle qu'elle est décrite sur le diagramme de classes.

Question 1.2 (1 point) :

Déclarer la classe `Echiquier` telle qu'elle est décrite sur le diagramme de classes. L'ensemble des `Case`s sera une matrice de 8 lignes et 8 colonnes.

Question 1.3 (2 points) :

Déclarer la classe `Joueur` telle qu'elle est décrite sur le diagramme de classes.

Question 1.4 (1 point) :

Déclarer la classe `Case` telle qu'elle est décrite sur le diagramme de classes.

Question 1.5 (3 points) :

Déclarer la classe `Piece` telle qu'elle est décrite sur le diagramme de classes. La méthode `fonction()` doit être transformée en un *opérateur fonction*.

Question 1.6 (1 point) :

Déclarer la classe `Roi` telle qu'elle est décrite sur le diagramme de classes.

Exercice 2.

Si vous avez bien compris le diagramme de classes précédent, l'échiquier est une matrice de `Case`s. Sur chaque `Case` est posée (ou pas) une `Piece`. Le constructeur (qui n'est pas à écrire) crée la matrice, les cases, y dépose les pièces (`Roi`, `Reine`, `Pion`s, ...) blanches (`KBlanc`) et les pièces noires (`KNoir`) et associe (dans les 2 sens) `Case`s et `Piece`s (un pointeur sur la `Piece` déposée dessus pour une `Case` et un pointeur vers la `Case` sur laquelle elle est posée pour une `Piece`). Le pointeur vers la `Piece` des `Case`s sur lesquelles ne sont posées aucune pièce sera un pointeur nul `nullptr`.

Question 2.1 (2 points) :

Ecrire la méthode `getPieces()` qui renvoie un vecteur qui contient les pointeurs sur les 16 `Piece`s dont la couleur (`KBlanc` ou `KNoir`) lui est passé en paramètre. Pour cela elle parcourt chaque `Case` de l'échiquier, regarde, en utilisant la méthode `getPiece()`, si une pièce est posée dessus (pointeur non nul). Si c'est le cas, si le couleur de la `Piece` (méthode `getCouleur()`) est la bonne, ajoute le pointeur sur cette `Piece` au vecteur qui sera renvoyé après examen de toutes les `Case`s.

Question 2.2 (1 point) :

Ecrire la méthode `getCase()` qui renvoie un pointeur sur la `Case` qui se trouve à la ième ligne et jième colonne de l'échiquier.

Question 2.3 (1 point) :

Ecrire le constructeur de la classe `Case` dont le diagramme de classe et les paramètres ne nécessitent aucune autre explication.

Exercice 3.

Si vous avez bien compris le diagramme de classes précédent, un `Joueur` a un nom, on lui associe un `Echiquier` et 16 `Piece`s. Une fois qu'il a choisi une `Piece`, il joue.

Question 3.1 (1 point) :

Ecrire le constructeur de la classe `Joueur`, comme décrit sur le diagramme de classes..

Question 3.2 (2 points) :

Ecrire la méthode `jouer()` qui

- demande à l'utilisateur un numéro de ligne et un numéro de colonne pour poser la `Piece` passée en paramètre ;
- vérifie que la `Case` repérée par ces 2 indices est valide (appel de l'*opérateur fonction* pour la `Piece` passée en paramètre ;
- si ce n'est pas le cas, refait les 2 opérations précédentes ;
- quand la `Case` est enfin valide, déplace la pièce passée en paramètre dans la case trouvée (méthode `move()` de la `Piece` jouée).

Question 3.3 (2 points) :

Ecrire la méthode `move()` qui :

- rend libre la `Case` actuellement occupée par la `Piece` référencée en paramètre (la référence à la `Piece` associée à cette `Case` devient nulle) ;
- associe la nouvelle `Case` (référencée en paramètre) à la `Piece` référencée en paramètre ;
- associe la `Piece` (référencée en paramètre) à la nouvelle `Case` (référencée en paramètre).

Exercice 4.

Question 4.1 (3 points) :

Ecrire le constructeur de la classe `Partie` qui :

- donne aux attributs de la classe les valeurs des paramètres de ce constructeur ;
- fixe le joueur qui va jouer ;
- dans une boucle infinie :
  - fait choisir au joueur fixé une `Piece` (méthode `choisir()` (qui n'est pas à écrire) de la classe `Joueur`) ;
  - fait jouer cette `Piece` au joueur fixé ;
  - passe au joueur suivant.

Exercice 5.

Question 5.1 (3 points) :

Ecrire une partie de programme qui :

- Crée un `Echiquier` ;
- Crée 2 `Joueurs` ;
- Crée une `Partie`.