

Evaluation de BTSNIR2 Sept 2020.

Ce devoir se compose en deux parties :

La première partie consiste à mettre en œuvre le déplacement des différentes pièces d'un jeu d'échec. Dans cette partie apparaîtront les notions d'héritage et de surdéfinition d'opérateur. La deuxième partie traite le fonctionnement d'une chaîne de caractères. En plus de la première partie elle traite les notions de retaillage des tableaux.

Echiquier :

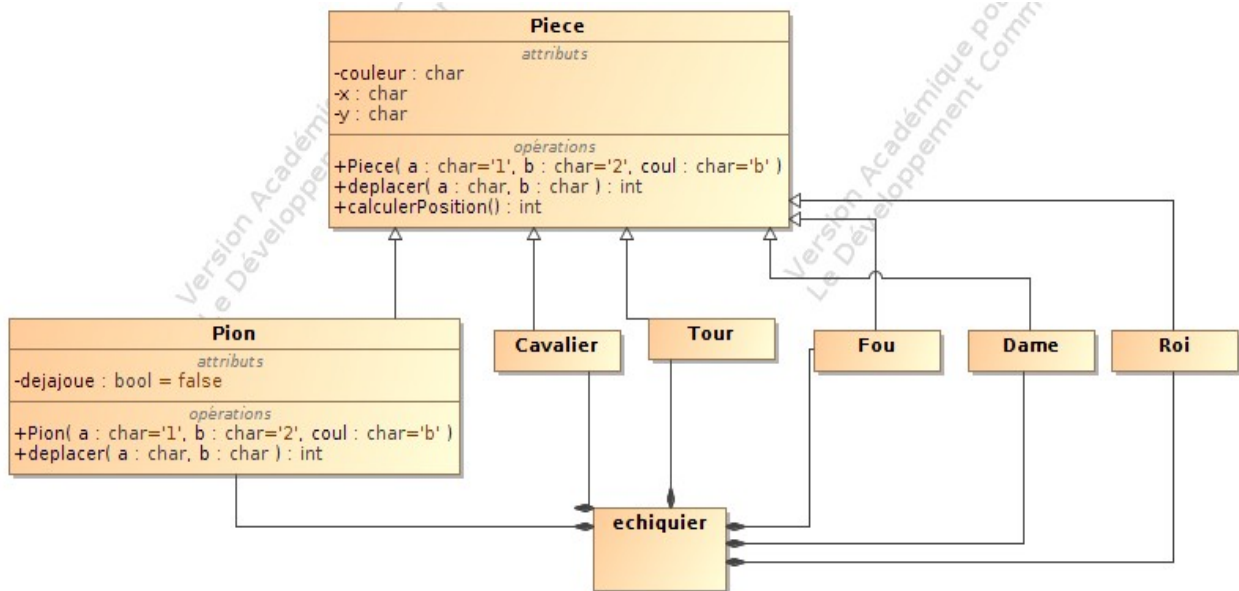
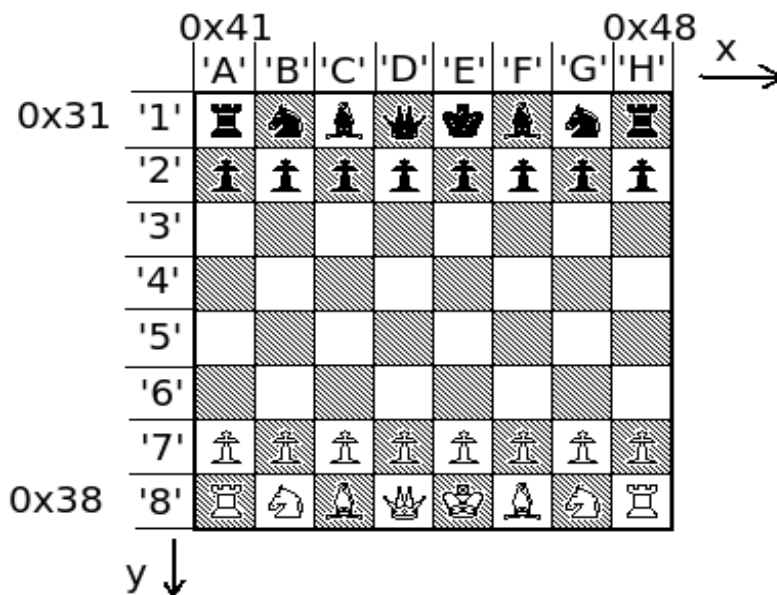


Illustration 1: sous ensemble du diagramme de classe du jeu d'échec.

Dans cette partie nous ne traiterons que les déplacements des différentes pièces sur l'échiquier.

X et y sont les attributs qui définissent la position d'une pièce sur l'échiquier
couleur défini la couleur de la pièce 'b' pour une pièce blanche et 'n' pour une pièce noire.

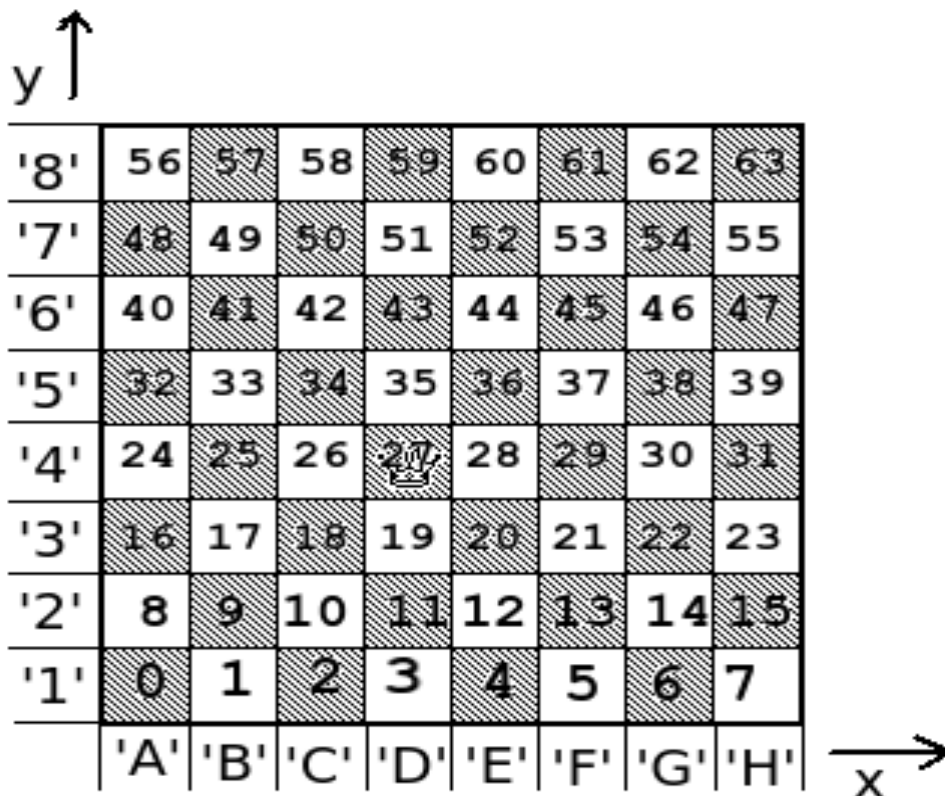
Organisation de l'échiquier.



Différentes cases du jeu d'échec.

Les lignes sont représentées par des lettres allant de 'A' de code ASCII 0x41 à 'H', les colonnes par des chiffres de '1' de code ASCII 0x31 à '8'

le positionnement des pièces :



'8'	56	57	58	59	60	61	62	63
'7'	48	49	50	51	52	53	54	55
'6'	40	41	42	43	44	45	46	47
'5'	32	33	34	35	36	37	38	39
'4'	24	25	26	27	28	29	30	31
'3'	16	17	18	19	20	21	22	23
'2'	8	9	10	11	12	13	14	15
'1'	0	1	2	3	4	5	6	7
	'A'	'B'	'C'	'D'	'E'	'F'	'G'	'H'

Dans la première partie vous créerez les fichiers echiquier.cpp Piece.h et Piece.cpp afin de développer vos différents programmes.

- 1) En vous servant du Illustration 1: sous ensemble du diagramme de classe du jeu d'échec. Réaliser le fichier Piece.h où seront définis les attributs privés x, y et couleur ainsi que le constructeur Piece.

Le constructeur est déclaré de la façon suivante :

```
Piece (char a , char b , char coul );
```

- 2) Réaliser le constructeur Piece dans un fichier Piece.cpp.
NB : les paramètres a, b et coul correspondent respectivement aux attributs x,y et couleur.
Voir le diagramme de classe pour définir le prototype du constructeur de la classe Piece.
Piece(char a, char);

- 3) Instancier dans un programme principal une Piece p de façon dynamique, cette Piece sera initialisée de la façon suivante par défaut:

```
x= 'A' ;  
y='1' ;  
couleur='b' ;
```

- 4) Réaliser le constructeur qui initialise par défaut les attributs avec les valeurs du I.3).
Modifier la déclaration du constructeur pour obtenir le bon résultat (.h)

Montrer votre résultat en utilisant le debugger.

- 5) Quelles sont les valeurs des attributs lorsque dans un programme de test on fait les appels suivants :
 1. `Piece *p1 =new Piece();`
 2. `Piece *p2 =new Piece('D');`
 3. `Piece *p3=new Piece('C','1','n');`
- 6) Réaliser une méthode `afficher()` de la classe `Piece`, qui permettra d'afficher les attributs de cette classe.
- 7) Compléter votre programme principal afin que soit affichés les attributs des différentes pièces `p1`, `p2` et `p3` déclarés ci-dessus.

Montrer votre résultat.

- 8) Réaliser et tester les Getteurs et les Setteurs
- 9) Mettre en œuvre et tester la méthode `calculerPosition()` définie ci dessous :


```
int Piece::calculerPosition(){
    int position=((int)8*(y-0x31)+(int)(x-0x41));
    return position;
}
```
- 10) Réaliser la méthode `int deplacer(char a, char b)`.
 Cette méthode déplacera la pièce sur l'échiquier de `a` et `b`. Dans le cas où la pièce sort de l'échiquier aucun mouvement ne sera possible, cette méthode retourne la position de la pièce sur l'échiquier.

Montrer votre résultat en mettant en œuvre les tests nécessaires à votre démonstration.

Dans la suite la classe `Pion` qui hérite de la classe `Piece` sera placée dans deux fichiers `Pion.h` et `Pion.cpp`. Vous devrez aussi modifier le `Makefile` afin d'intégrer ces fichiers.

- 11) En vous servant du Illustration 1: sous ensemble du diagramme de classe du jeu d'échec. Réaliser le fichier `Pion.h` où seront définis la classe `Pion` qui hérite de la classe `Piece`.
 Les pions blancs seront initialisés sur la ligne '2' et les noirs sur la ligne '7'
 Un attribut membre « `dejajoue` » permettra de définir si le pion a déjà été joué.
 Cet attribut sera initialisée à « `false` » au lancement du programme
 Si cet attribut est à « `false` » alors le pion pourra avancer de deux cases sinon il ne pourra avancer que d'une case.
 (par exemple un pion blanc pourra passer, lors de son premier mouvement, de la case 'B','2' à la case 'B','4' ou 'B','3' et un pion noir de la case 'B','7' vers les cases 'B','5' ou 'B','6').

 Vous réaliserez le constructeur et le destructeur de ce pion initialisé en 'A','2' comme un pion blanc 'b'.
- 12) Est il nécessaire de modifier les attributs de la classe `Piece` pour que la suite du programme fonctionne ?
 Si oui, faites la modification dans le programme de la classe `Piece`.
 Si nécessaire apporter les modifications au niveau du diagramme de classes.
- 13) Pour les tests, réaliser l'initialisation d'un pion blancs et d'un pion noir.

- 14) Réaliser la méthode déplacer. cette méthode permettra d'avancer le pion noir ou blanc d'une case ou de deux cases si c'est possible. Attention l'avancée est différente selon que le pion est blanc ou noir.

Le prototype de cet opérateur sera le même que celui d'une piece :

int déplacer(char a, char b); cette méthode retournera la position finale du pion.

Dans cet exercice vous ne traiterez pas le cas où le pion puisse prendre une autre piece, ce qui signifie que b sera toujours égal à 0.

- 14.a) Quelles sont les valeurs que peut prendre les paramètres « a » et « b » lors de l'appel de la méthode déplacer.

Dans un premier temps vous supposerez que « a » et « b » ne peuvent prendre que ces valeurs.

- 14.b) Dans un premier temps écrire la méthode déplacer en sous entendant que le pion est blanc.

- 14.c) Compléter cette méthode en distinguant les pions blancs et noir.

- 14.d) Compléter cette méthode en faisant intervenir l'attribut dejajoue, si a prend pour valeur 2 au passage de paramètres le pion ne se déplacera pas.

- 14.e) Finir la méthode en faisant en sorte que lorsque « a » et « b » prennent les valeurs du 13a. Aucun déplacement soit possible.

- 15) vous testerez la méthode déplacer de la Classe Pion pour différents cas d'utilisation d'un pion.

Montrer votre résultat en mettant en œuvre les tests nécessaires à votre démonstration.

- 16) Dans le programme principal tous les pions sont d'abord initialisés comme des pièces de la façon suivante :

```
Piece *mpiece;  
Pion *mpion=new Pion();  
mpiece=mpion;  
mpiece->afficher();  
mpiece->deplacer(0,2);  
mpiece->afficher();  
mpiece->deplacer(0,2);  
mpiece->afficher();  
delete mpion;
```

Le pion se déplace t'il comme vous le souhaitez ? Si ce n'est pas le cas faire la modification nécessaire.