Lycée TOUCHARD WASHINGTON

Programmation Orientée Objet

TP de Contrôle

SNir 1

2021-2022

La bataille navale

A - Présentation

La bataille navale est un jeu de société devenu populaire en 1931 lors de son apparition dans les publications américaines de divertissement.

1. Règles du jeu

Comme son nom l'indique, la bataille navale est un jeu consistant à torpiller la flotte de son adversaire. Les joueurs, au nombre de deux, commencent par disposer verticalement ou horizontalement (jamais en diagonale) leur flotte sur leur grille cible, qu'ils sont seuls à voir. Ensuite, tour à tour, ils lancent virtuellement des torpilles sur l'ennemi en indiquant des coordonnées de tir (par exemple B4). L'adversaire doit alors indiquer si l'un de ses bateaux a été atteint ou pas. Si, la torpille rate un bateau, il doit dire "Manqué". En revanche, si elle atteint sa cible, il indique "Touché", lorsque toutes les cases constituant la coque du bateau sont touchées, il dit "Coulé". Le gagnant est celui qui parvient à torpiller complètement l'ensemble des navires de l'adversaire avant que tous les siens le soient.

2. Liste des navires

Chaque joueur possède toujours les mêmes navires. Voici la configuration la plus courante, même si d'autres existent : 1 porte-avions (5 cases), 1 cuirassé (4 cases), 1 croiseur (3 cases), 1 sous-marin (3 cases) et 1 destroyer (2 cases).

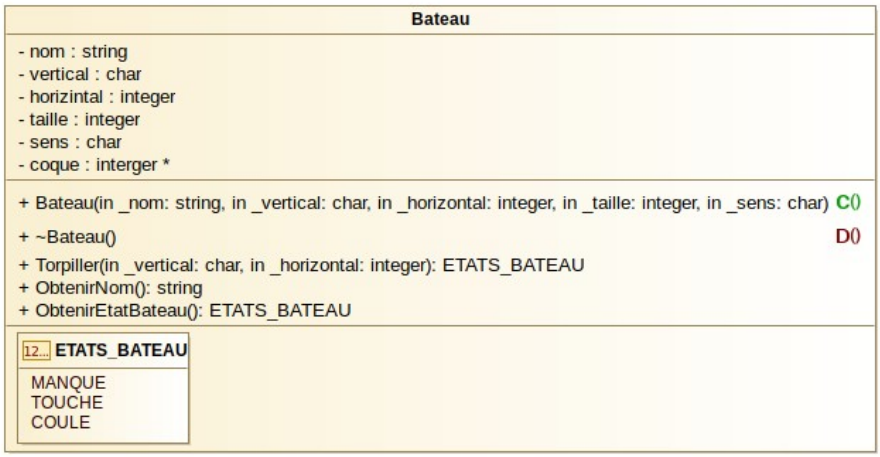
3. Grille de jeu

La grille de jeu est toujours numérotée de A à J verticalement et de 1 à 10 horizontalement. Sur cette grille est positionné un croiseur vertical en C3 et un cuirassé horizontal en H5.

	1	2	3	4	5	6	7	8	9	10
A										
B										
C										
D										
E										
F										
G										
H										
I										
J										

B - Réalisation de la classe Bateau

1. Réalisez la déclaration de la classe **Bateau** en respectant la représentation UML suivante :



   Par défaut, la valeur du paramètre **_sens** dans le constructeur est 'H' pour horizontal. La constante énumérée **ETATS_BATEAU** possède les valeurs **MANQUE**, **TOUCHE** et **COULE**.

2. Écrivez le code du constructeur de la classe, celui-ci initialise les différents attributs à partir des paramètres reçus. L'attribut **etatBateau** reçoit la valeur **MANQUE**. En ce qui concerne le tableau **coque**, il s'agit d'un tableau dynamique dont la taille est fonction du paramètre **_taille**. Chaque case du tableau est ensuite initialisée à 0 pour indiquer que le bateau n'est pas encore touché.
3. En commentaire dans vortre code, justifiez la présence d'un destructeur explicite pour cette classe, donnez le code correspondant.
4. Codez la méthode **Torpiller()** de la classe **Bateau** en utilisant les informations suivantes :
   - Le tableau **coque** est utilisé pour définir l'état d'un bateau placé horizontalement ou verticalement en fonction de la valeur de son attribut **sens** 'H' ou 'V'.
   - Dans ce tableau, **coque[0]** représente toujours la partie la plus à gauche pour les bateaux placés horizontalement et la partie la plus en haut pour ceux placés verticalement (0 pas touché, 1 touché). Ainsi, **coque[0]** définit l'état aux coordonnées horizontale et verticale passées en paramètres au constructeur de la classe. Les cases suivantes contiennent l'état pour le reste du bateau.
   - Après avoir vérifié si le bateau est placé horizontalement ou verticalement, on peut vérifier qu'il y a égalité soit de la lettre dans ce cas on est sur la ligne **α** soit du chiffre dans ce cas on est sur la ligne **β**. Il y a correspondance avec les indices de la coque.

Exemple cuirassé en H5 horizontalement

Exemple cuirassé en H5 verticalement

1er cas, torpillé en **H7**, égalité de la lettre **H**
2e cas, torpillé en **J5**, égalité du chiffre **5**

α	Indice d'un bateau horizontal	5	6	7	8
β	Indice d'un bateau vertical	H	I	J	K
	Indice de la coque	0	1	2	3
	État du bateau	0	0	**1**	0

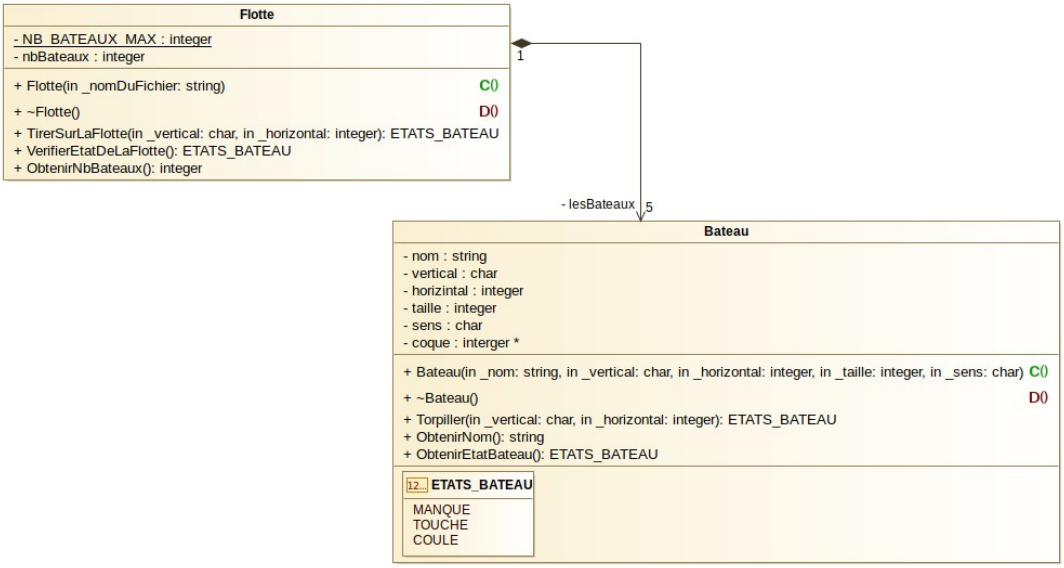
Il est possible de soustraire soit une lettre, soit un chiffre pour retrouver l'indice correspondant. Pour le 1er cas 7 – 5 → **2,** pour le 2e Cas 'J' – 'H' → **2**. Dans les 2 cas, le **2** correspond à l'indice de la coque où le bateau a été touché.

   - Lorsque la somme des éléments du tableau **coque** est égale à la taille du bateau, il est coulé. L'état du bateau change.

5. Codez le programme principal dans le fichier **testBateau.cpp** permettant le test unitaire de la classe **Bateau**, respectez au moins les consignes suivantes :
   - Déclarez dans le programme principal deux instances, un cuirassé et un croiseur comme le montre l'exemple de grille proposée
   - En utilisant la librairie ***iostream***, effectuez la saisie des coordonnées du tir une lettre et un chiffre séparés par un espace jusqu'à ce que le croiseur soit coulé. Le résultat de la fonction **Torpiller()** de la classe **Bateau** est affiché au fur et à mesure des appels.
   - Procédez de manière identique pour le cuirassé.
   - Vérifiez le comportement de votre programme lorsque le tir ne touche aucun bateau.

C - Réalisation de la classe Flotte

On propose à présent un nouveau diagramme de classe faisant apparaître la classe **Flotte** :



1. En commentaire dans votre code, comment se nomme la relation entre la classe **Flotte** et la classe **Bateau** ? Que signifie le 5 à l'extrémité de la flèche ?

On donne ici la déclaration de la classe **Flotte**

```
#include "bateau.h"

#include <fstream>
#include <sstream>
#include <iostream>
using namespace std;

class Flotte
{
public:
    Flotte(const string _nomDuFichier);
    ~Flotte();
    Bateau::ETATS_BATEAU TirerSurLaFlotte(const char _vertical, const int _horizontal);
    Bateau::ETATS_BATEAU VerifierEtatDeLaFlotte();
    int ObtenirNbBateaux() const;


private:
    static const int NB_BATEAUX_MAX = 5 ;
    int nbBateaux;
    Bateau *lesBateaux[NB_BATEAUX_MAX];
};
```

2. Ajoutez la classe **Flotte** à votre projet.

3. Le constructeur est chargé de lire un fichier texte contenant la liste des bateaux à mettre à l'eau ainsi que leurs coordonnées et d'initialiser la partie.

Exemple de fichier :

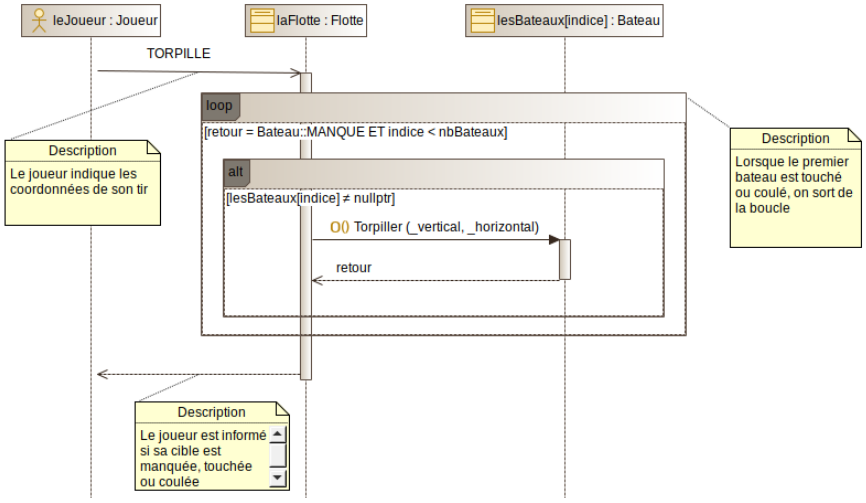
```
Croiseur C3 3 V
Cuirassé H5 4 H
Porte-avions A1 5 H
Sous-marin A10 3 V
Destroyer L9 2 H
```

On donne ici le code du constructeur

```
 Flotte::Flotte(const string _nomDuFichier) :
    nbBateaux(0)
{
    ifstream fichier(_nomDuFichier);
    int indice = 0;
    if(!fichier.is_open())
    {
        cout << "Erreur lors de l'ouverture du fichier " << endl ;
    }
    else
    {
        string nom;
        string position;
        int taille;
        char sens;
        char lettre;
        int chiffre;
        do {

            fichier >> nom >> position >> taille >> sens;
            if(fichier.good() && indice < NB_BATEAUX_MAX)
            {
                lettre = position.front();
                stringstream ss(position.substr(1));
                ss >> chiffre;
                lesBateaux[indice++] = new Bateau(nom, lettre, chiffre, taille, sens);
            }
        } while (!fichier.eof() && indice < NB_BATEAUX_MAX);
        nbBateaux = indice;
    }
    while(indice < NB_BATEAUX_MAX)
    {
       lesBateaux[indice++] = nullptr;
    }
}
```

4. Réalisez le code du destructeur, celui-ci libère la mémoire occupée par un bateau si le pointeur est différent de **nullptr**.
5. En vous inspirant du diagramme de séquence suivant, réalisez la méthode **TirerSurLaFlotte()**

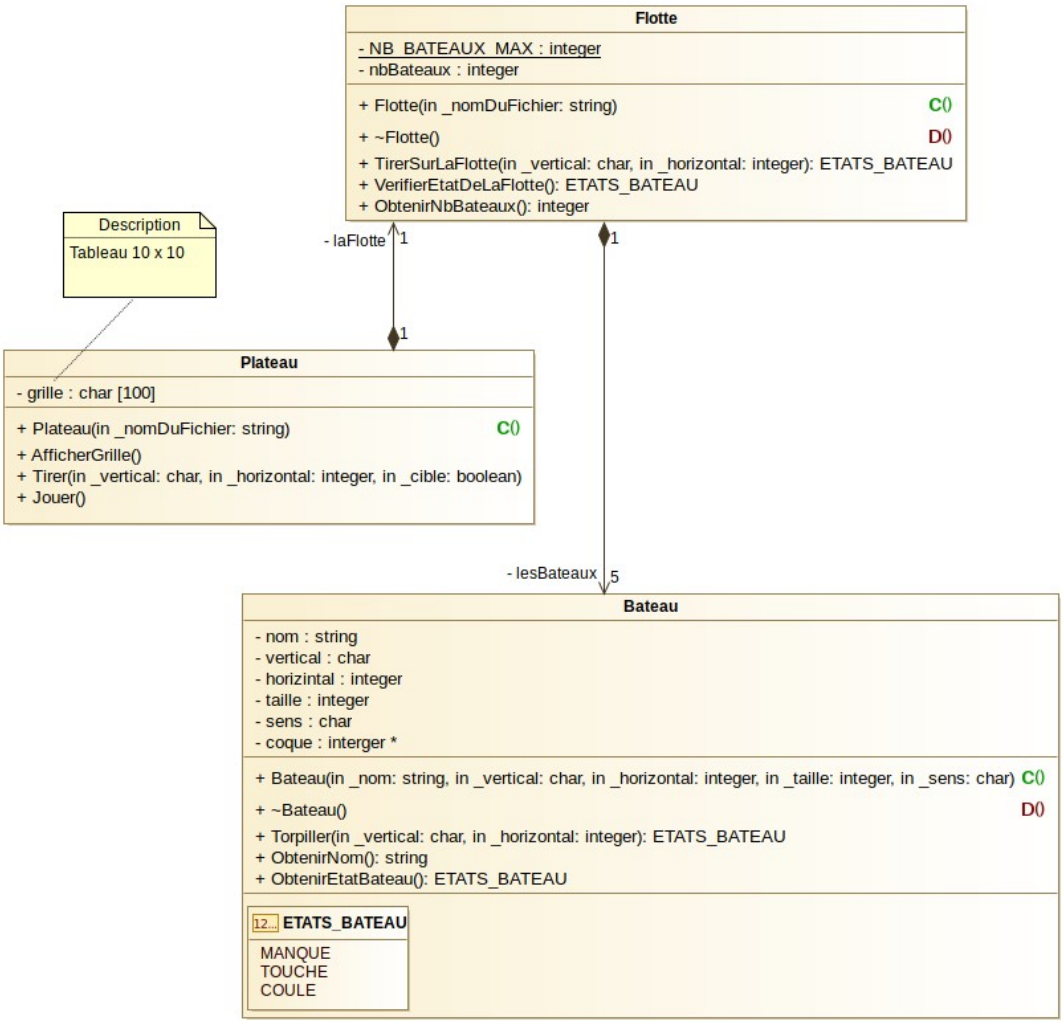


6. De même, réalisez la méthode **VerifierEtatDeLaFlotte().** Elle retourne **Bateau::COULE** lorsque tous les bateaux sont coulés, **Bateau::TOUCHE** si au moins un des bateaux n'est pas coulé et **Bateau::MANQUE** si aucun des bateaux n'est touché.
7. Après avoir retiré de votre projet le précédent programme principal sans l'effacer de votre répertoire, réalisez un nouveau programme principal dans un fichier nommé **testFlotte.cpp** permettant de vérifier le fonctionnement de la flotte pour un seul joueur. Il doit indiquer que la partie est terminée lorsque tous les bateaux sont détruits. Le programme doit fonctionner pour un fichier contenant moins de bateaux, par exemple pour 2. De même, il doit se terminer correctement si le fichier n'a pas été trouvé.

D - Réalisation du plateau

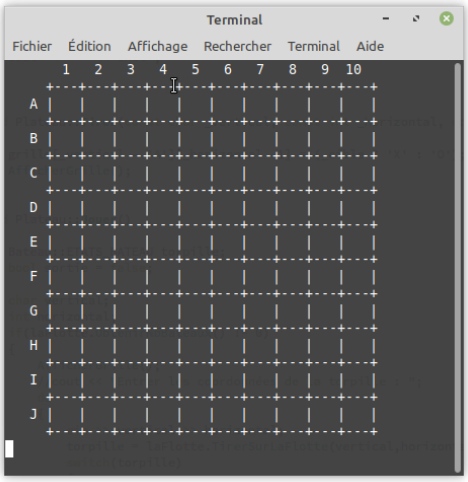
L'objectif est maintenant d'afficher graphiquement la torpille tirée sur une grille. Elle est matérialisé par un 'O' lorsqu'elle est tombée dans l'eau et par un 'X' lorsqu'elle a touché sa cible.

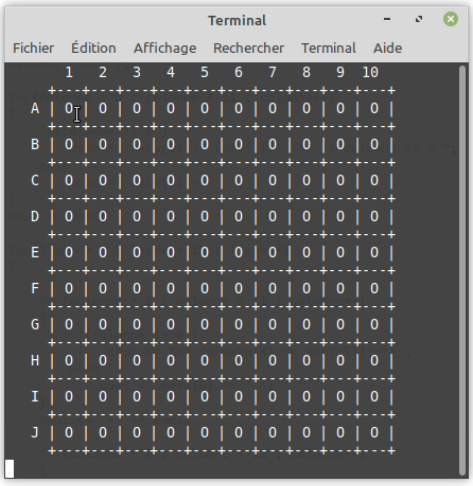
1. Créez une nouvelle classe **Plateau** en vous aidant du diagramme de classes ci-après, elle sera chargée de l'affichage et des interactions avec le joueur :



2. Codez le constructeur de cette classe qui après avoir initialisé le constructeur de la classe **Flotte**, remplit le tableau **grille** à 2 dimensions 10x10 avec le caractère espace.

3. Codez la méthode **AfficherGrille()**, elle affiche sur la console la grille comme le montre la figure suivante. Vous pouvez dans un premier temps remplir la grille avec le caractère 'O' pour bien visualiser chaque case.





Le principe est d'écrire d'abord tous les chiffres. Ensuite, pour chaque ligne, d'écrire la ligne de séparation, de passer à la ligne suivante, d'afficher la lettre et le contenu de la grille pour cette ligne en respectant le format avec le caractère '|'. Il reste, pour finir, à reproduire une fois la ligne de séparation.

Vous utiliserez le flux de sortie **cout** de la librairie **iostream** et les manipulateurs **setw()** et **setfill()** de la librairie **iomanip** pour formater vos affichages.

Vérifier votre affichage avec le programme principal suivant dans le fichier **batailleNavale.cpp**

```
#include "plateau.h"
int main()
{
    Plateau lePlateau("LaFlotte2.txt");
    lePlateau.AfficherGrille();

    return 0;
}
```

4. La méthode **Tirer()** consiste à mettre dans la case correspondante de la grille le caractère 'X' si la cible est touchée ou 'O' dans le cas contraire. Le paramètre **_cible** est vrai lorsque la cible est touchée. Pour terminer, elle appelle la méthode **AfficherGrille()**. Codez cette méthode.
5. Une première version de la méthode **Jouer()** est fournie à la page suivante. Complétez le programme principal pour que votre joueur puisse réaliser une partie de bataille navale contre l'ordinateur. L'emplacement des bateaux est stocké dans le fichier, seul le joueur propose les coordonnées de ses cibles.
6. Modifiez la fonction **Jouer()** pour que le joueur puisse saisir les coordonnées sans espace et que le programme s'arrête en cours de partie si le joueur tape **FIN**. Les lettres pourront être saisies en majuscule ou en minuscule.

Annexe : Code de la méthode **Jouer()**

```
void Plateau::Jouer()
{
    Bateau::ETATS_BATEAU torpille;
    bool sortie = false;
    char vertical;
    int horizontal;
    if(laFlotte.ObtenirNbBateaux() != 0)
    {
        cout << "Entrer les coordonnées de la torpille : ";
        do
        {
            cin >> vertical >> horizontal;
            torpille = laFlotte.TirerSurLaFlotte(vertical,horizontal);
            switch(torpille)
            {
            case Bateau::TOUCHE:
                Tirer(vertical,horizontal,true);
                cout << "Rejouer : " ;
                break;
            case Bateau::COULE:
                Tirer(vertical,horizontal,true);
                cout << "Bateau coulé... ";
                if(laFlotte.VerifierEtatDeLaFlotte() != Bateau::COULE)
                    cout << "rejouer : " ;
                else
                    sortie = true;
                break;
            case Bateau::MANQUE:
                Tirer(vertical,horizontal,false);
                cout << "Rejouer : " ;
                break;
            }
        } while(sortie == false);
    }
    cout << "fin de la partie !" << endl;
}
```