Bataille navale

Il s'agit d'une version simplifiée de la bataille navale. La différence est que les bateaux sont tous d'une case.

La partie se déroulera contre un ordinateur.

Après avoir placé leurs bateaux sur le champ de bataille les adversaires tenteront à tour de rôle de deviner où sont situé les bateaux de leur adversaire.

Exemple d'un champ de bataille avec la position des bateaux :

	1	2	3	4	5
1	~	o	~	~	~
2	o	~	o	~	~
3	~	~	o	~	~
4	~	~	~	~	~
5	~	~	~	~	o

Ici le champ de bataille est de longueur 5 (5 lignes et 5 colonnes). Les bateaux sont placés :

- à la ligne 1 colonne 2
- ligne 2 colonne 1
- ligne 2 colonne 3
- ligne 3 colonne 3
- ligne 5 colonne 5

Le champ de bataille de chaque joueur sera stocké dans un tableau à double dimension [][] (ligne, colonne). Voici les différents états que peut prendre une valeur du tableau :

- 0 = aucune action sur cette case, aucun pion (valeur initiale de toutes les cases) → ?
- 1= un pion non découvert (valeur quand un bateau est sur la case et non découvert) → ?
- 2= un pion découvert (valeur quand un bateau est sur la case et a été découvert par l'adversaire) → o
- 3= une case découverte sans pion (la case a été découverte et il n'y a pas de bateau) → x
- Exemple d'affichage d'un champ de bataille de l'adversaire qui n'a pas été entièrement découvert :

	1	2	3	4	5
1	?	?	?	x	x
2	?	?	o	?	x
3	?	?	o	?	?
4	?	?	?	?	?
5	?	x	x	?	o

le champ de bataille du joueur sera stocké dans le tableau tabjoueur[][].

Le champ de bataille de l'ordinateur sera stocké dans le tableau tabordi [][].

étape 1 : initialisation des tableaux tabjoueur & tabordi

aide : Voici comment déclarer un tableau de 10 lignes et 10 colonnes

```
int tabjoueur[][] = new int [10][10];
int tabordi[][]= new int [10][10];
```

Vous devez ensuite attribuer pour chaque ligne et chaque colonne la valeur 0

étape 2 : on place nos 5 pions : tabjoueur

2.1 Demandez à l'utilisateur à quelle ligne il veut placer son pion

2.2 Demandez à l'utilisateur à quelle colonne il veut placer son pion

Attention l'utilisateur ne doit pas placer un pion :

- dans un endroit déjà pris par un pion (**si l'utilisateur a déjà placé un pion dans cette case, alors la valeur de la case est de 1)**
- dans un endroit hors du périmètre du tableau

2.3 Si le choix est valable, le tableau du joueur devra prendre la valeur 1 pour la colonne et la ligne choisie

2.4. on répète étape 2.1 à 2.3 tant que le joueur n'a pas placé tous les pions.

étape 3 : on place les 5 pions de l'ordinateurs : tabordi

3.1 Attribuer une ligne aléatoirement (entre 1 et 5)

Rappel pour un nombre aléatoire entre 1 et N:
nbrandom = (**int**)(Math.*random*()*(N))+1;

3.2 Attribuer une colonne aléatoirement (entre 1 et 5)

Attention l'ordinateur ne doit pas placer un pion dans un endroit déjà pris par un pion (**si l'ordinateur a déjà placé un pion dans cette case, alors la valeur de la case est de 1)**

3.3 Si le choix est valable, le tableau de l'ordinateur devra prendre la valeur 1 pour la colonne et la ligne choisie (si le choix n'est pas valable on retourne à l'étape 3.1)

3.4. on répète étape 3.1 à 3.3 tant que l'ordinateur n'a pas placé tous les pions.

étape 4 : créer la procédure affichagetab

Cette procédure permet d'afficher un champ de bataille avec les bateaux positionnés. Cette procédure prend en paramètre un tableau d'entier à double dimension ainsi que le nombre de case du plateau.

Exemple d'affichage :

```
    1 2 3 4 5
1   o ~ ~ ~ ~
2   ~ o ~ ~ ~
3   ~ ~ o ~ ~
4   ~ ~ ~ o ~
5   ~ ~ ~ ~ o
```

4.1 Créer la procédure affichagetabJoueur avec les 2 paramètres en entrée.

Formalisme :

```
static void NomProcédure (int tab[][],int nbcase)
{

}
```

4.2 Afficher les caractères espace (le code à effectuer est en surbrillance)

```
    1 2 3 4 5
1   o ~ ~ ~ ~
2   ~ o ~ ~ ~
3   ~ ~ o ~ ~
4   ~ ~ ~ o ~
5   ~ ~ ~ ~ o
```

4.2 Afficher les 5 premiers numéros de case dynamiquement grâce à une boucle (de 1 à nbcase)

```
    1 2 3 4 5
1   o ~ ~ ~ ~
2   ~ o ~ ~ ~
3   ~ ~ o ~ ~
4   ~ ~ ~ o ~
5   ~ ~ ~ ~ o
```

4.3 Pour chaque colonne (nb case) on affiche

- le numéro de ligne
- le pion ou la vague si il n'y a aucun pion
    - si la valeur est 1 alors il faut afficher un pion (symbolisé ici par le caractère "o")
    - si la valeur est 0 alors il ne faut rien afficher (symbolisé ici par le caractère "~")

```
    1 2 3 4 5
1   o ~ ~ ~ ~
2   ~ o ~ ~ ~
3   ~ ~ o ~ ~
4   ~ ~ ~ o ~
5   ~ ~ ~ ~ o
```

4.4 On répète l'étape 4.3 pour toutes les lignes (nbcase)

4.5 Afficher votre Tableau quand vous avez terminé de placer l'ensemble des pions.

Aide : il suffit juste d'appeler votre procédure à la fin de l'étape 2. Une seule ligne est à rajouter donc…

étape 5 : créer la procédure affichagetabCache

Cette affichage permet d'avoir un état des lieux du champ de bataille découvert (cache les cases non découvertes)

Rappel des différents états :

- 0 = aucune action sur cette case, aucun pion (valeur initiale de toutes les cases) → ?
- 1= un pion non découvert (valeur quand un bateau est sur la case et non découvert) → ?
- 2= un pion découvert (valeur quand un bateau est sur la case et a été découvert par l'adversaire) → o
- 3= une case découverte sans pion (la case a été découverte et il n'y a pas de bateau) → x

Exemple :

	1	2	3	4	5
1	?	?	?	x	x
2	?	?	o	?	x
3	?	?	o	?	?
4	?	?	?	?	?
5	?	x	x	?	o

Vous pouvez copier coller la procédure de l'étape 4 et modifier votre code pour que l'affichage se fasse en fonction des états ci-dessus.

étape 6 : Le joueur cherche un pion

6.0 Afficher "A vous de Jouer !"

6.1 Demandez à l'utilisateur à quelle ligne il veut découvrir une case

6.2 Demandez à l'utilisateur à quelle colonne il veut découvrir une case

6.3 Afficher "Tir en cours" et laisser une attente de 2 secondes pour "simuler" le tir
Thread.sleep(nbsec*1000);

6.4 Afficher le résultat.

- Si la case était sans pion (valeur 0) : afficher Raté !
- Si la case était avec pion (valeur 1): afficher Touché ! + incrémenter la variable nbPionTrouvéJoueur.
- Toutes autres valeurs dans la case cela signifie que le joueur avait déjà découvert cette case : afficher Tir à blanc!

6.5 Afficher le tableau ordi : Faites juste un appel à la procédure crée en étape 5

étape 7 : L'ordinateur cherche un pion

7.0 Afficher "Tour de l'ordinateur"

7.1 Attribuer une ligne aléatoirement (entre 1 et 5)

Rappel pour un nombre aléatoire entre 1 et N:
nbrandom = (**int**)(Math.*random*()*(N))+1;

7.2 Attribuer une colonne aléatoirement (entre 1 et 5)

7.3 Attention l'ordinateur ne doit pas choisir une case qui a été découverte (valeur autre que 0 ou 1). Si la case a découvrir est autre que 0 ou 1 alors on retourne à l'étape 7.1

7.4 Afficher "Tir en cours" et laisser une attente de 2 secondes pour "simuler" le tir
Thread.sleep(nbsec*1000);

7.5 Afficher le résultat.

- Si la case était sans pion (valeur 0) : afficher Raté !
- Si la case était avec pion (valeur 1): afficher Touché ! + incrémenter la variable nbPionTrouvéOrdi.

7.6 Afficher le tableau joueur : Faites juste un appel à la procédure crée en étape 5

étape 8 : fin de partie ?

8.1 On répète l'étape 6 & 7 jusqu'à ce qu'un des joueurs n'aient trouvés les 5 pions.

8.2 Afficher le vainqueur

8.3 Demandez de refaire une partie

- **Mettre des commentaires pour rendre le code plus lisible**
- **Ne passer à la sous étape d'après qu'une fois avoir valider cette étape !**
- **Ajouter des affichages pour tester le code**
- **Bien comprendre ce qui est à faire avant de se lancer dans le code**