CHELLE Raymond 18/11/2025

Rapport de TP : Bataille navale

Sommaire :

I. Introduction

L'objectif était de programmer un jeu complet de bataille navale en Java, jouable dans la console contre un ordinateur. La grille est de taille 5×5, chaque joueur possède 5 navires d'une case, et le premier qui coule tous les navires adverses remporte la partie.

II. Le travail à réaliser

Étape 1 – Création et initialisation des grilles

J'ai d'abord créé deux grilles de jeu sous forme de tableaux à deux dimensions de 5 lignes et 5 colonnes : une pour le joueur et une pour l'ordinateur.

```
int[][] tabjoueur = new int[5][5];
int[][] tabordi = new int[5][5];
```

Toutes les cases ont été remplies avec la valeur 0, ce qui représente de l'eau vide. Pour cela, j'ai utilisé deux boucles for imbriquées qui parcourent chaque ligne et chaque colonne.

```
for (int i = 0; i < 5; i++) {
    for (int y = 0; y < 5; y++) {
        tabjoueur[i][y] = 0;
        tabordi[i][y] = 0;
    }
}
```

Étape 2 – Placement des navires par le joueur

Le joueur place ses cinq navires manuellement. À chaque fois, le programme lui demande une ligne et une colonne (entre 0 et 4). Si la saisie est incorrecte (hors limites ou case déjà occupée), le programme repose la question jusqu'à obtenir des coordonnées valides. J'ai utilisé un booléen et une boucle do…while pour gérer ces contrôles de manière claire et efficace.

```
import java.util.Scanner;

public class Bataillenavala {
    public static void main(String[] args) {
        int[][] tabjoueur = new int[5][5];
        Scanner scanner = new Scanner(System.in);

        int ligne;
        int colonne;
        boolean positionValide;


  for (int n = 1; n <= 5; n++) {
      System.out.println("Placement du pion n°" + n);
      positionValide = false;

      while (!positionValide) {
          System.out.print("Choisissez une ligne (0 à 4) : ");
          ligne = scanner.nextInt();

          System.out.print("Choisissez une colonne (0 à 4) : ");
          colonne = scanner.nextInt();

          if (ligne < 0 || ligne >= 5 || colonne < 0 || colonne >= 5) {
              System.out.println("Position hors du tableau. Réessayez.");
          } else if (tabjoueur[ligne][colonne] != 0) {
              System.out.println("Case déjà occupée. Réessayez.");
          } else {
              tabjoueur[ligne][colonne] = 1;
              System.out.println("Pion placé en [" + ligne + "][" + colonne + "]\n");
              positionValide = true;
          }
      }
  }
```

Étape 3 – Placement automatique des navires de l'ordinateur

L'ordinateur place ses propres navires de façon totalement aléatoire grâce à la classe Random. Le programme génère des coordonnées tant

que la case choisie n'est pas libre. Dès qu'une case contenant un 0 est trouvée, il y place un 1. Comme les nombres aléatoires sont générés entre 1 et 5, aucune vérification de dépassement n'est nécessaire.

```
Random random = new Random();

for (int n = 1; n <= 5; n++) {
    positionValide = false;

    while (!positionValide) {
        ligne = random.nextInt(5);
        colonne = random.nextInt(5);

        if (tabordi[ligne][colonne] == 0) {
            tabordi[ligne][colonne] = 1;
            positionValide = true;
        }
    }
}
```

Étape 4 – Affichage complet de la grille

J'ai écrit une procédure nommée affichagetab qui prend une grille en paramètre et l'affiche de façon lisible : les cases vides (0) deviennent des ~ et les navires (1) deviennent des o. La procédure affiche aussi les numéros des lignes et des colonnes pour que le joueur s'y retrouve facilement.

```java
public static void affichagetabJoueur(int[][] tableau, int nbcase) {
    System.out.print("    ");
    for (int y = 1; y <= nbcase; y++) {
        System.out.print(y + "   ");
    }
    System.out.println();

    for (int i = 0; i < nbcase; i++) {
        System.out.print((i + 1) + "  ");
        for (int y = 0; y < nbcase; y++) {
            if (tableau[i][y] == 1) {
                System.out.print("o   ");
            } else {
                System.out.print("~   ");
            }
        }
        System.out.println();
    }
}
```

```java
affichagetabJoueur(tabjoueur, 5);
```

Voici un résultat :

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

Étape 5 – Affichage masqué de la grille adverse

J’ai créé une seconde procédure, affichagetabCache, qui sert à montrer la grille de l’adversaire sans révéler la position de ses navires. Les cases non découvertes apparaissent avec un ?, les cases vides déjà tirées avec un -, et les navires touchés avec un x. Cela rend le jeu plus réaliste et plus intéressant.

```java
public static void affichagetabCache(int[][] tableau, int nbcase) {
    System.out.print("    ");
    for (int y = 1; y <= nbcase; y++) {
        System.out.print(y + "   ");
    }
    System.out.println();

    for (int i = 0; i < nbcase; i++) {
        System.out.print((i + 1) + "  ");
        for (int y = 0; y < nbcase; y++) {
            if (tableau[i][y] == 2) {
                System.out.print("o   "); // pion découvert
            } else if (tableau[i][y] == 3) {
                System.out.print("x   "); // case vide découverte
            } else {
                System.out.print("?   "); // case non découverte
            }
        }
        System.out.println();
    }
}
```

```java
affichagetabCache(tabordi, nbcase);
```

Résultat :

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

Étape 6 – Tour du joueur

J'ai codé une fonction séparée tirJoueur() qui gère entièrement le tir du joueur : demande des coordonnées, vérification de validité, mise à jour de la grille adverse et affichage du résultat (« Touché ! » ou « Raté ! »). La fonction renvoie 1 en cas de touche, ce qui permet d'incrémenter un compteur de navires coulés.

```java
public static int tirJoueur(int[][] tabordi, int nbcase, Scanner scanner) throws InterruptedException {
    System.out.println("\nÀ vous de jouer !");
    System.out.print("Choisissez une ligne (1 à " + nbcase + ") : ");
    int ligne = scanner.nextInt() - 1;
    System.out.print("Choisissez une colonne (1 à " + nbcase + ") : ");
    int colonne = scanner.nextInt() - 1;

    System.out.println("Tir en cours...");
    Thread.sleep(2000);

    if (tabordi[ligne][colonne] == 0) {
        System.out.println("Raté !");
        tabordi[ligne][colonne] = 3;
    } else if (tabordi[ligne][colonne] == 1) {
        System.out.println("Touché !");
        tabordi[ligne][colonne] = 2;
        affichagetabCache(tabordi, nbcase);
        return 1;
    } else {
        System.out.println("Tir à blanc (déjà tiré ici) !");
    }

    affichagetabCache(tabordi, nbcase);
    return 0;
}
```

```java
int nbPionTrouveJoueur = 0;
nbPionTrouveJoueur += tirJoueur(tabordi, 5, scanner);
```

Résultat :

```
À vous de jouer !
Choisissez une ligne (1 à 5) : 3
Choisissez une colonne (1 à 5) : 4
Tir en cours...
Raté !
```

Étape 7 – Tour de l'ordinateur

Le tir de l'ordinateur fonctionne exactement sur le même principe, mais les coordonnées sont choisies aléatoirement. Un second compteur est mis à jour à chaque fois que l'ordinateur touche un navire du joueur

```java
public static int tirOrdi(int[][] tabjoueur, int nbcase, Random random) throws InterruptedException {
    System.out.println("\nTour de l'ordinateur !");
    int ligne;
    int colonne;
    boolean positionValide = false;


    do {
        ligne = random.nextInt(nbcase);
        colonne = random.nextInt(nbcase);
        if (tabjoueur[ligne][colonne] == 0 || tabjoueur[ligne][colonne] == 1) {
            positionValide = true;
        }
    } while (!positionValide);

    System.out.println("L'ordinateur tire sur [" + (ligne + 1) + "][" + (colonne + 1) + "]...");
    Thread.sleep(2000);


    if (tabjoueur[ligne][colonne] == 0) {
        System.out.println("L'ordinateur a raté !");
        tabjoueur[ligne][colonne] = 3;
    } else if (tabjoueur[ligne][colonne] == 1) {
        System.out.println("L'ordinateur a touché un de vos pions !");
        tabjoueur[ligne][colonne] = 2;
        affichagetabJoueur(tabjoueur, nbcase);
        return 1;
    }


    affichagetabJoueur(tabjoueur, nbcase);
    return 0;
```

```java
nbPionTrouveOrdi += tirOrdi(tabjoueur, 5, random);
```

Résultat :

```
L'ordinateur tire sur [2][5] :
L'ordinateur a raté !
```

Étape 8 – Gestion de la fin de partie et possibilité de rejouer

La boucle principale du jeu continue tant qu'aucun des deux joueurs n'a atteint 5 navires coulés. Dès qu'un compteur arrive à 5, le programme affiche un message clair de victoire, puis demande au joueur s'il souhaite faire une nouvelle partie. En cas de réponse positive, toutes les grilles et compteurs sont entièrement réinitialisés pour recommencer à zéro.

```
if (nbPionTrouveJoueur == 5) {
    System.out.println("\n Vous avez gagné ! Tous les pions ennemis sont touchés !");
    break;
}
if (nbPionTrouveOrdi == 5) {
    System.out.println("\n L'ordinateur a gagné ! Tous vos pions ont été détruits !");
    break;
}
System.out.print("\nVoulez-vous rejouer ? (o/n) : ");
char reponse = scanner.next().toLowerCase().charAt(0);
if (reponse == 'o') {
    System.out.println("\nNouvelle partie !");
    main(args); // relance le jeu
} else {
    System.out.println("Fin du jeu. Merci d'avoir joué !");
}
```

Résultat :

```
Placement du pion n°1
Choisissez une ligne (0 à 4) : 3
Choisissez une colonne (0 à 4) : 4
Placement du pion n°2
Choisissez une ligne (0 à 4) : 3
Choisissez une colonne (0 à 4) :
2
Placement du pion n°3
Choisissez une ligne (0 à 4) : 1
Choisissez une colonne (0 à 4) : 4
Placement du pion n°4
Choisissez une ligne (0 à 4) : 3
Choisissez une colonne (0 à 4) : 2
Case déjà occupée. Réessayez.
Choisissez une ligne (0 à 4) : 3
Choisissez une colonne (0 à 4) : 4
Case déjà occupée. Réessayez.
Choisissez une ligne (0 à 4) : 5
Choisissez une colonne (0 à 4) : 1
Position hors du tableau. Réessayez.
Choisissez une ligne (0 à 4) : 4
Choisissez une colonne (0 à 4) : 1
Placement du pion n°5
Choisissez une ligne (0 à 4) : 2
Choisissez une colonne (0 à 4) : 1

À vous de jouer !
Choisissez une ligne (1 à 5) : 3
Choisissez une colonne (1 à 5) : 4
Tir en cours...
Raté !
    1   2   3   4   5
1   ?   ?   ?   ?   ?
2   ?   ?   ?   ?   ?
3   ?   ?   ?   x   ?
4   ?   ?   ?   ?   ?
5   ?   ?   ?   ?   ?

Tour de l'ordinateur !
L'ordinateur tire sur [2][1]...
L'ordinateur a raté !
    1   2   3   4   5
1   ~   ~   ~   ~   ~
2   x   ~   ~   ~   o
3   ~   o   ~   ~   ~
4   ~   ~   o   ~   o
5   ~   o   ~   ~   ~

À vous de jouer !
Choisissez une ligne (1 à 5) :
```

III. Conclusion

J'ai ainsi réussi à réaliser en java un jeu de bataille navale entièrement fonctionnel. Ce TP m'a permis de bien approfondir la manipulation des tableaux à deux dimensions, la création de procédures et de fonctions réutilisables, la gestion rigoureuse des saisies utilisateur, l'utilisation du hasard avec Random, et surtout la structuration claire d'un programme long et complet. Je suis très satisfait de ce que j'ai fait, ça m'a pris environ 5h car j'ai trouvé ce TP relativement compliqué.