Compte rendu de TP

Sujet : Compte rendu bataille naval

Nom : MEDJAHED

Enseignant : Mme.Beauvallet

Sommaire

Introduction

Dans le cadre du module « Bases de la programmation » en BTS SIO, il nous a été demandé de développer une version simplifiée du célèbre jeu Bataille Navale en langage Java.
L'objectif principal de ce TP est d'apprendre à manipuler les tableaux à double dimension, à structurer un programme en plusieurs étapes, et à gérer des interactions successives avec l'utilisateur.

Ce projet permet également de travailler la logique algorithmique, la vérification des saisies, l'utilisation de valeurs aléatoires ainsi que la mise en place de procédures d'affichage.
La réalisation de ce jeu constitue ainsi un excellent entraînement à la construction de programmes plus complexes tout en consolidant les compétences étudiées lors des cours précédents.

Présentation du projet

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

Le projet consiste à programmer une version simplifiée du jeu « Bataille Navale » en langage Java.
Dans cette version, chaque bateau est représenté par un pion occupant une seule case, et chaque joueur dispose de 5 pions placés sur un plateau de 5 lignes et 5 colonnes.

Le but du jeu est de découvrir l'emplacement des 5 pions adverses avant que l'adversaire ne trouve ceux du joueur. La partie se joue tour par tour : le joueur tire sur une case, puis l'adversaire tire à son tour. L'ordinateur place ses pions aléatoirement et joue également de façon aléatoire.

Ce TP permet de mettre en pratique les notions vues en cours : tableaux à deux dimensions, saisis utilisateurs, générateur de nombres aléatoires, conditions, boucles, procédures d'affichage et logique de jeu.

Analyse du sujet

Avant d'écrire le programme, il est essentiel d'analyser les besoins fonctionnels. Le sujet s'organise en plusieurs étapes allant de l'initialisation des tableaux jusqu'à la détection de la fin de partie.

Trois tableaux en deux dimensions sont nécessaires :

- tabJoueur[][] : stockage des pions placés par le joueur.
- tabOrdi[][] : stockage des pions placés par l'ordinateur.
- tabBataille[][] : tableau représentant l'état des cases découvertes.

Les cases peuvent prendre l'un des quatre états suivants :

- 0 : case jamais découverte et vide
- 1 : case contenant un pion non découvert
- 2 : case contenant un pion qui a été touché
- 3 : case vide déjà tirée

Le sujet impose également plusieurs règles :

- Les coordonnées doivent être comprises entre 1 et 5.
- Un pion ne peut pas être placé sur une case déjà occupée.
- Un tir déjà effectué ne doit pas être rejoué.
- La partie se termine lorsque les 5 pions d'un des joueurs ont été trouvés.

Cette analyse permet de structurer correctement le code avant de passer à l'implémentation.

Structure générale du programme

Le programme est organisé en plusieurs grandes étapes successives adaptées aux consignes du TP :

- Initialisation
  Les tableaux sont créés et remplis avec la valeur initiale 0.
- Placement des pions du joueur
  Une boucle impose au joueur de saisir des coordonnées valides et non occupées.
- Placement des pions de l'ordinateur
  L'ordinateur choisit des positions aléatoires, en évitant les doublons.
- Affichage des plateaux
  Deux procédures permettent d'afficher :
    - le plateau du joueur avec ses pions visibles,
    - le plateau adverse en mode caché
- Boucle principale du jeu
  Le joueur tire, puis l'adversaire tire à son tour, jusqu'à ce que l'un trouve les 5 pions adverses.
- Fin de partie
  Lorsque l'un des joueurs atteint 5 pions touchés, la partie s'arrête et le programme affiche le gagnant.

```java
import java.util.Scanner;

public class bataillenaval {

    public static final String S_RESET = "\u001B[0m";
    public static final String S_RED = "\u001B[31m";
    public static final String S_GREEN = "\u001B[32m";

    public static void main(String[] args) throws InterruptedException {

        Scanner scanner = new Scanner(System.in);

        int tabJoueur[][] = new int[5][5];
        int tabOrdi[][] = new int[5][5];
        int tabBataille[][] = new int[5][5];

        int nbPion = 5;

        System.out.println("Veuillez placer vos pions.");
```

Placement des pions

Le programme demande au joueur de choisir une ligne et une colonne entre 1 et 5.
Le code vérifie deux conditions :

1. La coordination doit être dans les limites du plateau.

2. La case ne doit pas déjà contenir un pion.

Si la position est valide, la case correspondante dans le tableau du joueur est marquée avec la valeur 1.
Cette opération se répète jusqu'à ce que les 5 pions soient positionnés.

Placement des pions de l'ordinateur

L'ordinateur place lui aussi 5 pions, mais de manière automatique et aléatoire.
À chaque tentative, une ligne et une colonne comprises entre 1 et 5 sont tirées au sort.
Le programme vérifie que la case n'est pas déjà occupée avant de placer le pion.

```
Ligne (1 à 5) : 1
Colonne (1 à 5) : 2
4 pions restants.
Ligne (1 à 5) : 2
Colonne (1 à 5) : 3
3 pions restants.
Ligne (1 à 5) : 3
Colonne (1 à 5) : 4
2 pions restants.
Ligne (1 à 5) : 4
Colonne (1 à 5) : 5
1 pions restants.
Ligne (1 à 5) : 5
Colonne (1 à 5) : 1
```

```
Plateau du joueur :
    1  2  3  4  5
1 - ~  o  ~  ~  ~
2 - ~  ~  o  ~  ~
3 - ~  ~  ~  o  ~
4 - ~  ~  ~  ~  o
5 - o  ~  ~  ~  ~

Tous les pions sont placés !
```

Affichage des plateaux

Deux fonctions d'affichage ont été développées :

Affichage du plateau du joueur

Ce plateau montre :

- o pour les pions du joueur
- ~ pour les cases vides

Ce mode est utile au joueur pour visualiser sa propre grille.

Affichage du plateau adverse (caché)

Le joueur ne peut pas voir les pions de l'adversaire directement. Le tableau affiché utilise donc :

- ? pour une case non encore tirée
- o si un pion a été trouvé (touché)
- x si la case tirée était vide (raté)

Ces affichages sont utilisés tout au long du jeu pour suivre l'évolution de la partie.

```
Votre plateau :
    1  2  3  4  5
1 - x  x  x  x  x
2 - x  x  o  x  o
3 - x  x  x  x  o
4 - o  x  o  ?  x
5 - x  x  x  ?  x
```

```
État du plateau adverse :
    1  2  3  4  5
1 - ?  x  x  x  x
2 - x  x  x  x  x
3 - ?  ?  o  ?  ?
4 - x  x  o  x  o
5 - ?  x  ?  ?  ?
```

Gestion des tirs du joueur et gestion des tirs de l'ordinateur

Tir du joueur

À chaque tour, le joueur entre une ligne et une colonne.
Le programme vérifie :

- si les coordonnées sont valides,
- si la case n'a pas déjà été tirée.

Selon la valeur dans `tabOrdi[][]`, le résultat est :

- Touché → si la case contient un pion (1), devient 2
- Raté → si la case était vide (0), devient 3
- Tir à blanc → si la case avait déjà été découverte

Le tableau `tabBataille[][]` est mis à jour pour refléter le résultat, puis affiché en mode caché.

Tir de l'ordinateur

L'ordinateur tire aléatoirement sur le plateau du joueur.
Le programme s'assure qu'il ne rejoue pas une case déjà testée.
Le fonctionnement est identique :

- 1 → touché
- 0 → raté

Le tableau du joueur est ensuite réaffiché en mode caché pour montrer les impacts adverses.

```
Votre tir - Ligne (1 à 5) : 1
Votre tir - Colonne (1 à 5) : 2
Tir en cours...
Touché !

État du plateau adverse :
    1  2  3  4  5
1 - ?  o  ?  ?  ?
2 - ?  ?  ?  ?  ?
3 - ?  ?  ?  ?  ?
4 - ?  ?  ?  ?  ?
5 - ?  ?  ?  ?  ?
```

```
Tour de l'ordinateur...
L'ordinateur tire en 3,1
Raté par l'ordinateur.

Votre plateau :
    1  2  3  4  5
1 - ?  ?  ?  ?  ?
2 - ?  ?  ?  ?  ?
3 - x  ?  ?  ?  ?
4 - ?  ?  ?  ?  ?
5 - ?  ?  ?  ?  ?
```

Fin de partie

La partie continue tant que :

- le joueur n'a pas trouvé les 5 pions de l'ordinateur
- ET que l'ordinateur n'a pas trouvé les 5 pions du joueur

Dès que l'un des deux atteint **5 pions touchés**, la boucle principale se termine et le programme affiche le résultat :

- « Vous avez gagné ! »
  ou
- « L'ordinateur a gagné… »

Il est éventuellement possible d'ajouter une option pour relancer une partie.

```
Votre plateau :
    1  2  3  4  5
1 - x  o  x  x  ?
2 - x  x  o  x  ?
3 - x  x  x  o  x
4 - x  x  x  x  o
5 - o  x  x  x  x

L'ordinateur a gagné...
```

Conclusion

La réalisation de ce TP de Bataille Navale m'a permis de mettre en pratique plusieurs notions importantes : la manipulation des tableaux 2D, la gestion des entrées utilisateur, les conditions, les boucles et l'affichage dynamique.
Même si certaines étapes se sont révélées difficiles, notamment la structure générale du programme et la logique des différents états d'une case, j'ai pu m'appuyer sur les exercices précédents, en particulier celui du jeu **Shifumi**, pour comprendre plus facilement comment organiser mon code.

J'ai également utilisé l'IA de manière intelligente, principalement comme **outil d'aide à la compréhension**, pour vérifier certaines parties de mon code et m'assurer que je suivais correctement les consignes du TD.
Je n'ai pas utilisé l'IA pour « tricher », mais pour apprendre plus efficacement, clarifier certains points et corriger mes erreurs lorsque j'étais bloqué.

Ce TP m'a permis de progresser en Java, de renforcer mon autonomie et de mieux comprendre comment structurer un programme complet.

ANNEXES

bataillenaval_code.pdf