"Tic Tac Toe"
Ou le jeu du Morpion

TIC TAC TOE

Ce TP a été le premier long exercice sur lequel j'ai pu travailler en Java.
Il nous a été donné par Mr Lefebvre Pierre, Professeur de programmation Python, Java et C.

Enoncé :

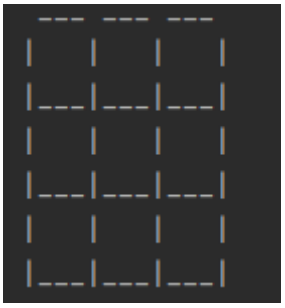
Vous devez recréer le jeu du morpion.

Pour ce faire il faudra suivre quelques règles :

- il doit y avoir 2 joueurs
- le joueur qui aligne 3 de ses symboles gagne
- si 9 coups sont joués sans qu'il n'y ai de vainqueur, il y aura match nul.

Présentation d’une partie :

Mise en place de la partie :



Début de partie :

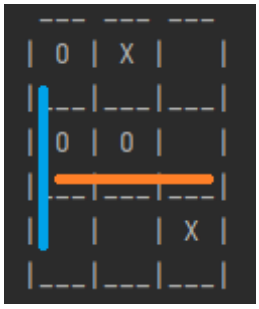
```
   ___ ___ ___
  |   |   |   |
  |___|___|___|
  |   |   |   |
  |___|___|___|
  |   |   |   |
  |___|___|___|
tour n°1
Joueur 1
donnez la ligne (1-3):
1
donnez la colonne (1-3):
1

   ___ ___ ___
  | O |   |   |
  |___|___|___|
  |   |   |   |
  |___|___|___|
  |   |   |   |
  |___|___|___|
tour n°2
Joueur 2
donnez la ligne (1-3):
```

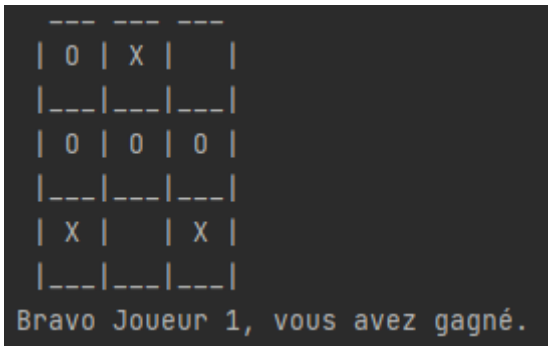
Le joueur 1 à demandé une case déjà sélectionnée :

```
  --- --- ---
 | O | X |   |
 |___|___|___|
 |   | O |   |
 |___|___|___|
 |   |   | X |
 |___|___|___|
tour n°5
Joueur 1
donnez la ligne (1-3):
1
donnez la colonne (1-3):
2
case déjà prise, essaie encore.
  --- --- ---
 | O | X |   |
 |___|___|___|
 |   | O |   |
 |___|___|___|
 |   |   | X |
 |___|___|___|
tour n°5
Joueur 1
donnez la ligne (1-3):
```

Le joueur 2 comprend qu’il va perdre car le joueur 1 a deux possibilités :

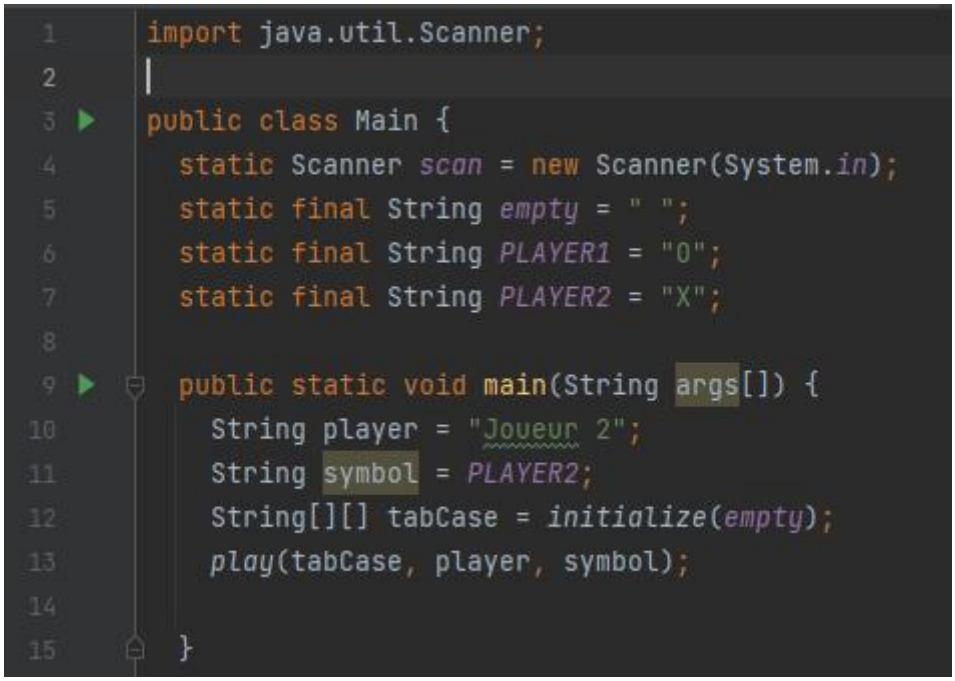


Fin de partie :



Le code source et leur action

La classe Main :



```java
import java.util.Scanner;

public class Main {
  static Scanner scan = new Scanner(System.in);
  static final String empty = " ";
  static final String PLAYER1 = "O";
  static final String PLAYER2 = "X";

  public static void main(String args[]) {
    String player = "Joueur 2";
    String symbol = PLAYER2;
    String[][] tabCase = initialize(empty);
    play(tabCase, player, symbol);

  }
```

- Déclaration des méthodes réutilisées durant tout le code

- Mise en place des méthodes player, symbol et du tableau tabCase :
  player équivaut au joueur dont c'est le moment de jouer
  symbol : X ou 0
  tabCase : le tableau de jeu sur lequel la partie se déroule

- Appel la fonction initialize() avec paramètre empty

- Appel la fonction play

La fonction play :

```java
public static void play(String[][] tabCase, String player, String symbol) {
  int tour = 1;
  String win = " ";
  while (win != "Joueur 1" & win != "Joueur 2" & tour != 9) {
    String caseEmpty = player;
    player = (player == "Joueur 1") ? "Joueur 2" : "Joueur 1";
    symbol = (symbol == PLAYER1) ? PLAYER2 : PLAYER1;
    while (caseEmpty != empty) {
      display(tabCase);
      System.out.print("tour n°");
      System.out.println(tour);
      System.out.println(player);
      System.out.println("donnez la ligne (1-3):");
      int ligne = scan.nextInt() - 1;
      System.out.println("donnez la colonne (1-3):");
      int colonne = scan.nextInt() - 1;
      if (tabCase[ligne][colonne] == empty) {
        caseEmpty = empty;
        tabCase[ligne][colonne] = (player == "Joueur 1") ? PLAYER1 : PLAYER2;
      } else {
        System.out.println("case déjà prise, essaie encore.");
        caseEmpty = player;
      }
    }
    win = victory(tabCase, player, symbol);
    tour++;
  }
}
```

- Déclaration des paramètres tour et win :
  tour compte le nombre de tours dans une partie
  win peut être vide ou contenir joueur 1 ou joueur 2

- La première boucle While continue la partie tant que personne n'a gagné.
  la condition tour sert de sécurité

- caseEmpty permet de connaître si un joueur à déjà placé un pion dessus ou si la case est vide
- player est un opérateur ternaire pour changer de joueur
- symbol est aussi un ternaire, il change le symbole

- La deuxième boucle While écrit les instructions et vérifie que la case n’est pas déjà prise

- Appel de la fonction win() avec pour paramètres tabCase, player et symbol

- Tour incrémente de 1

Fonction Victory :

```java
public static String victory(String[][] tabCase, String player, String symbol) {
  String win = null;
  for (int i = 0; i < 3; i++) {
    if (tabCase[i][0] == symbol) {
      if (tabCase[i][1] == symbol) {
        if (tabCase[i][2] == symbol) {
          display(tabCase);
          System.out.println("Bravo " + player + ", vous avez gagné.");
          win = player;
        }
      }
    }
    if (tabCase[0][i] == symbol) {
      if (tabCase[1][i] == symbol) {
        if (tabCase[2][i] == symbol) {
          display(tabCase);
          System.out.println("Bravo " + player + ", vous avez gagné.");
          win = player;
        }
      }
    }
  }
  if (tabCase[0][0] == symbol) {
    if (tabCase[1][1] == symbol) {
      if (tabCase[2][2] == symbol) {
        display(tabCase);
        System.out.println("Bravo " + player + ", vous avez gagné.");
        win = player;
      }
    }
  }
  if (tabCase[0][2] == symbol) {
    if (tabCase[1][1] == symbol) {
      if (tabCase[2][0] == symbol) {
        display(tabCase);
        System.out.println("Bravo " + player + ", vous avez gagné.");
        win = player;
      }
    }
  }
  return win;
}
```

La fonction victory permet de définir s'il y a un vainqueur ou non
il retournera le joueur gagnant ou null s'il n'y en a pas

Fonctions initialize et display :

```java
public static String[][] initialize(String initChar) {
    return new String[][] { { initChar, initChar, initChar }, { initChar, initChar, initChar },
            { initChar, initChar, initChar } };
}

public static void display(String[][] tabCase) {
    System.out.println("  ___ ___ ___ ");
    for (int i = 0; i < tabCase.length; i++) {
        for (int j = 0; j < tabCase[i].length; j++) {
            System.out.print(" | ");
            System.out.print(tabCase[i][j]);
        }
        System.out.println(" |");
        System.out.println(" |___|___|___|");
    }
}
}
```

La fonction initialize définie le contenu du tableau
La fonction display affiche le tableau de jeu ainsi que les pions s'il y en a.

Ce que ce projet m’a apporté :

Avec cet exercice, j’ai pu commencer à appréhender ce qu’est un projet, ses difficultés autant en termes de temps que de code.
Je n'étais pas vraiment habitué à coder en java et ça n'a pas été simple mais c'était gratifiant d'arriver à faire un jeu par soi-même.

Ce qui pourrait être amélioré :

- Refactoriser de façon orientée objet
- Donner la possibilité de se donner un pseudo
- Faire apparaître les valeurs des abscisses et ordonnées
- Passer l’abscisse de chiffres à lettres pour plus de compréhension
- Mettre en place un système de nouvelle partie
- Compter les points et les conserver (score)
- Enregistrer les pseudos et leurs points
- Faire une interface utilisateur