# Introduction au projet

Temps de lecture : 3 minutes

## Objectifs du jeu

L'objectif du chapitre est de programmer un morpion très simple.

Il faut afficher la grille du jeu et faire jouer deux joueurs qui vont mettre successivement une croix et un cercle.

Le premier joueur qui arrive à faire une ligne gagne. Il faut aussi prévoir le cas où il y a égalité.

## Mise en place du jeu

### Le programme principal

Voici la première étape de la méthode `main` :

```
public class Main {
    public static void main(String[] args) {

        final var scanner = new Scanner(System.in);
        final var game = new TicTacToe();

        while (true) {
            System.out.println(game);
            System.out.println("Veuillez saisir un des chi
ffres [1-9] :");
            final var playerInput = scanner.nextInt();
        }
    }
}
```

Comme pour le projet du Pendu, nous allons utiliser la classe du `Scanner` pour les entrées utilisateur. Attention, nous allons avoir besoin de saisir un chiffre entre 1 et

9, il faudra donc par la suite gérer les cas d'erreurs pour toute saisie non attendue. Ce sera l'occasion de mettre en pratique la gestion des exceptions.

Ensuite, nous utiliserons une classe `TicTacToe` qui sera responsable d'encapsuler les données du jeu ainsi que les comportements du jeu.

Pour finir, la bouche `while (true)` nous permet de définir la boucle principale du jeu.

A chaque tour du jeu, un joueur entrera un chiffre correspondant à l'une des cases (0 à 9). Il faudra ensuite remplir la case avec son symbole (croix ou rond) et passer au deuxième joueur.

Il faudra également vérifier les conditions de victoire ou de match nul lors de chaque tour.

## La classe `TicTacToe`

Voici la première étape de cette classe :

```
public class TicTacToe {

    private char[][] grid = new char[][]{
        {'.', '.', '.'},
        {'.', '.', '.'},
        {'.', '.', '.'}
    };

    @Override
    public String toString() {
        final var builder = new StringBuilder();
        builder.append("Grille du Morpion : ").append(LINE
_SEPARATOR);
        for (char[] line : grid) {
            for (char cell : line) {
                builder.append(SPACE).append(cell).append
(SPACE);
            }
            builder.append(LINE_SEPARATOR);
        }
        return builder.toString();
```

```
        }
    }
```

L'attribut `grid` est un tableau de caractère à deux dimensions de taille 3x3, ce qui nous permet de définir la grille de jeu du Morpion. Chaque cellule est initialisée avec un `'.'` ce qui représente une cellule vide qui n'a pas encore été remplie par un des joueurs.

Ensuite, la méthode `toString` nous permet de définir la fonction d'affichage de la grille de jeu qui sera appelée à chaque tour de jeu pour montrer son évolution.

*À propos de `StringBuilder` : Cette classe nous permet de construire une chaine de caractère. Pour résumer l'intérêt de cette classe : `StringBuilder` permet de concaténer des chaînes de caractères de manière optimisée. On alloue un gros bloc de mémoire dès le début et on ajoute au fur et à mesure des caractères dans ce bloc. Une fois la chaîne de caractère complètement construire avec le Builder, on peut la construire avec la méthode `toString` de `StringBuilder`.*

Pour parcourir la grille de jeu, nous utilisons deux boucles `for` afin de parcourir les deux dimensions du tableau.

Pour afficher des espaces et des sauts de ligne, nous utilisons des constantes que nous définissons dans une classe de constante séparée :

```
public class StringConstant {
    public static final String LINE_SEPARATOR = System.lin
eSeparator();
    public static final String SPACE = " ";
}
```