TP .NET JEUX du morpion

Le **morpion** est un jeu de réflexion se pratiquant à deux joueurs au tour par tour et dont le but est de créer le premier un alignement sur une grille.

Régle du jeu :

Les joueurs inscrivent tour à tour leur symbole sur une grille qui n'a pas de limites ou qui n'a que celles du papier sur lequel on joue. Le premier qui parvient à aligner trois de ses symboles horizontalement, verticalement ou en diagonale gagne un point.

Le morpion donne un avantage assez important à celui qui commence. Des formes évoluées existent, comme le Gomoku ou le Pente, qui ajoutent à la notion d'alignement une notion de prise. Le Renju prévoit des handicaps pour le joueur qui commence, ce qui permet d'équilibrer les chances des deux joueurs. Une partie dure environ une minute

Objectif du TP :

Le TP à pour objectif la réalisation du morpion.

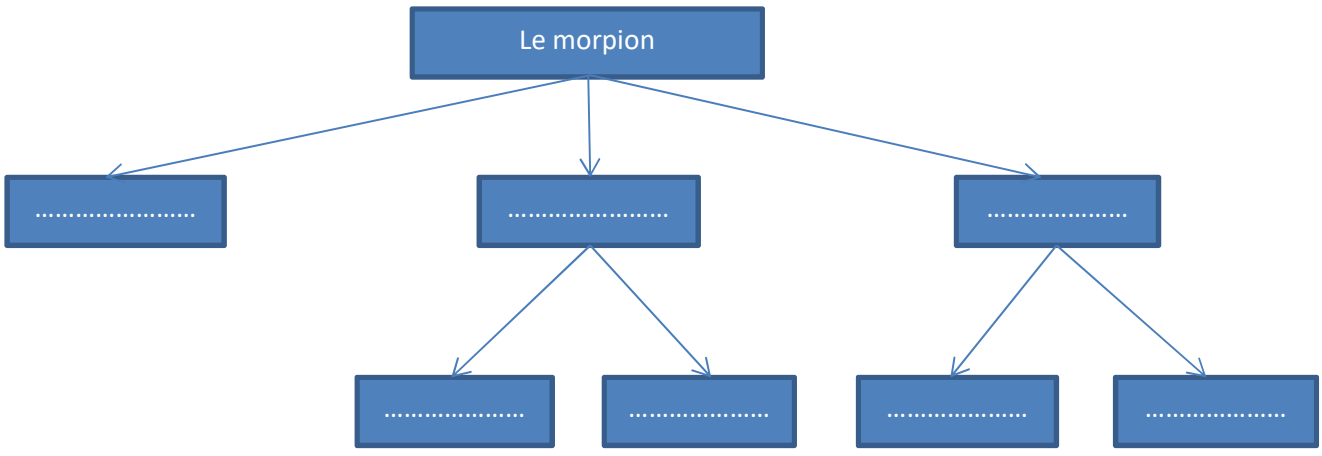
But du TP :

Le but du TP la décomposition d'un programme en plusieurs parties et l'apprentissage de la programmation procédurale.

Partie 1 :

Pour la réalisation du jeu on cherche à décomposé le en plusieurs partie équivalente pour chaque fonctionnalité, pour une bonne visibilité du programme et évité les répétitions du code.

Veuillez remplir le schéma suivant :



Partie 2 :

Pour la partie 2 on essaye de réaliser le programme suivant la décomposition faite dans la première partie.

Pour la réalisation de ce jeu on utilisera une fonctionnalité qui n'a pas été vue en cour une Enum cette fonctionnalité nous aidera à déterminé l'état de chaque case de notre grille.

```
enum EtatCase
{
    Vide,
    Rond,
    Croix
}
```

Pour mémorisé notre grille on aura besoin d'un tableau d'EtatCase appelé grille et d'un générateur aléatoire pour que l'ordinateur puisse choisir lui-même la case.

```
static EtatCase[,] grille; // grille de 3 * 3 cases
static Random generateur; // générateur aléatoire
```

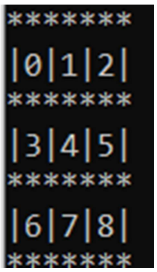
1. la fonctionnalité de l'affichage :
   Pour affiche la grille on doit réaliser une méthode qu'on appellera **AfficherGrille**, cette méthode ne prend aucun paramètre et ne fait aucun retour.

```
private static void AfficherGrille()
{

}
```

   Cette fonction sera faite en deux étapes :

   1. Première étape :
      On affiche une grille vide.
      Deuxième étape :
      On affiche une grille avec l'état de chaque case si Etat case est vide la grille doit comporter le numéro de la case qui est compris en 1 et 9



2. la fonctionnalité de l'ordinateur joue :
   Cette fonctionnalité permet à l'ordinateur Choisir aléatoirement une case vide. On appellera cette méthode **ChoisirCaseOrdinateur**, cette méthode ne prend aucun paramètre et ne fait aucun retour.

L'idée dans cette fonctionnalité est de créer une boucle jusqu'a ce qu'on trouve une case vide, le numéro de la ligne et de la colonne seront choisie aléatoirement.

```
  private static void ChoisirCaseOrdinateur()
  {

        }
```

3. La fonctionnalité de joueur joue :
   Cette fonctionnalité permet à l'utilisateur de choisir sa case, qui doit être libre, et y met une croix. On appellera cette méthode **ChoisirCaseUtilisateur**, cette méthode ne prend aucun paramètre et ne fait aucun retour.

   Pour cette fonctionnalité on boucle jusqu'à ce que le choix de l'utilisateur soit correct.

```
private static void ChoisirCaseUtilisateur()
{

      }
```

4. La fonctionnalité vérification du jeu gagnant :
   Cette fonctionnalité détermine si le joueur passé en paramètre a gagné ou pas, On appellera cette méthode **JeuGagnant**, cette méthode ne prend un paramètre EtatCase et nous retourne un bool.

   Cette fonctionnalité peut être développée en 3 parties :
   La première on vérifie si une ligne est gagnante.
   La deuxième on vérifie si une colonne est gagnante.
   La troisième on vérifie si une diagonale est gagnante.
   Si oui on retourne Vrai sinon on retourne Faux.

```
private static bool JeuGagnant(EtatCase etatCase)
{

      }
```