

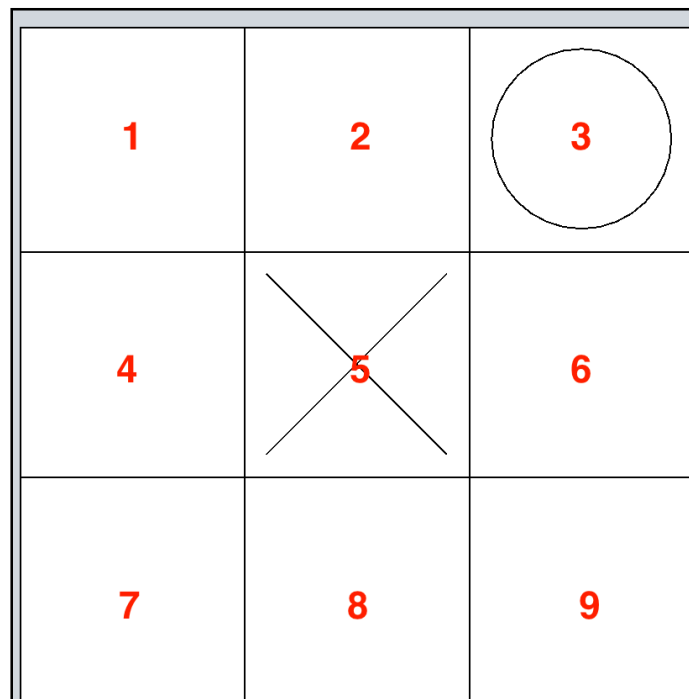
# Exercice X – Tic Tac Toe

Le but de cet exercice est de programmer une version simple du jeu Tic Tac Toe. Le diagramme de classe se trouve en dernière page.

1. Copiez le projet **ExerciceX\_M** dans votre répertoire de classe. Ouvrez le projet dans NetBeans.

## La classe `Input`

La classe `Input` représente une entrée de l'utilisateur, c.-à-d. un X ou un O. Pour cela possède deux attributs. L'attribut `isCross` est un booléen qui indique s'il s'agit d'un X (`true`) ou d'un O (`false`). L'attribut `position` indique dans quelle case de la grille l'entrée se trouve. Les différentes cases de la grille sont numérotées comme à l'image suivante :



2. Ajoutez la méthode suivante :

```
public void draw(Graphics g, int x, int y, int width, int height)
```

Elle permet de dessiner une entrée de l'utilisateur. Pour cela elle a besoin de connaître la position et la dimension de la case qui contient l'entrée. Entre le bord de la case et un X ou un O il y a toujours un espace blanc qui constitue 10% de la longueur totale de la case.

### La classe `GameLogic`

La classe `GameLogic` représente une partie de Tic Tac Toe. Pour cela elle possède deux attributs : une liste `alInputs` qui contient toutes les entrées des utilisateurs et un booléen `nextIsCross` qui indique si l'entrée suivante sera un X (`true`) ou un O (`false`).

3. Ajoutez la méthode `searchInputWithPosition` qui accepte un numéro de case en paramètre et cherche l'entrée correspondante dans la liste.
4. Ajoutez la méthode ...

```
public void draw(Graphics g, int pSize)
```

... qui dessine toutes les entrées de l'utilisateur. `pSize` est la dimension de la grille entière.

---

*Il faut vérifier pour chaque case s'il y a une entrée et seulement s'il y en a une il faut la dessiner.*

---



5. Ajoutez la méthode `newGame` qui réinitialise la liste avec les entrées.
6. Ajoutez la méthode `setInput` qui accepte une position en paramètre et ajoute une entrée dans la case indiquée si celle-ci est encore vide.

---

*Il faut penser à utiliser et à actualiser l'attribut `nextIsCross`.*

---



7. Enlevez les commentaires autour de la méthode `isWon` qui vérifie si un joueur a gagné.
8. Ajoutez la méthode `isFull` qui vérifie si la grille de jeu est pleine.

### La classe `DrawPanel`

9. Ajoutez un attribut `gl` de la classe `GameLogic` et initialisez-le avec la valeur `null`.
10. Ajoutez un manipulateur pour cet attribut.
11. Définissez la méthode `paintComponent` qui dessine un arrière-plan blanc et la grille de jeu elle-même.

### La classe `MainFrame`

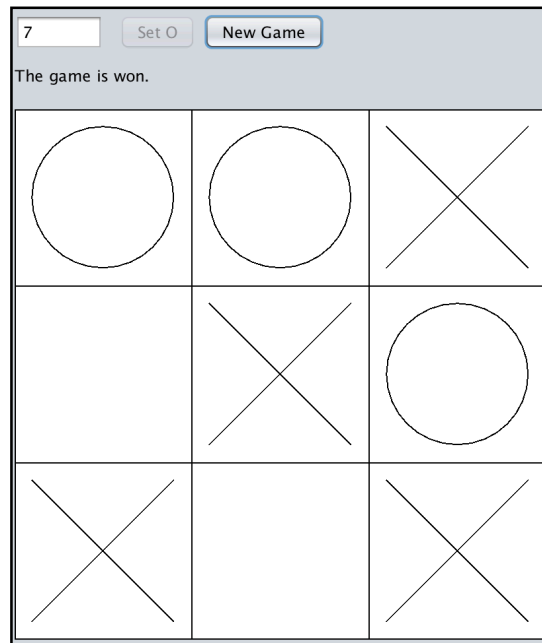
12. Ajoutez un attribut `gl` de la classe `GameLogic` et initialisez-le avec un nouvel objet.
13. Complétez le constructeur comme suit :
  - a. Au début l'étiquette affichant le message est vide.
  - b. Initialisez l'attribut `gl` du panneau.
14. Le bouton **Set X / Set O** utilise la position indiquée dans le champ de texte pour ajouter un X ou un O à la grille. Il faut vérifier que la position est bien entre 1 et 9. Le texte du bouton est actualisé de façon à ce qu'il indique à tout instant si la prochaine entrée sera un X ou un O. Il faut aussi vérifier si le jeu est gagné ou si la grille est pleine après

avoir ajouté une entrée. Dans les deux cas le jeu est terminé et un message approprié est affiché :

- The game is won.** (Un joueur a gagné.)
- The game is a draw.** (La grille est pleine, mais aucun joueur n'a gagné.)

Quand le jeu est terminé il faut désactiver le bouton **Set X / Set O**.

**Exemple – Le joueur avec les X vient d'insérer un X dans la case 7, ce qui termine le jeu.**



15. Le bouton **New Game** permet de commencer une nouvelle partie.

