

---

# TP8 SMP

---

But : Ce TP a pour but de réaliser un jeu de plateau avec des animaux. A chaque tour, les animaux sur le plateau se déplacent. Lorsque des animaux se retrouvent à occuper la même case sur le plateau, ils s'affrontent au Shifumi. Le vainqueur du Shifumi occupe alors la case pour le tour courant.

Le thème du TP est la réalisation du jeu sous forme de classe en c++.

## I. Réalisations des programmes

### 1. Construction des classes relative aux animaux:

#### a. Définition de la classe Animal :

Nous réaliserons tout d'abord une classe virtuelle **Animal** de laquelle découlerons les classes animal concrètes comme **Loup**, **Lion**, **Ours** ou encore... **Pierre**.

La classe Animal est défini comme suit :

```
Animal

# string nom
# int x //abscisse de l'animal
# int y //ordonnée de l'animal
# bool vivant
# Attaque typeAttaque

+ Animal(int maxX, int maxY)
+ Animal(int maxX, int maxY, int a, int b)
+ string getNom() const
+ int getX() const
+ int getY() const
+ bool getVivant() const
+ Attaque getAttaque()const
+ void setVivant(bool v)
+ bool attaque(Animal &a)
+ void setAttaque(Attaque atq) //virtuelle pure
+ void deplace(int maxX, int maxY) //virtuelle pure
```

Cette classe comportera l'ensemble des méthodes et attributs utiles pour définir un anima, parmi elles on retrouve deux méthodes virtuelles (*setAttaque* et *deplace*) qui devront obligatoirement être redéfini par chaque classe d'animaux qui hérite de **Animal**.

Chacune de ces deux méthodes a une implémentation spécifique qui dépend du type d' Animal.

Un animal d'un certain type se déplace d'une certaine manière et attaque d'une certaine manière qui diffère d'un autre type d'animal.

b. Définition de la classe Attaque :

Avec cette classe **Animal**, on réalise une classe **Attaque**. Une instance de cette classe sera utilisée par la classe **Animal** pour définir les différents types d'attaques d'une instance **Animal**.

La classe **Attaque** est défini comme suit :

Attaque
<pre>- int type // 0 :pierre, 1: Feuille, 2:Ciseaux</pre>
<pre>+ Attaque() //crée une attaque random + Attaque(int a) //crée une attaque spécifique + int getTypeAttaque() const + bool resoudreAttaque(Attaque &amp;a) const + string getNomAttaque() const</pre>

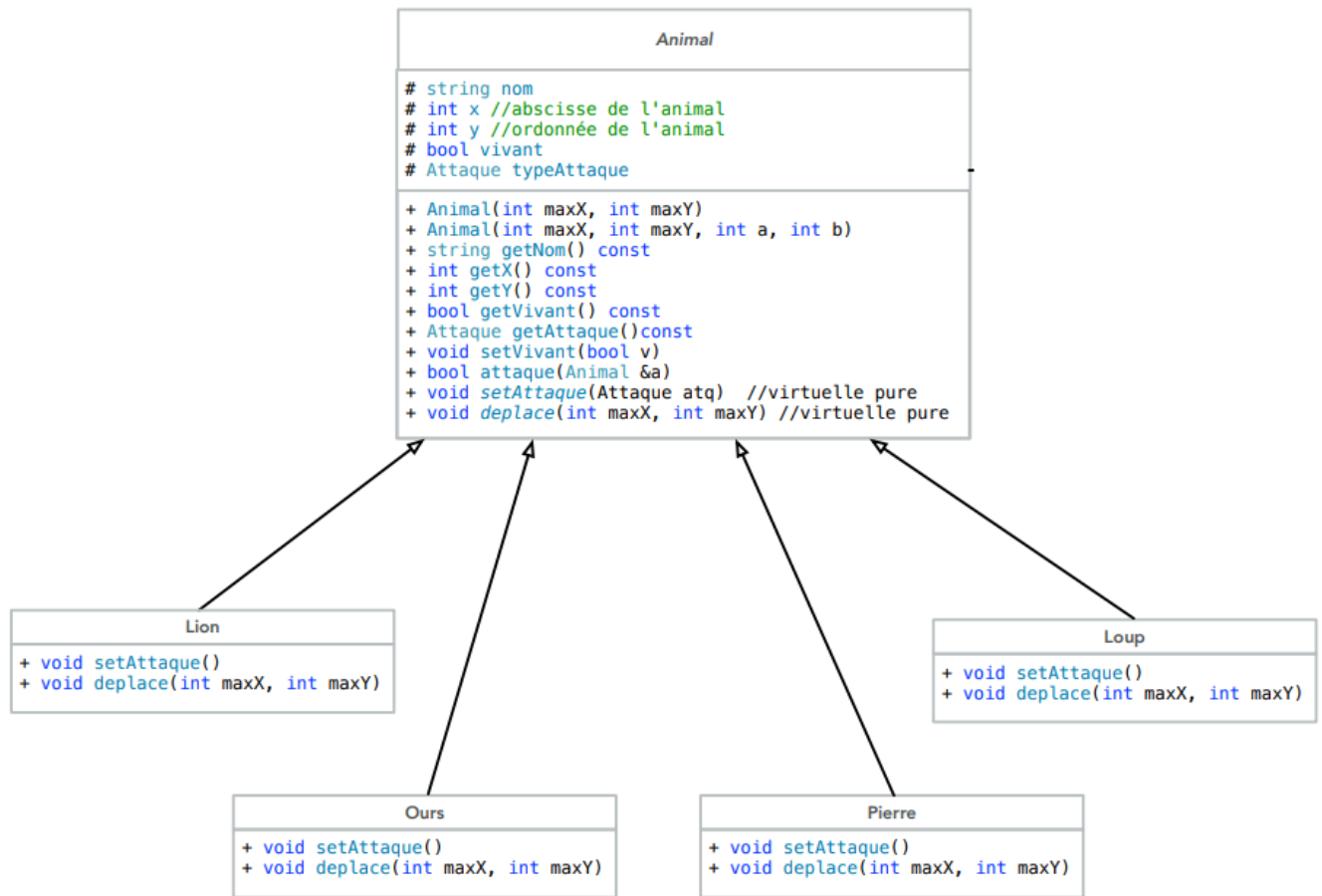
Une instance de type **Attaque** introduit un entier *type* qui prend pour valeur 0, 1, 2 qui correspondent respectivement à Pierre, Feuille et Ciseaux. La classe **Attaque** implémente aussi une méthode *resoudreAttaque* qui vérifie si l'attaque est « plus forte » qu'une autre attaque. Cela nous permettra de gérer les conflits entre les animaux dans le plateau. Si deux attaques sont équivalentes, le choix de celle qui l'emporte est décidé aléatoirement.

c. Définition des classes d'animaux

Nous réalisons maintenant les classes correspondant aux différents types d'animaux en commençant par la classe **Loup**.

Pour chaque classe qui hérite d'**Animal** nous devrons simplement redéfinir les méthodes virtuelles *setAttaque* et *deplace*. Accessoirement, nous définirons le nom du type d'animal à l'aide du constructeur de chaque classe d'animal spécifique.

Ainsi le diagramme de classe des classes d'animaux est le suivant :



#### i. Définition des méthodes virtuelles de la classe Loup

Pour la classe **Loup** :

Nous définissons *setAttaque* en sachant que les loups attaquent de manière aléatoire avec l'une des trois attaques possibles.

Ensuite, nous définissons *deplace* en tenant compte du fait que les loups se déplacent au hasard sur n'importe quelle cellule du plateau.

#### ii. Définition des méthodes virtuelles de la classe Lion

Pour la classe **Lion** :

Nous définissons *setAttaque* en sachant que les lions attaquent de manière aléatoire avec feuille ou ciseaux.

Ensuite, nous définissons *deplace* en tenant compte du fait que les lions se déplacent, au hasard, à chaque tour dans une des 4 directions (1,1), (1,-1), (-1,-1) ou (-1,1) (cf. Figure 1.1).

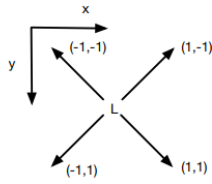


FIGURE 1.1 – Déplacements des Lions.

### iii. Définition des méthodes virtuelles de la classe Ours

Pour la classe **Ours** :

Nous définissons *setAttaque* en sachant que les ours attaquent uniquement avec Feuille.

Ensuite, nous définissons *deplace* en tenant compte du fait que les ours se déplacent au hasard à chaque tour selon une des 8 directions suivantes (2,1), (1,2), (-1,2), (-2,1), (-2,-1), (-1,-2), (1,-2), (2, -1) (cf. Figure 1.2).

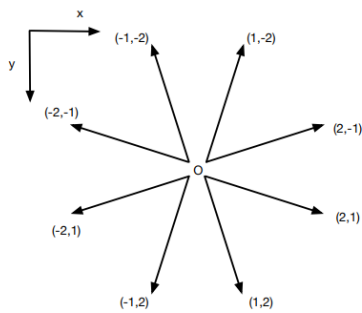


FIGURE 1.2 – Déplacements des Ours.

### iv. Définition des méthodes virtuelles de la classe Pierre

Pour la classe **Pierre** :

Nous définissons *setAttaque* en sachant que les pierres attaquent uniquement avec des pierres.

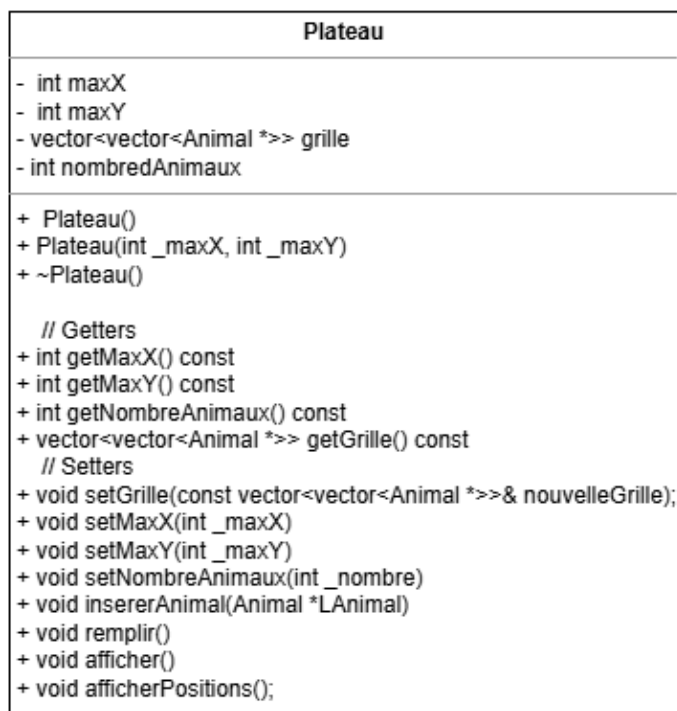
Ensuite, nous définissons *deplace* en tenant compte du fait que les pierres ne se déplacent pas.

## 2. Définition de la classe Plateau

Nous définissons maintenant la classe **Plateau** qui contiendra un tableau bi-dimensionnelle d'animaux, qui représentera la grille, le plateau en lui-même, ainsi que des méthodes permettant d'afficher le plateau, le remplir en début de partie et afficher les positions de chaque animal sur le plateau.

En remplissant le tableau avec la méthode *remplir*, on prendra garde à placer de manière équitable chaque type d'animaux et à remplir le tableau à 25% de sa capacité. Nous réaliserons une méthode *insérerAnimal* qui sera utiliser dans *remplir* et qui permettra d'ajouter les animaux dans le tableau bi-dimensionnelle correspondant à la grille. *InsérerAnimal* vérifiera si la case du plateau est vide avant d'affecter la case à l'animal à insérer. Si la case sur laquelle étant censée être placé l'animal est occupé alors l'animal est déplacé et on vérifie à nouveau que la nouvelle case où devra être placé l'animal est vide. On répète la procédure tant que la case n'est pas vide et tant que toutes les positions du tableau n'ont pas été testées.

Le diagramme de classe de Plateau est alors le suivant :



## 3. Définition de la classe Jeu

### a. Les méthodes de mise en place du jeu

La classe Jeu permettra de lancer le jeu de plateau. Pour ce faire, cette classe commencera tout d'abord par remplir le plateau puis afficher les positions courantes des animaux sur la grille ainsi que le nombre d'animaux. Elle réalisera ensuite le déplacement des animaux sur le plateau, la mise à jour des attaques de chaque animal et la gestion des conflits lorsque deux ou plus animaux se retrouvent sur la même case. Ces

actions seront réalisés à chaque tour(un tour est défini par le déplacement des animaux, la mise à jour des attaques des animaux et, s'il y a collisions c'est-à-dire si des animaux occupent la même case, leur affrontement.

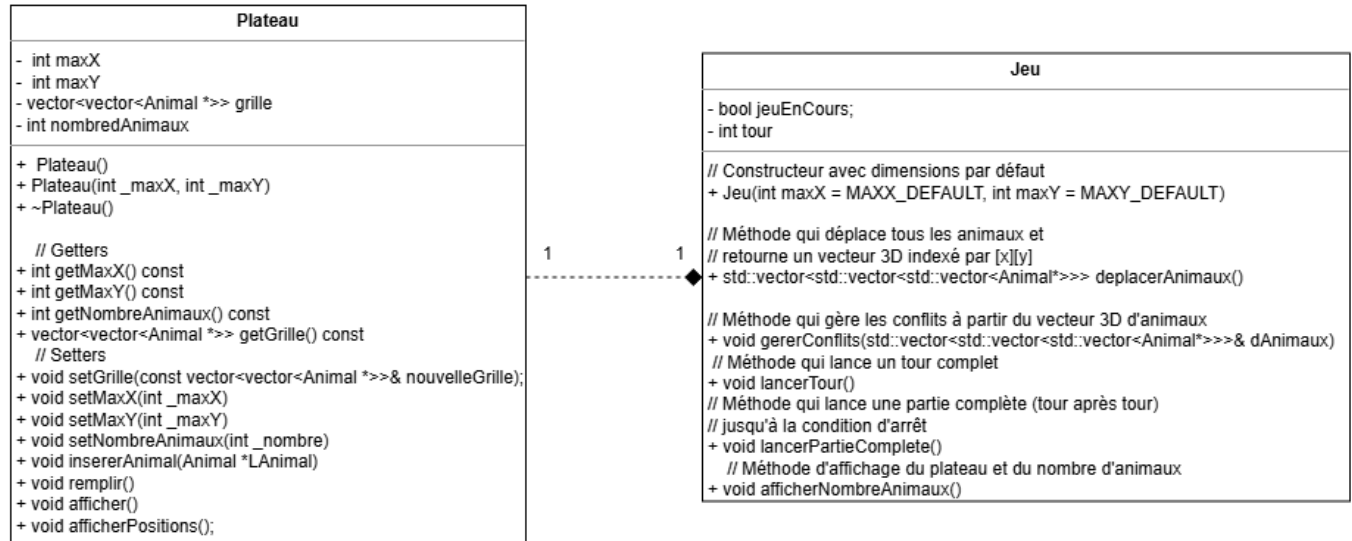
Pour pouvoir plusieurs animaux sur la même case du plateau, nous avons décidé de stocker les animaux dans un tableau tri-dimensionnelle d'animaux après chaque déplacement. Ceci permet d'éviter que lorsqu'un animal vient à se déplacer sur une case déjà occuper de pouvoir éviter d'écraser et remplacer directement l'animal sur la case sans que les deux aient eu à s'affronter.

Une fois les animaux déplacés et stockés dans le tableau 3D, on vient parcourir le tableau 3D et pour chaque case ayant plus d'un seul élément, on vient faire s'affronter entre eux chaque animal sur la case.

Le vainqueur occupe alors la case pour le tour courant. Lorsque toutes les cases du tableau 3D n'occupent plus qu'un seul case alors on peut mettre à jour la grille et l'afficher puis passer au tour suivant.

Finalement, les tours se succèdent jusqu'à ce qu'il ne reste plus qu'un seul animal sur le plateau ou jusqu'à ce que les seuls animaux restants sur le plateau soient des pierres. Cette dernière condition vient compléter la première et évite de bloquer le programme : les pierres ne pouvant pas se déplacer, lorsque qu'il ne reste que ce type d'animaux ces derniers ne pourront jamais se rencontrer et s'affronter s'ils n'occupent pas la même case et donc la première condition ne pourra jamais être validée.

Le diagramme de la classe Jeu est le suivant :



#### b. Le menu

Dans cette classe Jeu, on définit aussi le menu du jeu.

Ce menu propose 3 choix à l'utilisateur :

- 1- Lancer un tour
- 2- Lancer une partie complète
- 3- Quitter la partie

L'option 1 permet de lancer un tour(déplacement des animaux + éventuellement l'affrontement de certains d'entre eux).

Après chaque tour, le menu se réouvre.

L'option 2 permet de lancer une partie jusqu'à ce que la condition de fin ( plus qu'un seul animal sur plateau ou le seul type d'animaux restant est pierre) soit rencontrée.

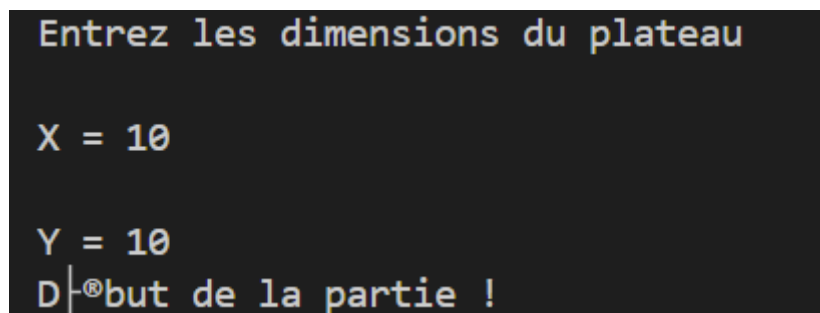
L'option 3 permet de simplement quitter le programme.

#### 4. Programme principale

Le programme principale définira simplement une instance de la classe Jeu afin de pouvoir lancer le jeu.

Il demandera au préalable les dimensions en X et en Y du plateau pour la génération du jeu.

Un fois ces informations renseignées, la partie se lancera :



```
Entrez les dimensions du plateau  
  
X = 10  
  
Y = 10  
D|®but de la partie !
```

Figure 2: Exemple d'entrée utilisateur pour la génération du jeu

## II. Jeux d'essais

### 1. Essai avec un plateau 4 x 4

#### a. Lancement du jeu

Pour tester le fonctionnement de notre programme, nous définissons d'abord un plateau de 4 x 4 (taille minimale du plateau pour avoir à la fois le plateau rempli à 25% et au moins un animal de chaque type sur le plateau).

Nous obtenons bien un plateau rempli à 25% avec un animal de chaque type en début de partie.

```
Entrez les dimensions du plateau

X = 4

Y = 4
D|®but de la partie !
Lion: (1; 0)
Pierre: (1; 1)
Ours: (2; 2)
Loup: (3; 2)
|---|---|---|---|
|   |  Li  |   |   |
|---|---|---|---|
|   |  Pi  |   |   |
|---|---|---|---|
|   |   |  Ou  |  Lo  |
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|

|État initial du plateau : 4 animaux.
```

Nous pouvons déjà constater que les positions affichées de manière textuelle correspondent bien à celle afficher sur la grille. Le nombre d'animaux quant à lui est cohérent.



## b. Le menu

Le menu de sélection apparaît juste après l’affichage de l’état initial du jeu.

```
Options :
1 - Lancer un tour
2 - Lancer une partie complète
3 - Quitter la partie
Choix : 2
```

### i. Choix de l’option 3 :

Nous choisissons d’abord l’option 3 permettant de quitter la partie et vérifions que le programme prend fin.

```
Options :
1 - Lancer un tour
2 - Lancer une partie complète
3 - Quitter la partie
Choix : 3
Fin de la partie.
```

On observe bien la fin de l’exécution du programme marquée par l’affichage du message « Fin de la partie ».

### ii. Choix de l’option 1 :

Nous choisissons désormais l’option 1 permettant de lancer un tour et observons la bon exécution du tour.

```
Début du tour 1...
Lion: (1, 0) -> (0, 3)
Pierre: (1, 1) -> (1, 1)
Ours: (2, 2) -> (0, 3)
Loup: (3, 2) -> (3, 3)
Conflit en (0, 3) : 2 animaux s'affrontent.
Lion vs Ours :
> Lion attaque avec Ciseaux
> Ours attaque avec Feuille
Lion gagne !
|---|---|---|
|   |   |   |
|---|---|---|
|   | Pi |   |
|---|---|---|
|   |   |   |
|---|---|---|
| Li |   | Lo |
|---|---|---|

Après le tour 1 : 3 animaux restants.
```

Le tour 1 se lance alors et le menu de sélection se réouvre.

Nous étudierons en détail ce tour plus tard.

Pour l’instant, nous validons le fonctionnement de l’option 1 et nous choisissons l’option 2 permettant de lancer une partie complète.

### iii. Choix de l'option 2 :

Les tours se succèdent alors jusqu'à ce qu'il ne reste qu'un seul survivant. L'option 2 fonctionne comme prévu.

Nous allons désormais étudier en détail le déroulement de chaque tour.

#### c. Etude du fonctionnement du jeu :

On vérifie le bon fonctionnement du jeu pour différents tour :

Pour le tour 1 :

```
Début du tour 1...
Lion: (1, 0) -> (0, 3)
Pierre: (1, 1) -> (1, 1)
Ours: (2, 2) -> (0, 3)
Loup: (3, 2) -> (3, 3)
Conflit en (0, 3) : 2 animaux s'affrontent.
Lion vs Ours :
> Lion attaque avec Ciseaux
> Ours attaque avec Feuille
Lion gagne !
|----|----|----|----|
|    |    |    |    |
|----|----|----|----|
|    | Pi |    |    |
|----|----|----|----|
|    |    |    |    |
|----|----|----|----|
| Li |    |    | Lo |
|----|----|----|----|

Après le tour 1 : 3 animaux restants.
```

Nous observons les déplacements pour ce tour :

- Le lion réalise un déplacement cohérent avec son comportement. Le lion passe de la case (1 ; 0) à la case (0 ; 3) ce qui signifie qu'il se déplace de -1 en X mais aussi de -1 en Y puisque étant sur le bord en Y (0), il se retrouve à l'opposé du plateau en Y (3). En résumé, son mouvement est **(-1, -1)** en tenant compte de l'effet de bord. Ce qui correspond bien à ce que l'on a défini plutôt sur la classe Lion.

- La pierre ne se déplace pas quant à elle. Ce qui correspond bien à ce qui est attendu.
- Le déplacement de l'ours de (2 ; 2) à (0, 3) suggère qu'il a fait un mouvement (dx ; dy) de (-2 ; 1). Ce qui est bien un des déplacements attendus pour l'ours.

L'ensemble des déplacements réalisés par chaque animal est cohérent avec le comportement défini plus tôt dans chaque classe.

Pour ce tour, nous constatons un conflit de position en (0,3) où se trouvent le lion et l'ours.

Un combat est alors enclenché.

```

Conflit en (0, 3) : 2 animaux s'affrontent.
Lion vs Ours :
> Lion attaque avec Ciseaux
> Ours attaque avec Feuille
Lion gagne !
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|
|   | Pi |   |   |
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|
| Li |   |   | Lo |
|---|---|---|---|

Après le tour 1 : 3 animaux restants.

```

On observe le détail de l'affrontement :

- Le lion attaque avec Ciseaux, ce qui est cohérent puisqu'il est censé pouvoir attaquer de manière aléatoire avec Feuille ou Ciseaux.
- L'ours attaque avec Feuille, ce qui est bien cohérent puisqu'il ne peut attaquer qu'avec Feuille.
- Le lion attaque avec Ciseaux tandis que l'ours attaque Feuille. Selon les règles du Shifumi, l'issue du combat serait une victoire du lion. C'est bien le cas ici.
- La position (0 ; 3) est désormais occupé par le lion, vainqueur de l'affrontement. L'affichage de la grille est bien cohérente ( Lion en (0 ; 3)).

Il ne reste donc pour ce tour que 3 animaux comme indiqué sur le terminal.

Pour le tour 5 :

On se contentera pour ce tour de vérifier que les attaques des animaux sont mis à jour.

```
D|@but du tour 5...
Loup: (0, 1) -> (2, 1)
Pierre: (1, 1) -> (1, 1)
Lion: (3, 0) -> (2, 1)
Conflit en (2, 1) : 2 animaux s'affrontent.
Loup vs Lion :
> Loup attaque avec Ciseaux
> Lion attaque avec Feuille
Loup gagne !
|----|----|----|----|
|    |    |    |    |
|----|----|----|----|
|    | Pi | Lo |    |
|----|----|----|----|
|    |    |    |    |
|----|----|----|----|
|    |    |    |    |
|----|----|----|----|

Après le tour 5 : 2 animaux restants.
```

Nous constatons que le lion a une attaque différente de précédemment. Pour le tour 1, le lion attaquait avec Ciseaux désormais il attaque avec Feuille. Cela implique qu'entre le tour 1 et le tour 5, l'attaque du lion a été mis à jour. Par ailleurs, on constate bien que cette attaque fait partie de ces attaques possibles à savoir Ciseaux et Feuille, pour ce tour il s'agit de Feuille.

Accessoirement pour ce tour, on peut constater que la pierre ne s'est toujours pas déplacé. On vérifiera une nouvelle fois ce comportement pour les 2 tours qui suivent.

Pour les tours 6 et 7 :

Après le tour 6 : 2 animaux restants.

Début du tour 7...

Pierre: (1, 1) -> (1, 1)

Loup: (3, 3) -> (1, 2)

	Pi		
	Lo		

Après le tour 7 : 2 animaux restants.

Début du tour 8...

Pierre: (1, 1) -> (1, 1)

Loup: (1, 2) -> (2, 0)

		Lo	
	Pi		

On constate que le loup se déplace de manière aléatoire sur la plateau et qu'à aucun moment la pierre ne s'est déplacé.

On passe directement à l'étude du dernier tour pour connaître l'issue de la partie.

Pour le tour 20(dernier tour) :

```

Après le tour 20 : 2 animaux restants.
Début du tour 21...
Loup: (1, 0) -> (1, 1)
Pierre: (1, 1) -> (1, 1)
Conflit en (1, 1) : 2 animaux s'affrontent.
Loup vs Pierre :
> Loup attaque avec Pierre
> Pierre attaque avec Pierre
Pierre gagne !
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|
|   | Pi |   |   |
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|
|   |   |   |   |
|---|---|---|---|

Après le tour 21 : 1 animaux restants.
La partie est terminée ! Vainqueur : Pierre

```

On observe qu'au bout de 20 tours la pierre et le loup, qui étaient les derniers animaux restants sur le plateau, finissent par se croiser en (1 ;1) et croisent alors le fer :

- Le loup attaque avec Pierre, ce qui est cohérent puisque le loup peut attaquer avec les 3 attaques possibles.
- La pierre quant à elle attaque avec Pierre, ce qui correspond au comportement attendu.
- Etant donnée que les deux animaux attaquent avec la même attaque. Le sort de l'affrontement est décidé au hasard. Ici c'est la pierre qui remporte le combat et qui finit donc par occuper la position et par gagner la partie puisqu'elle est la dernière survivante.

Finalement, on a vérifié que les déplacements et les affrontements des animaux étaient cohérents. On peut donc valider le bon fonctionnement du jeu avec un plateau de dimensions 4x4.

Désormais, nous allons faire le tester avec un plateau de dimensions 10x10. Ceci nous permettra de valider le fonctionnement du jeu lorsqu'il y a plus d'un animal par type sur le plateau.

## 2. Essai avec un plateau 10 x 10 :

L'utilisateur commence par définir la taille du plateau.

```
Entrez les dimensions du plateau  
  
X = 10  
  
Y = 10  
D|@but de la partie !
```

Une fois les dimensions du plateau définies le plateau est créé et le jeu peut débuter.

```
D|@but de la partie !  
Pierre: (1; 1)  
Loup: (2; 1)  
Pierre: (2; 3)  
Ours: (2; 4)  
Lion: (2; 7)  
Lion: (3; 1)  
Ours: (3; 3)  
Loup: (5; 0)  
Lion: (5; 2)  
Pierre: (5; 5)  
Ours: (6; 1)  
Pierre: (6; 5)  
Loup: (7; 1)  
Ours: (7; 3)  
Loup: (7; 4)  
Ours: (8; 2)  
Lion: (8; 5)  
Lion: (8; 6)  
Loup: (8; 7)  
Pierre: (9; 2)  
Ours: (9; 3)  
Lion: (9; 4)  
Loup: (9; 5)  
Pierre: (9; 8)  


|  |    |    |    |  |    |    |    |    |    |
|--|----|----|----|--|----|----|----|----|----|
|  |    |    |    |  | Lo |    |    |    |    |
|  |    |    |    |  |    |    |    |    |    |
|  | Pi | Lo | Li |  |    | Ou | Lo |    |    |
|  |    |    |    |  | Li |    |    | Ou | Pi |
|  |    |    |    |  |    |    |    |    |    |
|  |    | Pi | Ou |  |    |    | Ou |    | Ou |
|  |    |    |    |  |    |    |    |    |    |
|  |    | Ou |    |  |    |    | Lo |    | Li |
|  |    |    |    |  |    |    |    |    |    |
|  |    |    |    |  | Pi | Pi |    | Li | Lo |
|  |    |    |    |  |    |    |    |    |    |
|  |    |    |    |  |    |    |    | Li |    |
|  |    |    |    |  |    |    |    |    |    |
|  |    | Li |    |  |    |    |    | Lo |    |
|  |    |    |    |  |    |    |    |    |    |
|  |    |    |    |  |    |    |    |    | Pi |
|  |    |    |    |  |    |    |    |    |    |
|  |    |    |    |  |    |    |    |    |    |

  
|état initial du plateau : 24 animaux.
```

On observe bien que le plateau a 10 colonnes et 10 lignes indicées de 0 à 9. De plus, il est rempli à près de 25% comme souhaité. Il y a 24 animaux pour une plateau pouvant contenir jusqu'à 100 animaux ce qui fait une occupation de 24% du plateau. Par ailleurs, on retrouve le même nombre d'animaux pour chaque type. Les coordonnées des animaux sont cohérentes : il n'y en a aucune supérieur à 9 ou inférieur à 0.

On lance désormais différents tours et analysons le comportements du programme :

Pour le tour 1, on obtient l'affichage suivant :

```

Début du tour 1...
Pierre: (1, 1) -> (1, 1)
Loup: (2, 1) -> (5, 1)
Pierre: (2, 3) -> (2, 3)
Ours: (2, 4) -> (4, 3)
Lion: (2, 7) -> (3, 6)
Lion: (3, 1) -> (4, 2)
Ours: (3, 3) -> (5, 4)
Loup: (5, 0) -> (8, 6)
Lion: (5, 2) -> (4, 1)
Pierre: (5, 5) -> (5, 5)
Ours: (6, 1) -> (5, 3)
Pierre: (6, 5) -> (6, 5)
Loup: (7, 1) -> (0, 0)
Ours: (7, 3) -> (6, 1)
Loup: (7, 4) -> (7, 2)
Ours: (8, 2) -> (9, 0)
Lion: (8, 5) -> (7, 6)
Lion: (8, 6) -> (7, 5)
Loup: (8, 7) -> (8, 8)
Pierre: (9, 2) -> (9, 2)
Ours: (9, 3) -> (8, 1)
Lion: (9, 4) -> (0, 3)
Loup: (9, 5) -> (9, 9)
Pierre: (9, 8) -> (9, 8)

```

Lo									Ou
	Pi			Li	Lo	Ou		Ou	
				Li			Lo		Pi
Li		Pi		Ou	Ou				
					Ou				
					Pi	Pi	Li		
							Li	Lo	
								Lo	Pi
									Lo

```

Après le tour 1 : 24 animaux restants.

```

Pour ce tour, les déplacements sont cohérents. Prenons par exemple la pierre précédemment en (1 ; 1) et le lion précédemment en (2 ; 7). La pierre ne bouge pas et occupe la même position au lancement de ce tour. Le lion quant à lui réalise un déplacement (1 ; -1) lui permettant d'arriver en (3 ; 6). Le mouvement réalisé par le lion fait partie des déplacements possibles qu'il peut réaliser donc le comportement du lion est validé.



On passe désormais au tour suivant, le tour 2.

Pour ce tour, plusieurs affrontements ont lieu. Il y a eu exactement 4 combats. Nous analysons l'un d'entre eux afin de valider la fonctionnalité d'affrontements.

```
Début du tour 2...
Loup: (0, 0) -> (0, 9)
Lion: (0, 3) -> (9, 4)
Pierre: (1, 1) -> (1, 1)
Pierre: (2, 3) -> (2, 3)
Lion: (3, 6) -> (2, 5)
Lion: (4, 1) -> (5, 2)
Lion: (4, 2) -> (5, 1)
Ours: (4, 3) -> (3, 1)
Loup: (5, 1) -> (6, 0)
Ours: (5, 3) -> (7, 2)
Ours: (5, 4) -> (7, 3)
Pierre: (5, 5) -> (5, 5)
Ours: (6, 1) -> (7, 9)
Pierre: (6, 5) -> (6, 5)
Loup: (7, 2) -> (5, 5)
Lion: (7, 5) -> (6, 6)
Lion: (7, 6) -> (6, 5)
Ours: (8, 1) -> (6, 2)
Loup: (8, 6) -> (2, 9)
Loup: (8, 8) -> (2, 9)
Ours: (9, 0) -> (8, 2)
Pierre: (9, 2) -> (9, 2)
Pierre: (9, 8) -> (9, 8)
Loup: (9, 9) -> (6, 2)
Conflit en (2, 9) : 2 animaux s'affrontent.
Loup vs Loup :
> Loup attaque avec Pierre
> Loup attaque avec Feuille
Loup gagne !
Conflit en (5, 5) : 2 animaux s'affrontent.
Pierre vs Loup :
> Pierre attaque avec Pierre
> Loup attaque avec Pierre
Pierre gagne !
Conflit en (6, 2) : 2 animaux s'affrontent.
Ours vs Loup :
> Ours attaque avec Feuille
> Loup attaque avec Ciseaux
Loup gagne !
Conflit en (6, 5) : 2 animaux s'affrontent.
Pierre vs Lion :
> Pierre attaque avec Pierre
> Lion attaque avec Feuille
Lion gagne !
```

Observons le combat en (6 ; 2) entre l'ours et le loup :

- L'ours attaque avec Feuille comme prévu.
- Le loup attaque avec Ciseaux. Ce qui est cohérent puisqu'il peut attaquer avec les attaques possibles.
- Comme attendu, le loup gagne puisque son attaque (Ciseaux) est plus forte que celle de l'ours (Feuille).

L'issue du combat en (6 ; 2) est cohérente. En observant rapidement les autres combats, on peut affirmer que leurs issues sont logiques et donc on peut valider la fonctionnalité d'affrontements dans le jeu.

						Lo			
	Pi		Ou		Li				
					Li	Lo	Ou	Ou	Pi
		Pi					Ou		
									Li
		Li			Pi	Li			
					Li				
									Pi
Lo		Lo					Ou		

Après le tour 2 : 20 animaux restants.

Nous observons la grille et constatons que seul les vainqueurs des affrontements occupent les cases où avaient lieu les conflits. Par ailleurs, on observe que le nombre d'animaux restants est cohérent. En effet, il y a eu 4 combats ce qui signifie que 4 animaux sont amenés à disparaître du plateau. Ce qui est bien le cas car on passe de 24 à 20 animaux après le tour 2.

Pour la suite de ce test, nous passons directement à la vérification de l'issue de la partie.

La partie s'arrête au bout de 100 tours, lorsque les deux derniers animaux, à savoir le loup et le lion, se croisent en (5 ; 4) et s'affrontent :

```

Après le tour 100 : 2 animaux restants.
Début du tour 101...
Lion: (4, 5) -> (5, 4)
Loup: (6, 3) -> (5, 4)
Conflit en (5, 4) : 2 animaux s'affrontent.
Lion vs Loup :
> Lion attaque avec Ciseaux
> Loup attaque avec Pierre
Loup gagne !
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|Lo|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|
|---|---|---|---|---|---|---|---|---|

Après le tour 101 : 1 animaux restants.
La partie est terminée ! Vainqueur : Loup

```

Le lion attaque avec l'une de ces deux attaques possibles (Feuille ou Ciseaux). Ici, il s'agit de Ciseaux.

Il perd face au Loup qui attaque avec Pierre. Le loup occupe alors la position (5 ; 4) comme indiqué ci-dessus.

Avec ces 2 jeux d'essais, nous pouvons affirmer le bon fonctionnement du jeu. Les fonctionnalités de déplacements, de gestions de conflits, d'affichages des positions, des déplacements et de la grille sont validées.