Développement orienté objet

TD7 : Jeu de l'oie

Le jeu

Ce jeu se présente sous la forme d'un plateau constitué de cases numérotées à partir de 1. Ci-dessous un plateau traditionnel de 63 cases.

Le plateau est constitué de différents types de case qui ont des conséquences différentes sur le jeu lorsqu'un joueur les atteint. Pour gagner et finir la partie, il faut qu'un joueur arrive exactement sur la dernière case.

Les différents joueurs, en nombre quelconque, jouent chacun leur tour. Selon la case où il se trouve un joueur a ou non le droit de quitter cette case. Si ce n'est pas le cas, il passe son tour. S'il est autorisé à joueur, il lance 2 dés à 6 faces et avance son pion d'un nombre égal à la somme des 2 dés pour arriver sur sa case destination. En fonction de cette case de destination, il est possible que le pion du joueur *rebondisse* et revienne en arrière sur une autre case qui devient sa position d'arrivée finale.

Si son lancer de dés lui fait *dépasser* la dernière case, il revient en arrière du nombre de cases en excès. Par exemple, si un jouer est sur la case 57 et fait 9 aux dés, il avance jusqu'à la case 63 en dépensant 6 points et recule ensuite des 3 points restants pour se retrouver sur la case 60.

L'arrivée sur une case a deux conséquences :

1. Elle déclenche un *rebond* :

- si la case est une *case oie*, le joueur *rebondit* en avançant encore son pion d'autant de cases qu'indiqué par les dés (au total, le pion du joueur aura donc avancé de deux fois la valeur des dés) ;
- si la case est une *case de téléportation*, le pion *rebondit* jusqu'à une autre case prédéfinie du plateau qui dépend de la case de téléportation atteinte. La taille du rebond correspond donc à la différence des indices des deux cases ;
- pour toutes les autres cases, il ne se produit rien de particulier : le joueur reste sur la case atteinte, ce qui convient à considérer qu'il *rebondit* sur place ou qu'il fait un *rebond de taille 0*.

2. si la case d'arrivée (après rebond) est déjà occupée, le pion du joueur qui arrive remplace le pion déjà présent et ce dernier est placé sur la case précédemment occupée par le jouer (au moment de son lancé de dés).

Indication : si un rebond fait arriver à une case qui devrait aussi déclencher un rebond, ce dernier n'est pas effectué.

De plus certaines cases ne peuvent être quittées que sous certaines conditions :

- la case est une *case piège*, le joueur ne pourra plus lancer les dés tant qu'il restera sur cette case (c'est-à-dire jusqu'à ce qu'un autre joueur arrive sur cette case et le remplace) ;

- si la case est une *case attente*, le joueur est temporairement empêché de jouer : il ne pourra pas quitter la case pendant un nombre de tours dont la valeur est prédéfinie pour chaque case. Il pourra échapper à cette attente uniquement si un autre joueur arrive sur cette case et le remplace.

Dans le jeu traditionnel, l'emplacement des différents types de cases est prédéfinie. Mais nous souhaitons ici pouvoir définir un plateau de jeu personnalisé.

Mise en oeuvre

Un exemple de partie est donnée ci-dessous :

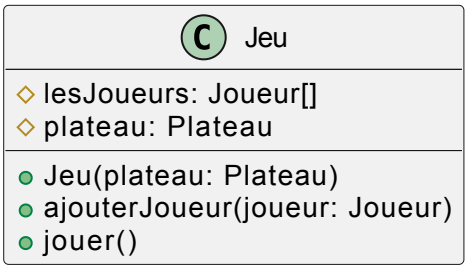
```
Paul est à la case 0, Paul fait 9 et atteint la case 9 (oie) et rebondit à la case 18
(oie)
John est à la case 0, John fait 6 et attein la case 6 (téléportation à la case 12) et
rebondit à la case 12
George est à la case 0, George fait 9 et atteint la case 9 (oie) et rebondit à la case
18 (oie) la case est occupée, Paul est renvoyé à la case 0
Paul est à la case 0, Paul fait 11 et atteint la case 11
John est à la case 12, John fait 11 et atteint la case 23
George est à la case 18, George fait 7 et atteint la case 25
Paul est à la case 11, John fait 8 et atteint la case 18 (attendre 2 tours)
John est à la case 23, John fait 6 et atteint la case 29
George est à la case 25, George fait 7 et atteint la case 32
Paul est à la case 19, Paul ne peut pas jouer
John est à la case 29, John fait 10 et atteint la case 39
...
Paul est à la case 25, Paul fait 6 et atteint la case 31 (piège)
...
George à la case 55, George fait 11 et atteint la case 60 la case est occupée, John est
renvoyé à la case 55
```

```
...
George est à la case 60, George fait 8 et atteint la case 58 (teleportation à la case
1) et rebondit à la case 1
...
John est la case 59, John fait 7 et atteint la case 60
...
Paul est à la case 31, Paul ne peut pas jouer
John est à la case 56, John fait 7 et atteint la case 63
John a gagné
```

Une analyse préalable du problème a identifié les classes suivantes (il sera nécessaire de les compléter) :

La classe `Jeu`

Elle est caractérisée par un plateau et une liste de joueurs et permet de jouer une partie du jeu de l'oie. Voici son diagramme de classe UML :



- la méthode `jouer` déroule une partie jusqu'à son terme quand il y en a un (une partie peut être infinie si tous les joueurs sont dans des cases pièges, on ne testera pas cette situation). Jouer une partie consiste à faire jouer successivement chaque jouer selon les règles décrites ci-dessus.
- La méthode `ajoutJoueur` permet d'ajouter un joueur au jeu. On peut considérer que le jouer est placé sur une case 0, qui est créée spécialement pour lui.

La classe `Joueur`

Un joueur est caractérisé par son nom et la case qu'il occupe. Le code de la classe Joueur est donné ci-dessous :

```
class Jouer {
    protected _case: Case | null;
    protected _nom: string;

    constructor(nom: string) {
        this._nom = nom;
        this._case = null;
    }

    toString(): string {
        return this._nom;
    }
```

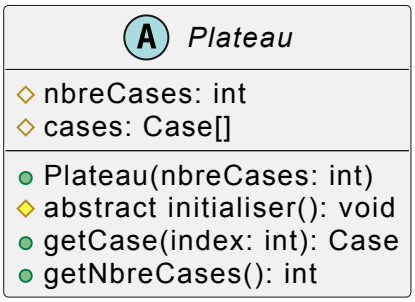
```
    get case(): Case | null {
        return this._case;
    }

    set case(c: Case) {
        this._case = c;
    }

    lancerDes(): number {
        return randInt(1, 6) + randInt(1, 6);
    }
}
```

La classe `Case`

La classe case définie les cases du plateau de jeu.

La classe `Plateau`

La classe `Plateau` est une **classe abstraite**. Elle permet de représenter le plateau de jeu contenant les différentes cases. Le diagramme de classe UML de `Plateau` est donné ci-dessous :



Un plateau est constitué d'un nombre de cases `nbreCases`. Chaque case numéro *i* est rangé dans la case d'indice *i* du tableau `cases`. Le constructeur fait appel à la méthode abstraite `initialiser`. C'est dans cette méthode que sont créées les différentes cases du plateau.

Les sous-classes de `Plateau` peuvent donc personnaliser la configuration du plateau. Vous créerez notamment deux sous-classes :

- `PlateauClassique` qui initialise un plateau de 63 cases avec les emplacements traditionnel des cases oie (9, 18, 27, 36, 45, 54), de téléportation (6 vers 12, 42 vers 30 et 58 vers 1), de piège (31, 52), et d'attente (19 pour 2 tours) aux emplacements traditionnels ;
- `PlateauAléatoire` qui initialise un plateau dont le nombre de cases est choisis au moment de la création, ainsi que le nombre de cases oie et de cases pièges. Ces cases sont placées aléatoirement sur le plateau. Il n'y a pas de cases de téléportation ni d'attente.

Travail à faire

Question 1

Définissez l'algorithme de la méthode `jouer` de la classe `Jeu`. (on ne cherchera pas à détecter les cas où la partie est infinie).

Question 2

Implémentez le jeu de l'oie en TypeScript.

Question 3

Ecrire un programme qui demande si on souhaite jouer à la version classique ou aléatoire du jeu de l'oie et qui lance une partie. Si on choisit une version aléatoire, on demande le nombre de cases du plateau, le nombre de cases oie et le nombre de cases pièges.

Question 4

Donnez le diagramme de classes UML le plus précis de votre jeu de l'oie. Votre diagramme doit doit notamment faire apparaître les associations avec leur type et leur cardinalité.