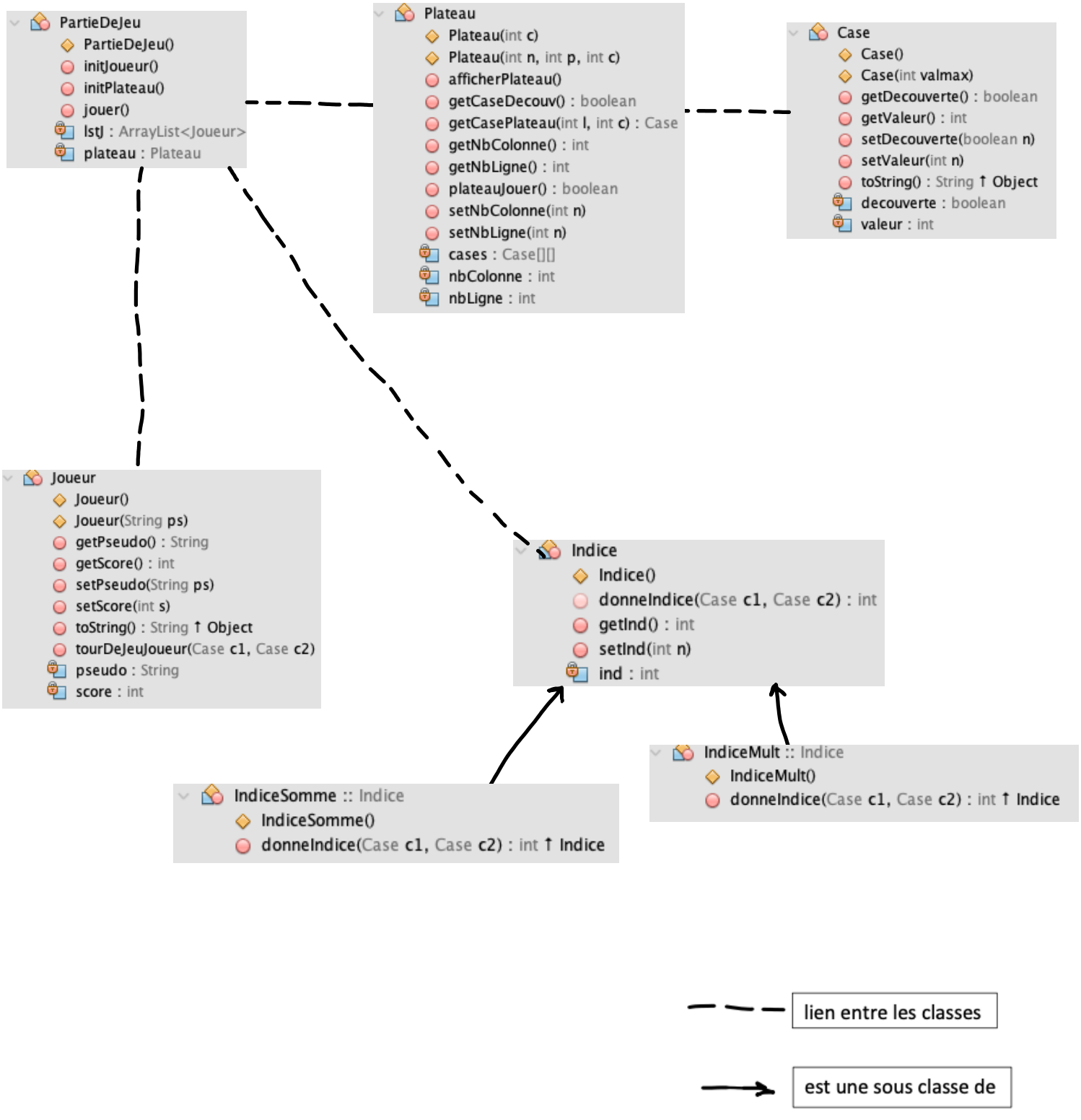
Rapport Projet Info2A

Diagramme de classes



Descriptif des classes

La classe PartieDeJeu( ) :

Cette classe regroupe tous les éléments nécessaires à une partie de jeu.
Elle contient deux attributs :

- ⇒ un attribut lstJ qui est une ArrayList de joueur ;
- ⇒ un attribut plateau de type Plateau ;

Le constructeur de cette classe est composé de :

- ⇒ la déclaration de l'attribut lstJ qui stockera la liste de joueurs pendant le jeu ;
- ⇒ l'appel de la méthode initJoueurs() qui sert à remplir notre attribut lstJ de joueurs (pour que le programme puisse fonctionner normalement dans sa totalité, seulement un joueur peut être ajouté);
- ⇒ l'appel de la méthode initPlateau() qui sert à initialiser notre plateau, le joueur peut alors décider de jouer avec une taille de plateau prédéfinie ou bien, il peut lui-même choisir la taille de son plateau. Il y a également un choix sur le niveau de jeu.
- ⇒ l'appel de la méthode jouer() qui lance le jeu, elle demande au joueur quelles cases il choisit, s'il veut faire des propositions (appel de la méthode tourDeJeuJoueur())...
    - Toutes ces méthodes sont écrites dans la classe LeJeuLoto.
    - Ces méthodes se trouvent dans le constructeur.

Cette classe utilise différents types d'objets. Pour cela, il faut créer d'autres classes qui se chargeront des méthodes associées et créer différents objets qui caractérisent une partie de jeu.

La classe Plateau( ) :

Cette classe permet de créer un plateau de jeu.
On caractérise un plateau de jeu par : le nombre de lignes du plateau, son nombre de colonnes, et un tableau composé d'objets de type Case.

- ⇒ Les attributs nbLigne et nbColonne sont des entiers qui servent à fixer les dimensions du plateau ;
- ⇒ un attribut *cases* qui est un tableau à deux dimensions composé d'objets de type *Case*.

On crée un constructeur par défaut avec en paramètre un entier c qui est la valeur maximale qu'une case peut prendre :

- ⇒ le nombre de lignes et de colonnes est fixé (3 et 4), puis on décrit le tableau où à chaque emplacement on associe une *Case* avec le paramètre c.

On crée également un constructeur standard avec en paramètres le nombre de lignes et de colonnes (de type entier) de notre plateau puis un entier c pour la valeur max de *Case*, pour que le joueur puisse choisir la dimension du plateau de jeu :

- ⇒ le nombre de lignes et de colonnes est fixé avec les paramètres associés, puis on décrit le tableau où pour chaque case, on associe une *Case* avec le paramètre c.

Dans le jeu, on doit pouvoir, avec notre plateau :

⇒ choisir une case spécifique de notre plateau ;
- avec la méthode getCasePlateau(l,c), qui prend en paramètres deux entiers l et c qui sont les numéros de ligne et de colonne de la case, et renvoie cette case.

⇒ savoir si toutes les cases du plateau sont découvertes ;
- une méthode getCaseDecouve() est écrite, elle renvoie un boolean : True si toutes les cases sont découvertes, sinon false.

⇒ l'afficher ;
- rendu possible avec la méthode afficherPlateau(), qui est un affichage standard pour un tableau 2X2.

⇒ savoir si le nombre de cases est pair
- La méthode plateauJouer() compte le nombre de cases du plateau, une variable compt de type entier est alors initialisée à 0 et si ce nombre est pair alors la méthode renvoie true sinon false.

La classe Indice( ) :

La classe *Indice* est une classe abstraite qui définit ce qu'est un indice dans sa globalité.
On la caractérise par un nombre entier.

⇒ Il y a donc en attribut un entier *ind* de type int ;

Cette classe dispose d'un constructeur par défaut :

⇒ on initialise *ind* à 0 ;

Elle aura une méthode abstraite donneIndice(Case c1, Case c2).

⇒ Imposée à l'ensemble de ses sous-classes, elle renvoie un nombre entier en fonction de la valeur des cases c1 et c2;

Elle admettra des sous-classes qui représenteront un indice particulier par exemple une somme ou une multiplication.

La classe IndiceSomme( ) :

La classe *IndiceSomme* est une sous-classe de la classe abstraite *Indice*.
Dans son constructeur on appelle le constructeur de la classe *Indice*
On a une méthode donneIndice() rendue obligatoire par la classe *Indice*.

⇒ Cette méthode renvoie la somme de la valeur de ces deux cases.

La classe IndiceMult( ) :

De même, la classe IndiceMult est une sous-classe de la classe abstraite Indice.
Dans son constructeur on appelle le constructeur de la classe Indice
On a une méthode donneIndice() rendue obligatoire par la classe Indice.

⇒ Cette méthode renvoie cette fois-ci le produit de la valeur de ces deux cases.

La classe Case( ) :

Cette classe permet de représenter une case de notre plateau de jeu.
On caractérise la classe *Case* par un entier (qui est la valeur à découvrir pendant la partie de jeu) et un boolean qui indique si le joueur a découvert cette case.

Deux constructeurs sont créés, un par défaut où la valeur de la case est donnée aléatoirement entre 0 et 9….

- ⇒ Il y a un attribut *valeur* de type int, qui, dans le constructeur, est initialisé avec Math.random() pour générer un nombre aléatoire entre 0 et 9.
- ⇒ un attribut de type boolean initialisé à false, car la case n'est pas découverte au début du jeu.

…et un standard où le joueur, s'il le souhaite, peut choisir la valeur maximum des cases. La valeur à trouver sera donc entre 0 et cette valeur maximale de type int passée en paramètre :

- ⇒ Il y a un attribut valeur de type int, qui ,dans le constructeur, est initialisé avec Math.random() pour générer un nombre aléatoire entre 0 et le paramètre.
- ⇒ un attribut de type boolean initialisé à false, car la case n'est pas découverte au début du jeu.

Une méthode toString() pour afficher l'objet (découvert ou non).

La classe Joueur( ) :

Cette classe permet de représenter un joueur de la partie.
Un joueur est caractérisé par une chaîne de caractères qui est un pseudo et un nombre entier, son score.

- ⇒ Il y a donc un attribut *pseudo* de type string et un attribut *score* de type int.

Deux constructeurs, dont un par défaut…

- ⇒ où pseudo est initialisé à une chaine de caractère vide et le score est initialisé à 0

…et un standard avec un paramètre de type string, pour que le joueur ait un pseudo choisi par l'utilisateur :

- ⇒ où pseudo est initialisé avec le paramètre et le score est initialisé à 0

Un joueur réalise une action qui lui est propre, qui consiste à vouloir émettre une hypothèse sur la valeur des deux cases choisies :

- ⇒ Une méthode tourDeJeuJoueur() est écrite pour demander si le joueur veut faire une supposition sur les valeurs des deux cases.
- ⇒ S'il trouve les bonnes valeurs alors la somme de ces deux cases est ajoutée à son score et les cases sont découvertes

Une méthode toString() est écrite pour afficher l'objet de type joueur.

Le projet sera considéré comme réussi si l'ensemble des conditions décrites ci-dessus sont respectées.