APO2

Projet : Jeu de plateau

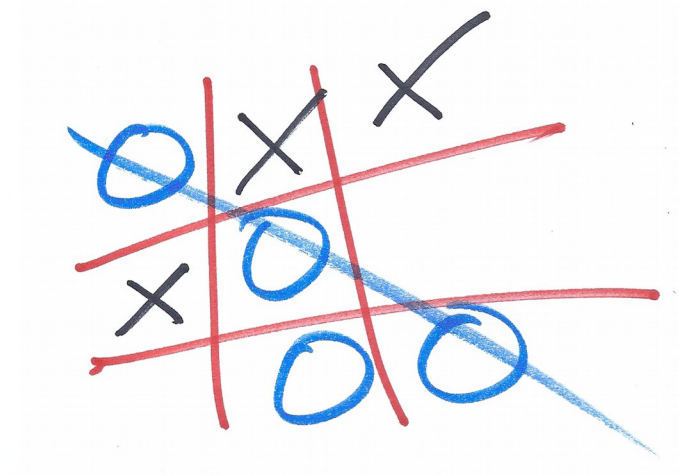
Salima Hassas, Laëtitia Matignon, Antoine Gréa

☘ Compétences à acquérir

- ✔ **Savoir réaliser l'architecture d'un projet** en Utilisant UML
- ✔ **Savoir combiner et hériter les classes Java** dans le cadre de l'implémentation du début du projet.
- ✔ Apprendre à faire du code simple et générique

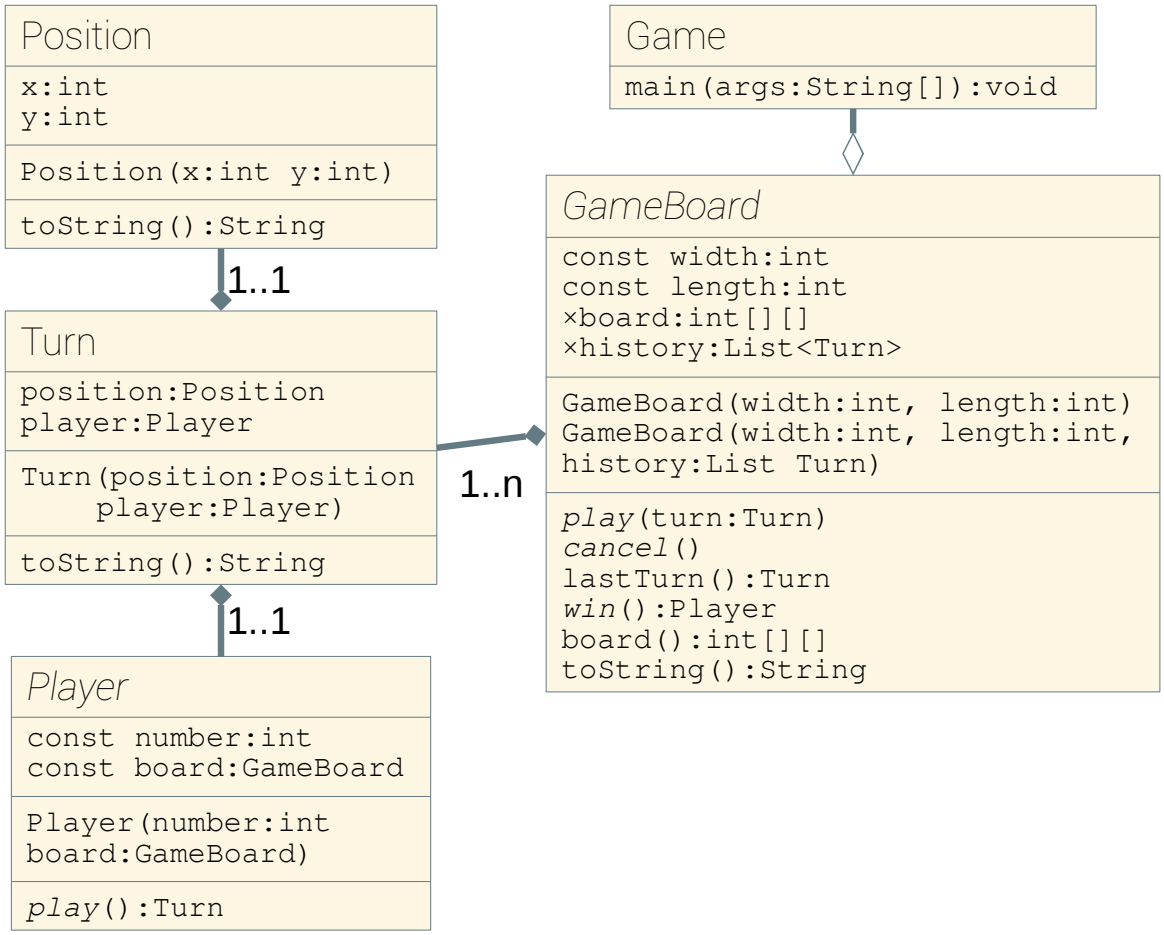
☑ Évaluation

- ✔ Code compilable, organisé, lisible et compréhensible
- ✔ Présentation complète du projet
  - Difficultés rencontrées
  - Spécificité du projet
  - Exemple d'exécution
- ✔ Séance de présentation le 9 janvier 2015



Exercice 1 Base du projet

Le projet de base est un système de jeu de plateau. Le plateau est générique et contient un tableau de case qui sont représenté par des entiers. Le jeu enregistre un historique des coups des joueurs. Voici un diagramme de classe simplifié :



Question 1 Implémentation simple

Créez une implémentation complète de ce schéma.

Pourquoi ne pouvez-vous pas lancer un jeu à ce stade ?

Question 2 Affichage

Implémentez le nécessaire pour afficher tous les éléments du jeu (modification du schéma possible)

Question 3 Sauvegarde

Faite que la classe `Turn` et `GameBoard` implémente l'interface *`Serializable`* et mettez en place le nécessaire pour la sauvegarde de l'historique des coups.

Question 4 Joueur humain

Créez une classe `Human` qui hérite de *`Player`* ainsi que la fonction `Position.parse` qui permet d'entrer une position depuis une chaîne de caractères.

Question 5 Joueur aléatoire

Créez une classe `Random` qui hérite de *`Player`* qui jouera des coups aléatoirement.

Exercice 2 Puissance 4

Question 1 Base

Implémentez la classe `ConnectFour` en héritant `GameBoard`. Créer une énumération `Cell` qui permet de représenter une cellule de la grille. Améliorez cette énumération pour pouvoir afficher les pions rouges et jaunes.

Une **`enum`** est une classe spéciale de java. Elle est **`final`** et contient un nombre fixe d'instance ainsi que quelques fonctions très utiles comme `MY_ENUM_ELEMENT.ordinal()`. On peut ajouter des constructeurs et aussi des fonctions et attributs.

```
  public enum MyEnum {
1   ELEM1("one"), ELEM2("dos"), ELEM3("san");
2   public String content;
3   public static void spoon() { System.out.println("spoon"); }
4   private MyEnum(String content) { this.content = content; }
5 }
```

Pour l'implémentation du jeu :

1. On rappelle que le jeu de puissance 4 se joue sur une grille 7×6.
2. On utilisera si possible un `StringBuilder` ou `String.format()` pour le `toString()` du plateau
3. Voici quelques caractères qui peuvent vous être utiles pour vos pièces : ●○◌

Question 2 Règles

Implémentez les règles du jeu en modifiant `play` et `win`. Ajoutez la gravité et ne permettez de jouer les pions seulement sur la première ligne. Héritez de la classe `Humain` et `Random` afin de prendre en compte les mécaniques du jeu.

Ajoutez de la sécurité sur la gestion des coups et faites une vraie gestion des exceptions pour les coups non valides.

Question 3 Jouez !

Créez une boucle de jeu dans la fonction `play` dans la classe `Game` avec la signature suivante :

```
public static void play( GameBoard board, Player[] palyers)
```

Celle-ci doit afficher le jeu et fait jouer chaque joueur chacun son tour.

Lorsqu'un coup ne peu pas être joué faite rejouer le joueur courant jusqu'à ce qu'a ce que son tour soit joué. Si votre gestion est correcte vous pourrez vous passer des classes de la question précédente.

Question 4 Chuck Noris

Chuck Noris peut finir un puissance 4 en seulement 3 coups !

Implémentez un joueur `ChuckNoris` qui peut gagner le jeu en trois coup et sans modifier l'architecture de base.

Question Bonus Intelligence Artificielle

Réalisez une intelligence artificielle. Notez que ce jeu a été résolu et qu'en jouant en premier et de manière parfaite il est impossible de perdre. Basez-vous sur l'architecture présentée dans la suite.

Exercice 3 Morpion

Question 1 Base

Réalisez une classe `TicTacToe` qui permet de jouer au jeu du morpion classique.

> *i* On rappelle que le jeu de morpion se joue sur une grille 3×3.
> Voici quelques caractères qui peuvent vous être utiles pour vos pièces : ×○

Question 2 Stratégies

1 Stupide

Réaliser la classe `Stupid` qui joue aléatoirement dans une des cases adjacente au dernier coup de l'adversaire.

2 Malin

Reprenez votre code de victoire pour connaître les case qui seront gagnante au prochain tour. Créer une classe `Smart` qui hérite de `Stupid` et qui joue sur ces cases en priorisant sa propre victoire. Dans les autre cas reprenez le comportement de la classe mère.

Question 3 Menu de sélection

Dans la classe `Game`, ajoutez un menu de sélection de Jeu qui permet de jouer à l'un ou l'autre des jeux.

Exercice Bonus Teeko

Vous devrez implémenter le jeu du Teeko dans le programme. Pensez à toutes les subtilités apportée par ce jeu et trouver un moyen simple de coder le cas de victoire. Vous pouvez implémenter une intelligence artificielle plus ou moins avancée.