IUT Sénart-Fontainebleau, UPEC — Florent Madelaine
CPOO
IUT Info 2A — 2025-26

Séquence 2 (FA)

---

1 Introduction

Programme

CM: (vendredi 6/02) rappels de première année sur les classes abstraites, les interfaces. Illustration par des exemples tirés de java.util. Évocation des API.

TD: (vendredi 6/02) Mise en pratique.

Concepts à connaître • classes abstraite • interfaces • couplage faible

2 Les exercices

Exercice 1 (Abstraction)**.** *Disclaimer : Aucune information préalable sur le rugby n'est nécessaire pour réussir cet exercice.*
Afin de préparer les meilleures tactiques et gagner le prochain tournoi, le séléctionneur de l'équipe de france souhaite modéliser les capacités des joueurs de rugby.

Dans cet exercice, le type des attributs et des méthodes n'est pas très important. Il faut se concentrer sur les classes abstraites et les interfaces, et aussi évidemment quelques classes concrètes qui vont en hériter ou les réaliser.

Question 1 : Les joueurs Les joueurs possèdent un nom, un numéro, une nationalité et un club. Ils possèdent également **toujours** une position. On considérera dans cet exercice 3 types de positions : les *avants*, la *charnière* et les *arrières*. Chaque type de position représente un profil de joueur et les actions d'un joueur dépendent de sa position. De même, le numéro d'un joueur dépend de sa position :

- Un avant a un numéro entre 1 et 8
- Une charnière a un numéro entre 9 et 10
- Un arrière a un numéro entre 11 et 15.

Réalisez un *diagramme de classe* permettant de modéliser ces informations.

Question 2 : Les actions On n'attend pas les mêmes capacités d'un joueur selon sa position.

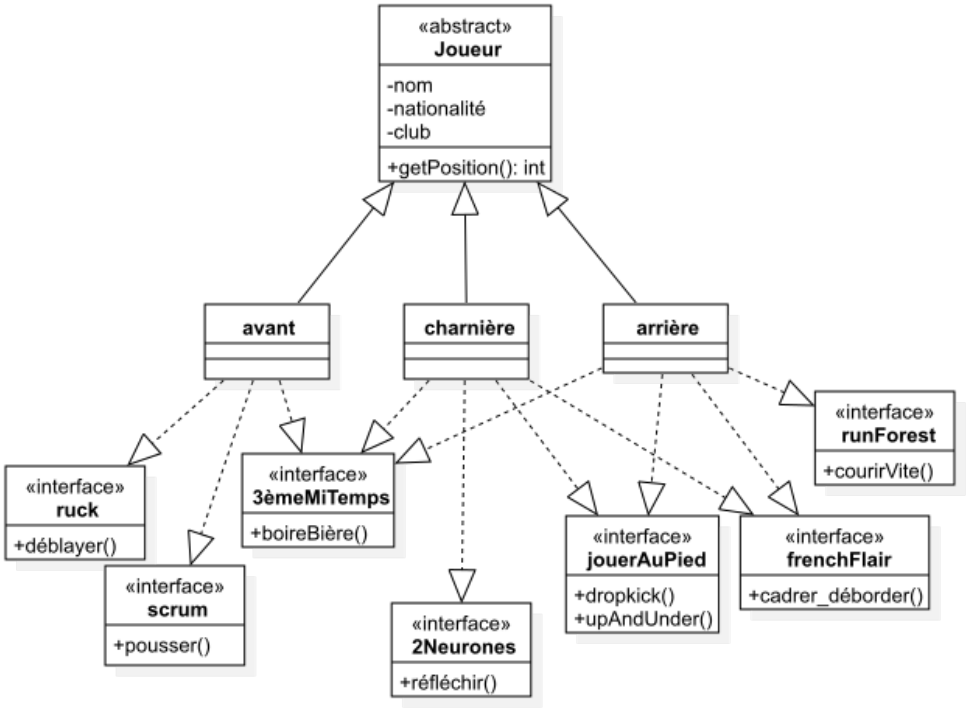
- Un avant doit pouvoir *déblayer* et *pousser dans la mélée*.
- Un arrière peut *faire un cadrage débordement*, *courir vite* et *jouer au pied*, jouer au pied étant soit *tenter un drop* soit *taper une chandelle*.
- Un charnière sait *réfléchir*, *faire un cadrage débordement* et *jouer au pied*.
- Enfin, tous les joueurs peuvent essayer de *passer* et *boire une bière*.

Enrichissez votre diagramme avec les actions des joueurs. On fera attention à **mettre en avant** le fait que certaines positions ont des capacités en commun.

Réponse. Question 1 : Les joueurs Pour les joueurs de rugby, on part sur une classe abstraite dont hérite les trois positions possibles qui devront implémenter les méthodes de Joueurs. En particulier, on a indiqué `getPosition()` qui devra renvoyer une valeur appropriée pour chaque type de joueur.

Une implémentation possible pourrait consister à avoir un attribut position dans Joueur et un getter spécifié dans joueur. C'est dans les constructeurs des classes avants, arrière et charnière qu'on fait en sorte que la valeur de position soit dans le bon intervalle.

Question 2 : Les actions Il vaut mieux séparer au maximum les différentes interfaces (c'est le I de SOLID qui correspond en anglais à *Interface segregation principle*). La seule exception est l'interface qui correspond au jeu au pied pour laquelle le texte indique assez clairement qu'il faut regrouper les deux méthodes.



Exercice 2 (Abstraction)**.** On se penche sur le cas d'un MMOG (*Massive Multiplayer Online Game*).

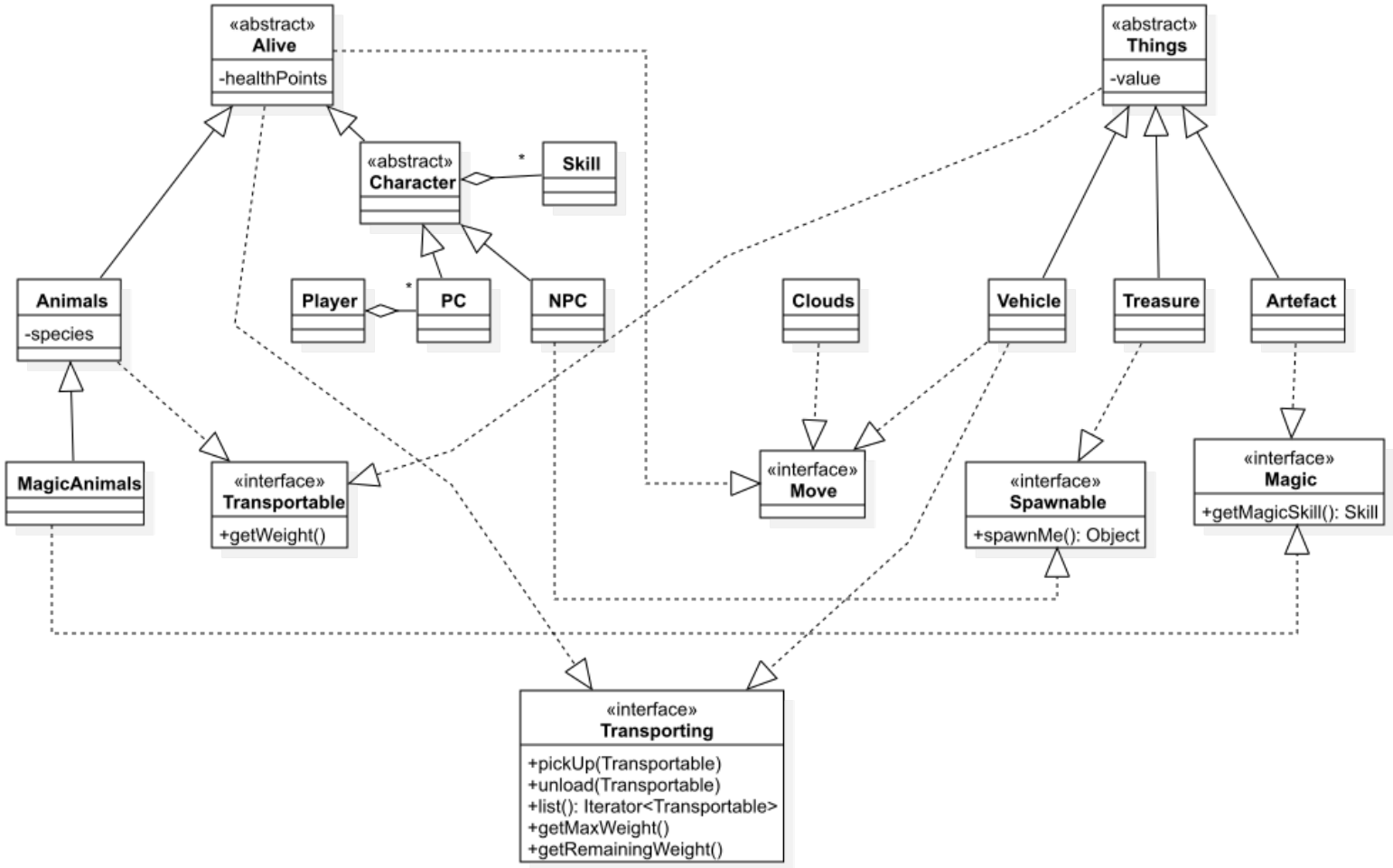
Le jeu gère des êtres vivants, ce qui signifie qu'ils ont des points de vie. Les animaux et les personnages sont vivants. Les animaux ont une espèce. Il y a deux sorte de personnages. Les PJ (personnages joueurs) qui sont contrôlables par un joueur du jeu et les PNJ (personnages non joueurs) qui sont gérés par le jeu. Les personnages ont des capacités. Les objets ont une valeur. Il y a trois sortes d'objets : les trésors, les artefacts et les véhicules. Le système gère aussi des nuages pour faire joli. Les animaux, les personnages, les véhicules et les nuages peuvent *se déplacer*. Les objets et les animaux sont potentiellement *transportables*. Ils ont un poids. Les véhicules, les animaux et les personnages peuvent potentiellement *transporter* des choses transportables. Ils ne peuvent pas dépasser un certain poids total. Ils peuvent évidemment *déposer* des choses. Les artefacts et certains animaux sont *magiques*, c'est-à-dire qu'ils donnent une capacité. Le moteur de jeu recopie de temps à autres des éléments de manière automatique, par exemple les trésors

et les PNJs. Ces éléments sont dit *spawnable*.

Proposez un diagramme de classe en détaillant bien les attributs et méthodes nécessaires (sauf s'il s'agit d'accesseurs classiques pour ne pas surcharger le diagramme).

Réponse.

La partie vraiment intéressante concerne le mécanisme pour transporter des choses transportables. On voit bien l'intérêt de passer par des interfaces pour manipuler des classes concrètes vastement différentes.


«abstract»
Alive
-healthPoints
«abstract»
Things
-value
«abstract»
Character
*
Skill
Animals
-species
Player
*
PC
NPC
Clouds
Vehicle
Treasure
Artefact
MagicAnimals
«interface»
Transportable
+getWeight()
«interface»
Move
«interface»
Spawnable
+spawnMe(): Object
«interface»
Magic
+getMagicSkill(): Skill
«interface»
Transporting
+pickUp(Transportable)
+unload(Transportable)
+list(): Iterator<Transportable>
+getMaxWeight()
+getRemainingWeight()