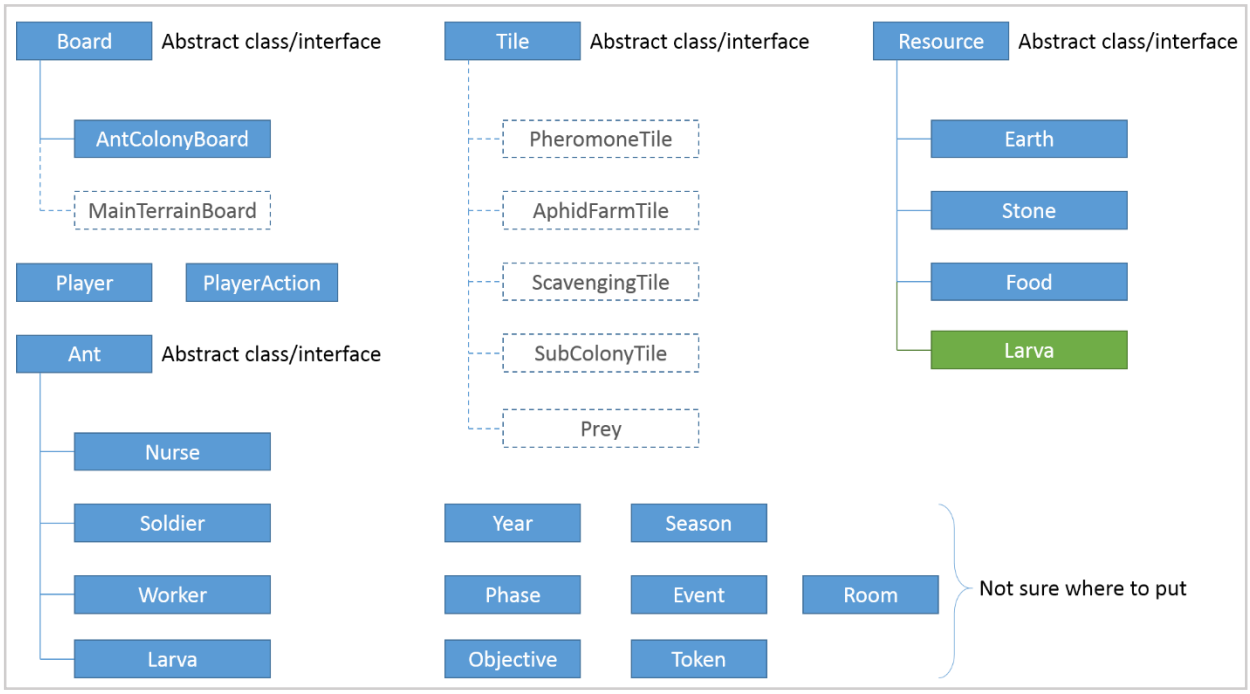
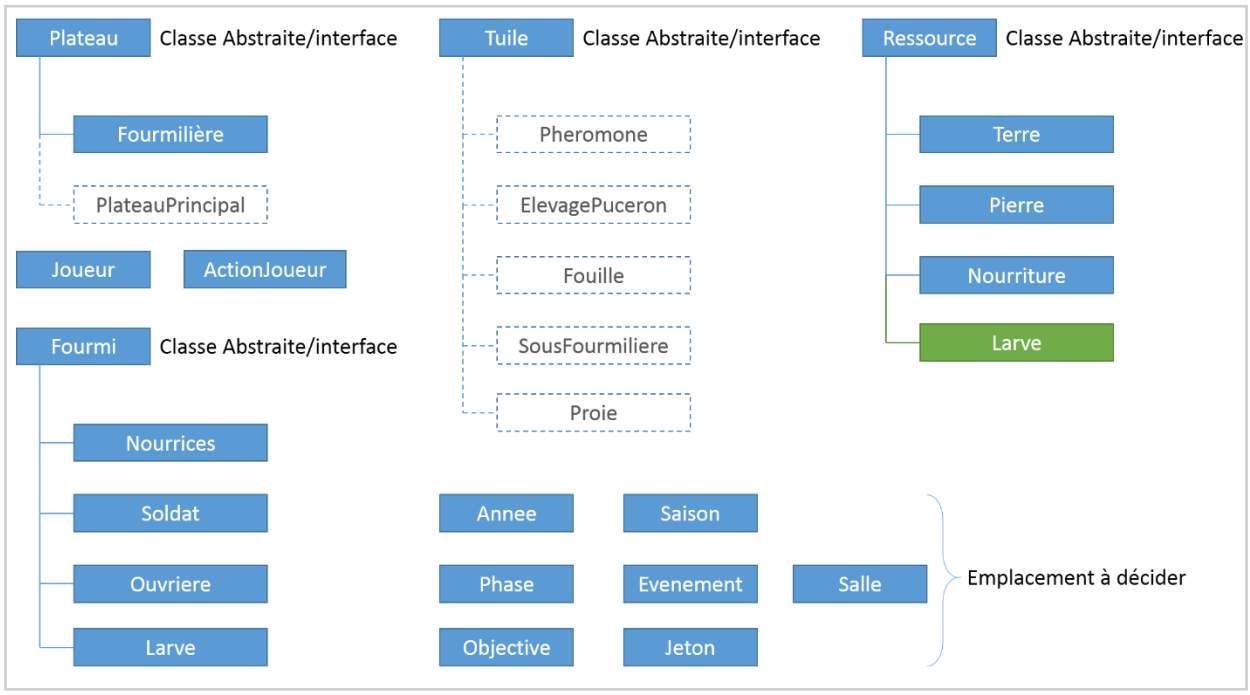
Modelling Myrmes the Board Game in Java

In this first version, only the ant colony board will be developed in detail. *On va d'abord modéliser les plateaux individuels*.

Classes

Myrmes should be modelled such that it follows the rules of the board game. There are different elements in the game such as the main playing board (*plateau principal)*, the ant colony board (*fourmilière/plateau individual*), the seasons – Spring, Summer, Autumn and Winter, the ants – larvae, nurses, workers and soldiers, the resources – earth (*terre*), stone (*pierre*) and food (*nourriture*) and so on. The following is a tentative list of classes that need to be created in order to model this board game. *Voici une liste prévisionnelle des classes à développer pour modéliser Myrmes – en pointillé c'est à gérer plus tard*) :





The ant colony board

In a game of Myrmes, there are four ant colony boards (one for each player – *ce sont les plateaux individuels*). This what the ant colony board looks like:

Figure 1 – The ant colony board - plateau individuel

To Model the AntColonyBoard (*Fourmilière*), a few other classes and enumerations (enums) will have to be developed as well. **The purple worlds are elements which might need to be developed as classes or enums, or they might be variables in some of the classes mentioned previously. Ideas for methods and method names are given in green.**

A) Event track : indicates the event of the current season

During the first phase of each round (*la première phase de chaque saison*), All the players can choose to sacrifice a certain number of larvae in order to influence the event which was decided by the dice corresponding to the season.

- A season has a random number between 1 to 6 to represent the dice that was thrown at the beginning of each year.
- All three dice are thrown simultaneously at the beginning of each year – **the events for Spring, Summer and Autumn are known from the start of each year**.
- The class Event might have methods like shiftLeft and shiftRight

B) Larvae room and birth track

This room keeps track of the number of larvae that a player has. Players can also place Nurses in this room to birth new larvae.

- The number of larvae of larvae (the birth track) that a player has should be included in the Player class
- The class LarvaeRoom might have methods like addLarva, useLarva.

- LarvaeRoom might also have a method like nurseCount, which returns the number of nurses that have been placed in the larvae room. Maybe also addNurse and removeNurse.

C) Nursery, where nurses are stored (and born)

In the beginning of the game, all players have 3 nurses in the nursery. When a player chooses the "new nurse" action in the atelier in phase 5 (by placing a nurse in the atelier room and spending 2 food cubes and 2 larvae), they receive a new nurse in the nursery.

- We have to think of the initialization of the game, including the initialization of the ant colony board. We could use an init function to put the correct number of ants in each room.
- The class Nursery (which would be subclass of Room) might have a method addNurse or newNurse.

D) Soldier room, where soldiers are stored (and born)

In the beginning of the game, the soldier room is empty – None of the players have soldier ants. The player can create new soldier ants by placing the necessary number of nurses in the soldier room.

- The soldierRoom class might have a method called addSoldier which checks to see if a sufficient number of nurses have been placed, and then creates a new soldier.

E) Worker room, where workers are stored (and born)

Similar to the soldier room, but for worker ants.

F) Atelier, where nurses that are not participating in births will be able to go to work

If a player places at least one nurse room in phase 2, they can perform the following actions in phase 5:

- increase the ant colony level by digging (*creuser*)
- complete an objective
- get a new nurse
- create new tunnel (*nouvelle galerie – créer une nouvelle entrée/sortie vers la fourmilière dans le jardin sur le plateau principal*)

The nurse that is placed in the atelier room in phase 2 is moved to the atelier on the main board and remains there.

Each round, each player can perform each action only once (*c.f. pg 9 des règles du jeu*)

G) Storeroom, where resource cubes will be stored

All resource cubes gained during the harvest in phase 4 (and by the workers in phase 3) are placed in this room. Resources are used in winter, and also for completing actions and objectives.

- It seems to me that the storeroom is really just the information that might be stored in variables like nbEarth, nbFood, nbStone in the Player class.

H) Ant colony level

The current ant colony level of the player determines what type of work the worker ants can do during one round. In the beginning of the game, on the level 0 is accessible. The ant colony level also determines what options are available for the workers that go outside (into the garden in the main board)

The player can place only one worker in each level of the colony. The ant colony level can be increased by completing the "increase the ant colony level by digging" action (*creuser*) in the atelier.

When a worker is placed in a certain level of the colony, the resource gathered is immediately added to the storeroom (no need to wait for the harvest phase)

A remark about the rooms in the ant colony board

It seems to me that the rooms are like counters (*ce sont des compteurs*). For example, the storeroom (le stock) just keeps track of the number of resources that the player has – perhaps it doesn't need to be modelled as a class? The counter argument (*contre argument*): if we modelled only some of the rooms, the code will not be homogenous. Also, if we model it as a class, then the advantage is that it makes it easier to modify the game rules later on, and also to add a graphical user interface more easily.

I also notice that the larva room, nursery, soldier room and worker room are all used for birthing. They can probably be grouped in a superclass called birthRoom (*ce sont tous des salles de naissances*).

Larvae – are they ants, or resources? And what about soldiers?

We have to decide if we will model the larvae as ants or resources. I think they should be resources, because they are treated as resources in the game (*on les dépense de la même manière que les cubes de terre, pierre et nourriture*). Furthermore, they cannot be moved around the ant colony board or the main board, unlike the worker and nurse ants (*les larves ne se déplacent pas*).

Similarly, I think soldiers should be considered as resources, because they cannot be moved around, but they can be used to replace food cubes. Also, the number of soldiers determines the type of prey that can be hunted, so they are more like a resource. The difference is that they are a resource that are not harvested, but instead they are born. *En effet, ce sont de ressources qu'on ne recolte pas, mais qu'on fait naître*.

Player Actions

I'm not sure if player actions should be a class of it's own. But I think that we need to think about the interface between the game and the player. Each turn, the player must be able to issue certain commands. For example:

- In phase 1 (Event) , the player can choose to sacrifice larvae to influence the event of the season

- In phase 2 (Births), the player chooses where he want to put his nurse ants. He can put them in birthing rooms to get more ants, or he can put them in the atelier room to complete actions and objectives in phase 5

- In phase 3 (Workers), the player chooses where he wants to put his worker ants. He can put them in the colony (1 worker per level), or he can send them outside to the Garden. (*Eventuellement, le joueur devrait pouvoir déplacer ses ouvrières dans le jardin et metre des phéromones, mais on gère ça plus tard*)

- In phase 4 (Harvest), there are mandatory harvests (*récoltes obligatoires*) and optional ones. The mandatory harvests are applied systematically, and so they do not require user input. (les cubes nourrices recolté par l'élevage de pucerons, terre et pierre par fouillage, la recolte grace à des phéromones sont tous mises dans le stock automatiquement et systématiquement sans intervention de la part du joueur). User input is only needed for the optional harvest (for example, the harvest+3 event which gives the player a bonus on their harvest for the season).

- In phase 5 (Atelier), only players who placed a nurse in the atelier room in phase 2 will participate – user input is only asked from these players. They can perform the actions in the atelier, including accomplish objectives.

Tokens

I think that tokens do not need to be modelled as a class (à la limite, si on faisait une interface graphique, peut-être qu'on modéliserait les jetons, mais encore, ce serait pas trop utile je pense). Tokens are just symbols used when playing the board game in real life to represent the current state of things -> In fact, that's exactly what **instance variables** do! En effet, les jetons ce ne sont qu'une representation physique pour rappeler les joueurs de l'état du jeu et les différentes données, et c'est ce qui est représenté par les variables d'instances.

Events

The seasonal events are decided randomly at the beginning of the year. The event is a sort of bonus that applies to all the players. The player can change the event that affects him for a particular season by sacrificing larvae at the beginning of the season.

There are some events that affect births, some that affect harvests, and some that affect the player's victory points. We have to think of how we'll handle these bonuses in our model of Myrmes. I think we should use an enum to enumerate the possible events.

Methods

Referring to M. Sanders' slides (link here), to write quality code, we must aim for high cohesion and low coupling. *Voici un extrait de son cours qui explique ces concepts:*