Projet Java : Generic Board Game

TP2 - Projet Java

Vers une gestion de plateau de jeu

Jérôme Buisine
Florian Leprêtre

jerome.buisine@univ-littoral.fr
florian.leprêtre@univ-littoral.fr

5 avril 2021

Durée : 6 heures

La finalité de la suite des TPs proposés, est de concevoir un système permettant à des utiliseurs de jouer à des jeux cartes en réseau. Les premiers jeux de cartes développés seront le jeu de Bataille classique ainsi qu'une version simplifiée du jeu de Poker.

1 Travail

L'objectif de ce TP est de continuer de développer notre framework de jeux. Nous allons dans le cadre de ce TP ajouter la notion de plateau au travers du jeu du Poker (version simplifiée).

1.1 Objectifs

L'objectif global est de mettre en place un système de jeu où chaque joueur peut se connecter à un serveur de jeu. Un serveur de jeu sera donc lié à un jeu qu'il héberge. Une fois le nombre de joueurs connectés attendu pour le jeu, le jeu peut démarrer. Le jeu de Poker y sera maintenant intégré !

Toujours orienté sur un système de jeu de 52 cartes, le jeu de Poker va également regrouper un ensemble de joueurs. La notion de **round** est également présente. Les joueurs auront maintenant en main des jetons (de certaines valeurs) qu'ils pourront miser en fonction de leur jeu (cartes en main). Au travers du bluff ou non, un joueur va essayer de récolter un maximum de jetons. Les règles envisagées dans le cadre du TP seront détaillées par la suite.

1.2 Environnement de travail

L'environnement de travail sera le même que le TP précédent :

— 1. IntelliJ/Idea (version Ultimate) comme environnement de développement (ou Eclipse en fonction de votre préférence) ;

— 2. Le langage Java avec la version 8 de Java SE ;

— 3. Git comme système de versionning du projet ;

— 4. Gitlab comme serveur d'hébergement de votre projet **Git**.

1.3 Récupération de la correction

En suivant le tutoriel suivant, il vous sera possible de récupérer la correction du TP précédent. Il vous faudra pointer sur la branche TP1.

2 Développement du jeu

Nous allons maintenant développer un second jeu dans notre framework, le jeu de Poker simplifié. L'objectif du Poker est de posséder une combinaison de cartes qui sera la plus forte à chaque fin de **round** afin de récupérer l'ensemble des mises du plateau.

2.1 Régles du jeu

Voici les différentes étapes du déroulement du jeu de Poker envisagé :

— En début de partie, chaque joueur reçoit le même nombre de jetons ;

— À chaque nouveau **round**, l'ensemble des 52 cartes sont mélangées aléatoirement ;

— Les cartes du jeu sont distribuées une à une à chacun des joueurs, jusqu'à ce que chaque joueur en possède **3**. Les cartes restantes sont posées face cachées, formant la pile ;

— En fonction de leur jeu en main, chaque joueur mise ici un jeton de valeur de son choix (jeton disponible dans sa main) et le pose sur le plateau de jeu (on limite ici le fait qu'un seul jeton soit misé par tour de mise) ;

— 3 nouvelles cartes (du haut de la pile) sont ensuite dévoilées à l'ensemble des joueurs et posées sur le plateau de jeu ;

— Les joueurs peuvent en fonction de leur main, c'est-à-dire maintenant le jeu disponible et leur main (cartes posées sur le plateau et cartes en main), miser de nouveau un jeton. Un joueur qui mise un jeton d'une grande valeur est généralement confiant de gagner (ou alors il bluff très bien) ;

— Une fois le dernier tour de mise de jetons réalisé par les joueurs, chaque joueur dévoile son jeu. Le joueur ayant le plus grand nombre de cartes identiques (et de plus grande hauteur si égalité) gagne le **round**. Il gagne l'intégralité de la mise disponible sur le plateau.C'est-à-dire que l'ensemble des jetons posés sur le plateau lui sont maintenant associés ;

— En cas d'une égalité extrême, c'est-à-dire, que par exemple le joueur 1 possède une paire de 8 et le joueur 2 également, l'attribution des jetons est aléatoire entre ces joueurs ;

— On considère ici que 3 **rounds** sont réalisés. Au bout de ces 3 rounds, la partie est terminée. Le gagant est le joueur ayant le plus de jetons en main.

2.2 Initialisation du jeu

Nous allons initialiser le jeu de Poker de la même manière qu'entrepris pour le jeu de la Bataille.

2.2.1 Ajout de la notion de plateaux

Le jeu de la Bataille tout comme le jeu de Poker peuvent être considérés comme des jeux de plateaux. Pour cela, nous allons introduire un nouvel élément à notre modélisation. L'interface `Board` proposée dans le diagramme UML disponible, correspond au contrat attendu d'une telle classe. La classe gérant un plateau.

Pour le jeu de la Bataille, le plateau accueillera temporairement les cartes posées par les joueurs. Dès qu'un joueur pose une carte, elle ne lui appartient plus, elle est présente dans le plateau.

Travail : Définir l'interface `Board` ainsi que la classe `CardBoard` qui accueillera un ensemble de cartes temporairement lors d'un round :

- L'interface `Board` sera stockée dans le même package que les autres interfaces ;
- Un nouveau package `ulco.cardGame.common.games.boards` accueillera les plateaux spécifiques ;
- L'état d'affichage de ce plateau sera l'ensemble des cartes posées sur ce plateau ;
- Ce plateau sera vide à chaque début de nouvelle manche ;
- La méthode `List<Component>` **getSpecificComponents(**`Class` **classType)** retournera l'ensemble des cartes, étant donné que ce plateau spécifique sera composé de cartes uniquement ;
- Vous pouvez maintenant ajouter la gestion de plateau à votre jeu de Bataille et l'afficher avant de déterminer le gagnant d'un round. Pour cela ajouter la gestion de plateau à votre framework de jeu (voir la classe `BoardGame` et l'interface `Game` du diagramme UML demandé) ;

2.2.2 Gestion des cartes et des jetons

Pour le jeu de Poker, on introduit la notion de jeton. Un jeton appartient à un joueur est peut être joué comme mise. Pour cela, une classe `Coin` qui héritera des fonctionnalités classiques de `Component` est nécessaire. Elle permettra de gérer un jeton par son identifiant, son nom (sa couleur) et sa valeur.

Un plateau sera également proposé pour le jeu du Poker. Celui-ci stockera temporairement pour chaque round les jetons misés par les joueurs et les 3 cartes retournées du round en cours. La classe `PokerBoard` définira ce plateau en gérant **à la fois** une liste de cartes, et une liste de jetons. La méthode `List<Component>` **getSpecificComponents(**`Class` **classType)** retournera la liste souhaitée de composants (cartes ou jetons) en fonction du type de classe passé en paramètre.

Travail : Ajouter la classe `Coin` à vos composants de jeu disponibles de votre framework puis définir la classe `PokerBoard` qui permettra de gérer le plateau du jeu du Poker.

Note : il est possible de comparer un type d'élément à un autre de la manière suivante :

```
// Using equality type comparison
if (classType == Card.class)
   System.out.println("It's a card");
else
   System.out.println("Not a card");
```

2.3 Le jeu de Poker

C'est ici que les choses vont devenir intéressantes !

2.3.1 Le joueur de Poker

Pour gérer le jeu de Poker, il faudra ajouter un nouveau type de joueur spécifique à ce jeu. On définira donc la classe **`PokerPlayer`** qui gérera une liste de cartes et une liste de jetons, toutes deux associées à notre joueur. La subtilité ici sera de gérer correctement l'association de composant à la liste de composants ciblées.

Vous pouvez vous appuyer sur l'exemple suivant :

```
// Using instanceof keywork
if (component instanceof Card)
        cards.add((Card)component);
```

Un nouveau comportement est ici requis, le fait de vider la main du joueur. En effet, dans le jeu du Poker à chaque fin de round, le jeu complet se mélange de nouveau. Cela revient au fait de supprimer les cartes de la main de chaque joueur et de les associer de nouveau au jeu. Pour cet aspect, vous pouvez vous baser sur les comportements souhaités de l'interface **`Player`** (voir le diagramme UML demandé). L'affichage de la main du joueur est également souhaité (cartes en main, jetons en main et somme des jetons).

Travail : Ajouter la classe **`PokerPlayer`** avec les comportement attendus et modifier la classe **`CardPlayer`** avec les nouveaux comportements communs souhaités.

2.3.2 Vers la boucle de jeu

Avant de gérer la boucle de jeu du Poker, il vous faut générer l'instance du jeu. Pour cela, un nouveau fichier de jeu est disponible dans le dossier **`resources/games`**. Il comprend l'instance du jeu de Poker qui comprend les 52 cartes comme précédemment mais également les jetons attendus pour **chaque** joueur de la partie. La couleur est associée au nom du jeton.

Lorsqu'un joueur souhaite jouer (méthode `play`), il doit choisir un jeton parmi sa liste de jetons disponibles. Dans ce sens, il faut bien vérifier qu'il joue un jeton dont la couleur est autorisée et est disponible en main. Si erreur de saisie il y a, la boucle de jeu lui demandera de nouveau un choix de couleur de jeton jusqu'à une saisie valide.

Travail : Créer la classe **PokerGame** puis définir la méthode `initialize` pour charger l'instance du jeu relativement au fichier proposé. Une fois l'instance du jeu chargée, implémenter la méthode `run` gérant la boucle de jeu. Il vous faudra bien reprendre les régles du jeu définies en section 2.1.

À noter que vous pouvez exploiter l'exemple suivant pour créer vos composants depuis le fichier d'instance :

```
String className = "Card";
// true or false
className.equals(Card.class.getSimpleName())
```

3 Bonus

On peut supposer qu'un joueur ne souhaite pas miser et passer son tour. Intégrer cette fonctionnalité en prenant en compte :

— Si le joueur passe son tour alors il ne soumet aucun jeton. Il retourne donc une instance nulle que le jeu devra traiter ;

— Si le joueur passe son tour, alors il ne pourra plus jouer pour cette manche et ne remportera aucun gain même s'il était en possession de la meilleure main ;

— Si seul un joueur continue de jouer, il remporte directement l'ensemble des gains même si son jeu est moins fort que l'un de ses adversaires. La manche sera donc directement terminée.