

---

## Projet

---



Le but de ce projet est de produire une implémentation en *Java* du jeu de cartes *Dominion*.

### 1 Présentation du jeu

*Dominion* est un jeu de cartes inventé par Donald X. Vaccarino pour 2 à 4 joueurs. C'est un jeu de *construction de deck* où les joueurs doivent à tour de rôle jouer des cartes de leur main pour obtenir de nouvelles cartes parmi un ensemble commun disponible. Lorsque la partie s'arrête le joueur possédant le plus de points de victoire dans son deck est déclaré vainqueur.

Vous pouvez consulter les règles du jeu dans le fichier « *Dominion-Règles.pdf* » disponible sur l'ENT (ainsi qu'une version en anglais « *Dominion-Rules.pdf* »).

**Vous êtes encouragés à lire ces règles avant de poursuivre la lecture du sujet.**

**Remarque :** Il existe de nombreuses extensions pour *Dominion* qui modifient considérablement les règles de base. Dans ce projet nous ne considérerons que le jeu de base.

### 2 Les cartes

Toutes les cartes du jeu (cf. section 7 pour la liste complète) sont caractérisées par

- un nom (en haut) ;
- un prix d'achat (en bas à gauche) ;
- un type (en bas) qui peut être *Trésor*, *Victoire*, *Malédiction*, *Action*, *Action/Attaque* ou *Action/Réaction* ;
- une description (centre) qui correspond à l'effet de la carte lorsqu'elle est jouée (*Action* ou *Trésor*) ou comptabilisée en fin de partie (*Victoire*).

Il peut exister plusieurs copies de chaque carte mais deux cartes ayant le même nom ont des caractéristiques identiques (on dira que deux cartes identiques sont de la même *famille*).

Par ailleurs, l'ensemble des cartes utilisées est déterminé en début de partie et à tout moment de la partie chaque carte est

- soit dans la *réserve* (*supply*), commune à tous les joueurs et disponible à l'achat ;
- soit dans la pile de cartes *écartées* (*trash*) ;
- soit en la possession d'un joueur.

## 2.1 La classe Card et ses sous-classes

Les cartes du jeu sont représentées par des objets de la classe **Card**. Chaque carte utilisée par la partie est représentée par une instance différente, et toutes les instances sont créées au démarrage de la partie (celles qui sont initialement distribuées aux joueurs peuvent éventuellement être instanciées à la création des instances représentant les joueurs).

Étant donné que le *type* d'une carte influe fortement sur la façon d'utiliser la carte, on définit une sous-classe de **Card** pour chacun des types possibles (cf. figure 2.1) :

**TreasureCard** pour les cartes *Trésor* ;

**VictoryCard** pour les cartes *Victoire* ;

**CurseCard** pour les cartes *Malédiction* ;

**ActionCard** pour représenter les cartes *Action*.

Les types composés *Action/Attaque* et *Action/Réaction* sont représentés par deux sous-classes de **ActionCard** : **AttackCard** et **ReactionCard** respectivement.

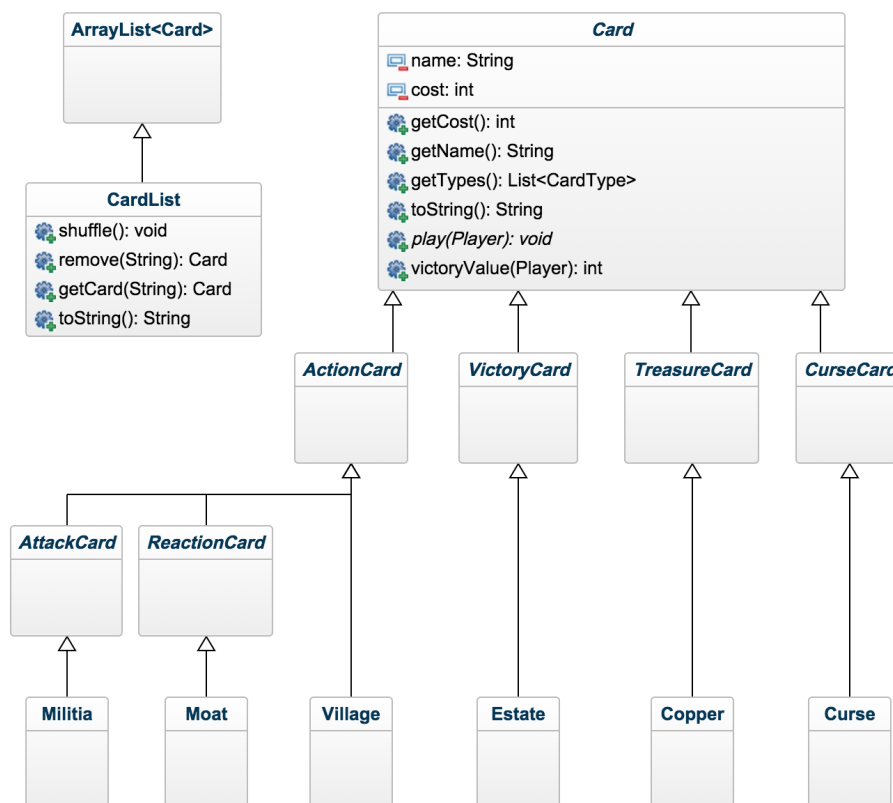


FIGURE 1 – Diagramme de classes de la hiérarchie représentant les cartes (pour alléger le diagramme, une seule carte est représentée pour chaque type).

## 2.2 Les listes de cartes

Afin de simplifier les tâches couramment effectuées par le jeu sur des ensembles de cartes, on définit une classe spécifique **CardList** pour représenter les listes de cartes. C'est cette classe qui est utilisée pour représenter toutes les listes de cartes en possession des joueurs (main, défausse, etc.) ainsi que les piles de la réserve et la pile de cartes écartées.

Cette classe hérite de **ArrayList<Card>** ce qui permet de l'utiliser comme un **ArrayList** (constructeurs, accesseurs, méthodes) et d'ajouter des méthodes lorsque c'est utile.

La classe `CardList` est fournie avec un certain nombre de méthodes supplémentaires déjà implémentées mais vous pouvez éventuellement ajouter des méthodes si vous le jugez nécessaire.

## 2.3 Les types de cartes

Pour représenter correctement les différents types possibles des cartes, un type énuméré `CardType` est fourni contenant les constantes

- `Treasure`
- `Action`
- `Victory`
- `Curse`
- `Reaction`
- `Attack`

C'est un `ArrayList` d'éléments de ce type qui est renvoyé par la méthode `getTypes()` de la classe `Card` (les cartes du jeu de base peuvent avoir un ou deux types).

## 3 Les joueurs

Les joueurs de la partie sont identifiés par un nom (de type `String`). À tout moment de la partie, les cartes que possède un joueur peuvent être dans l'un des 4 emplacements suivants :

- sa *main* (*hand*) ;
- sa *défausse* (*discard*) ;
- sa *pioche* (*draw*) ;
- la liste des cartes actuellement *en jeu* (*in play*).

L'ensemble des cartes possédées par le joueur (dans l'un des emplacements précédemment cités) constitue le *deck* du joueur. Chaque joueur commence la partie avec 3 cartes *Estate* et 7 cartes *Copper* dans sa défausse (et en pioche immédiatement 5 en main).

En plus de ses cartes, un joueur a différents compteurs de ressources :

- le nombre d'*actions* qu'il peut jouer (initialisé à 1 au début de son tour) ;
- le nombre de pièces dont il dispose pour acheter des cartes (initialisé à 0 au début de son tour) ;
- le nombre d'achats qu'il peut réaliser (initialisé à 1 au début de son tour)

### 3.1 Déroulement du tour

Le tour d'un joueur s'exécute en plusieurs étapes

**Préparation.** Les compteurs du joueur sont remis aux valeurs indiquées par les règles : 1 pour les actions et les achats, et 0 pour l'argent.

**Actions.** Le joueur peut jouer des cartes *Action* de sa main tant que son compteur d'actions est supérieur ou égal à 1. Lorsqu'une carte *Action* est jouée, le compteur d'actions du joueur est décrémenté de 1, la carte jouée est marquée comme étant *en jeu* et l'action de la carte est exécutée. Le joueur peut choisir de passer à la phase suivante même s'il lui reste des actions qu'il peut jouer.

**Trésors.** Le joueur peut jouer des cartes *Trésor* de sa main. Dans le jeu de base, il n'y a aucune situation où le joueur aurait un intérêt à conserver des trésors dans sa main. On pourra donc considérer ici que le joueur joue automatiquement tous les trésors qu'il a en main.

**Achats.** Le joueur peut acheter des cartes de la réserve en utilisant l'argent qu'il a amassé pendant les phases précédentes. Le joueur peut acheter une carte s'il lui reste au moins

un achat et que le prix de la carte est inférieur à la somme dont il dispose. Lorsqu'il achète une carte, son compteur d'achats est décrémenté de 1, son argent de la valeur de la carte achetée et la carte achetée est déplacée dans la défausse du joueur. Le joueur peut choisir de terminer cette phase même s'il peut encore acheter des cartes.

**Fin.** À la fin du tour toutes les cartes de la main du joueur et en jeu sont défaussées, les compteurs du joueur sont remis à 0 et le joueur pioche 5 nouvelles cartes en main.

**Remarque :** Il est important que les cartes en main soient piochées à la fin du tour car la main peut être affectée pendant le tour d'un autre joueur (cf. *Militia* ou *Moat* par exemple).

### 3.2 La classe Player

Les joueurs participant à une partie de *Dominion* sont représentés par des instances d'une classe **Player**. Le nom, les compteurs (actions, argent, achats), les différentes piles de cartes du joueur ainsi que la partie dans laquelle il se trouve sont représentés par des attributs.

La figure 2 illustre les attributs et méthodes qui doivent être implémentés dans la classe **Player** (voir le code fourni pour les spécifications de chaque méthode). Vous pouvez cependant ajouter des méthodes si vous le jugez nécessaire.

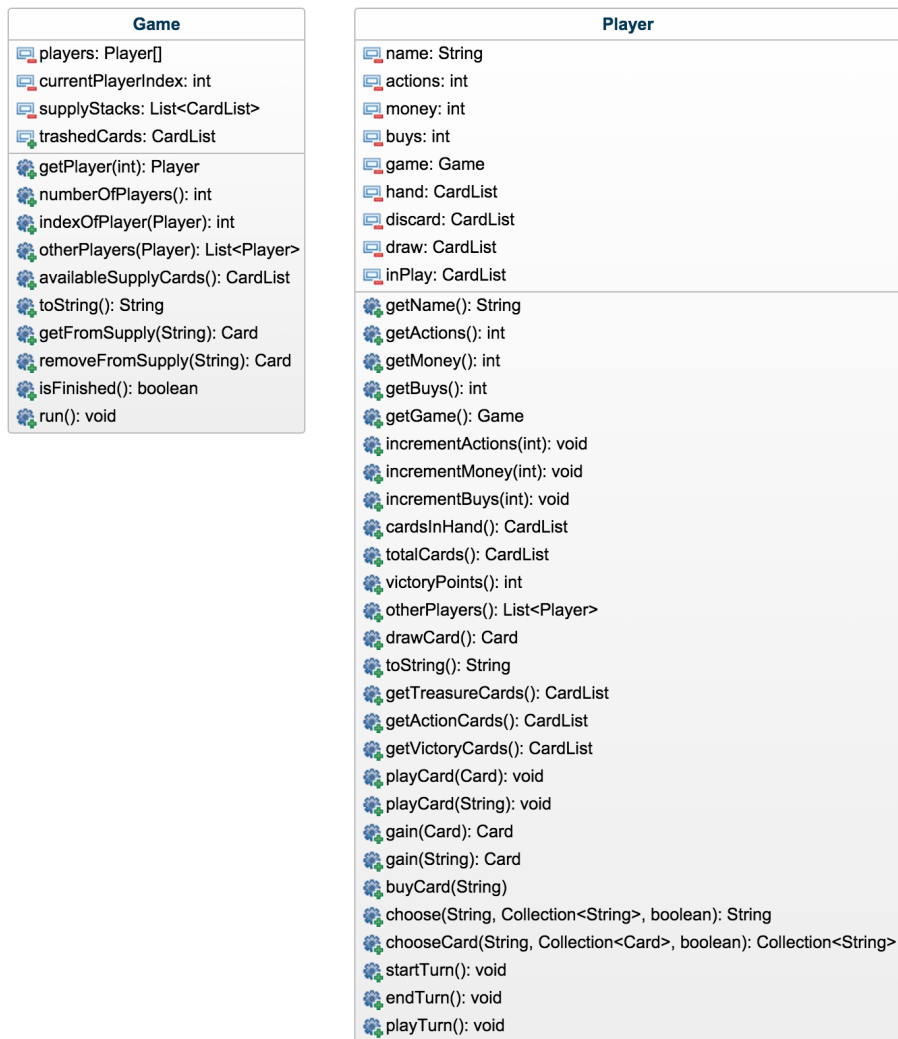


FIGURE 2 – Diagramme des classes **Game** et **Player**.

## 4 La partie

Une partie de *Dominion* est représentée par une instance de la classe **Game**. C'est la partie qui gère la liste des joueurs, l'ensemble des cartes communes et qui contrôle le déroulement de la partie : mise en place, alternance des tours des joueurs et fin de partie lorsque les conditions de fin sont remplies.

Pour démarrer une partie, il faut spécifier le nombre de joueurs qui y participent ainsi que la liste des cartes à utiliser comme piles de réserve. Le constructeur de la classe **Game** prend donc deux arguments :

**String[] playerNames** : la liste des noms des joueurs qui participent à la partie (c'est le constructeur de **Game** qui construit les instances de **Player** correspondantes)

**List<CardList> kingdomStacks** : une liste de piles de réserve à utiliser comme cartes *Royaume* dans la partie. Les règles du jeu prévoient 10 piles *Royaume* mais la partie peut fonctionner avec un nombre différent. Le constructeur de **Game** doit ajouter à ces piles les piles de réserve communes (cartes *Trésor*, *Victoire* et *Malédiction*)

La classe **Game** est relativement simple (par rapport à **Player**), et la figure 2 illustre les attributs et méthodes qu'il vous est demandé d'écrire. Comme précédemment, vous pouvez ajouter des méthodes si cela vous semble nécessaire.

## 5 Interface utilisateur

Pour des raisons de simplicité, seule une interface en ligne de commande vous est demandée. Une fois la partie lancée, toutes les interactions avec l'utilisateur se feront donc dans le terminal. Les informations du jeu seront affichées à l'écran en utilisant la sortie standard et les choix des joueurs se feront par lecture sur l'entrée standard (clavier).

Dans une partie à plusieurs joueurs, un même processus demande successivement aux joueurs de jouer leur tour dans le même terminal.

### 5.1 Choix

Lorsqu'un joueur doit choisir parmi un ensemble de cartes (par exemple la carte à jouer dans sa main, ou une carte à acheter dans la réserve ou encore une carte parmi deux *Trésors* après avoir joué un *Voleur*), il devra entrer le nom exact de la carte choisie (on utilisera les noms anglais des cartes pour éviter les problèmes de caractères accentués). Par exemple, il pourra choisir "Village" ou "Throne Room".

Pour indiquer qu'il souhaite *passer* (par exemple parce qu'il ne souhaite pas jouer de carte *Action*, qu'il ne veut rien acheter en fin de tour ou encore s'il ne veut plus écarter de cartes après avoir joué une *Chapelle*), le joueur entrera la chaîne de caractères vide "".

Enfin, lorsqu'il doit faire un choix parmi deux possibilités (par exemple choisir s'il veut défausser son deck après avoir joué un *Chancelier*, ou s'il veut défausser une carte *Action* piochée après avoir joué une *Bibliothèque*), il répondra à une question posée en entrant "y" pour oui et "n" pour non.

## 6 Rendu attendu

Vous devez fournir une archive au format **zip** ou **tar** contenant les fichiers **.java** complétés. Cette archive doit respecter exactement la hiérarchie de répertoires et les noms de fichiers du code qui vous a été fourni (vous devriez simplement compléter les fichiers fournis).

Toutes les méthodes indiquées dans le code fourni doivent être complétées selon les spécifications (en respectant les noms et les paramètres indiqués). Vous pouvez éventuellement ajouter des attributs et méthodes aux classes, lorsque cela vous semble nécessaire.

L'exécution de la méthode `main` de la classe `Main` (à la racine du projet) doit démarrer une partie avec un ensemble de 10 cartes *Royaume* de votre choix (cette fonction servira simplement à tester manuellement la bonne exécution d'une partie, mais l'ensemble des cartes sera évalué par la suite de manière automatique, indépendamment du choix de cartes pour la partie exécutée par la méthode `main`).

## 6.1 Évaluation


L'évaluation du projet se fera à l'aide de tests automatisés. Un premier jeu de tests vous sera fourni avant la fin du projet pour que vous puissiez vérifier le bon fonctionnement des fonctionnalités de base, puis nous utiliserons un second jeu de tests (secret) pour l'évaluation finale.

Il est donc attendu que les projets rendus passent le premier jeu de tests sans erreurs, mais vous devez également vérifier par vous-mêmes que le projet se comporte correctement dans les différents cas particuliers qui peuvent se produire et qui ne sont pas nécessairement couverts par ces premiers tests.


## 7 Liste des cartes

### 7.1 Cartes communes



*Cuivre (Copper)*  
+ 




*Argent (Silver)*  
+ 




*Or (Gold)*  
+ 



*Domaine (Estate)*  
1 




*Duché (Duchy)*  
3 



*Province*  
6 



*Malédiction (Curse)*  
-1 



## 7.2 Cartes royaume



### *Cave (Cellar)*

+1 Action.

Défaussez autant de cartes que vous voulez.

+1 Carte par carte défaussée.



### *Chapelle (Chapel)*

Écartez jusqu'à 4 cartes de votre main.



### *Douves (Moat)*

+2 Cartes.

Lorsqu'un adversaire joue une carte *Attaque*, vous pouvez dévoiler cette carte de votre main.

Dans ce cas, l'*Attaque* n'a pas d'effet sur vous.



### *Chancelier (Chancellor)*

+2.

Vous pouvez immédiatement défausser votre deck.



### *Village*

+1 Carte.

+2 Actions.



### *Bûcheron (Woodcutter)*

+1 Achat.

+2.



### *Atelier (Workshop)*

Recevez une carte coûtant jusqu'à 4.



### *Bureaucrate (Bureaucrat)*

Recevez une carte *Argent*; placez-la sur votre deck.

Tous vos adversaires dévoilent une carte *Victoire* et la placent sur leur deck (sinon ils dévoilent leur main afin que vous puissiez voir qu'ils n'ont pas de cartes *Victoire*).



### *Festin (Feast)*

Écartez cette carte.

Recevez une carte coûtant jusqu'à 5.



### *Jardins (Gardens)*

Vaut 1 pour chaque 10 cartes dans votre deck (arrondi à l'unité inférieure).





### *Milice (Militia)*

+2.

Tous vos adversaires défaussent leurs cartes de façon à n'avoir que 3 cartes en main.



### *Prêteur sur gages (Moneylender)*

Écartez une carte *Cuivre* de votre main.

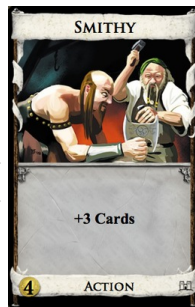
Dans ce cas, +3.



### *Rénovation (Remodel)*

Écartez une carte de votre main.

Recevez une carte coûtant jusqu'à 2 de plus que la carte écartée.



### *Forgeron (Smithy)*

+3 Cartes.



### *Espion (Spy)*

+1 Carte.

+1 Action.

Tous les joueurs (vous aussi) dévoilent la première carte de leur deck. Vous décidez ensuite si chaque carte dévoilée est défaussée ou replacée sur son deck.



### *Voleur (Thief)*

Tous vos adversaires dévoilent les 2 premières cartes de leur deck. S'ils dévoilent des cartes *Trésor*, ils en écartent 1 de votre choix. Parmi ces cartes *Trésor* écartées, recevez celles de votre choix. Les autres cartes dévoilées sont défaussées.



### *Salle du trône (Throne Room)*

Choisissez 1 carte *Action* de votre main.

Jouez-la deux fois.



### *Chambre du conseil (Council Room)*

+4 Cartes.

+1 Achat.

Tous vos adversaires piochent 1 carte.



### *Festival*

+2 Actions.

+1 Achat.

+2.



### *Laboratoire (Laboratory)*

+2 Cartes.

+1 Action.



### *Bibliothèque (Library)*

Piochez jusqu'à ce que vous ayez 7 cartes en main. Chaque carte *Action* piochée peut être mise de côté. Défaussez les cartes mises de côté lorsque vous avez terminé de piocher.



### *Marché (Market)*

+1 Carte.  
+1 Action.  
+1 Achat.  
+1.



### *Mine*

Écartez une carte *Trésor* de votre main. Recevez une carte *Trésor* coûtant jusqu'à 3 de plus ; ajoutez cette carte à votre main.



### *Sorcière (Witch)*

+2 Cartes.  
Tous vos adversaires reçoivent une carte *Malédiction*.



### *Aventurier (Adventurer)*

Dévoilez des cartes de votre deck jusqu'à ce que 2 cartes *Trésor* soient dévoilées. Ajoutez ces cartes *Trésor* à votre main et défaussez les autres cartes dévoilées.