

# Programmation Orientée Objets avec Java

## Projet POO

**Stéphane Malandain / Yassin Rekik**  
**Semestre d'automne 2023**

### Introduction

Dans le cadre du cours de Programmation Orientée Objet, vous devez réaliser un projet entier. Ce travail se divise en 4 étapes. Chaque étape fait l'objet d'un rendu à une date fixe, avec une note. La moyenne des 4 notes vous donne une note de projet.

Le projet est individuel. Plagiat ou copie seront sévèrement puni !





Le but de ce projet est de modéliser différents jeux de cartes, puis de réaliser concrètement le jeu du blackjack, avec des joueurs humains et IA.

Les modalités de dépôt vous seront communiquées sur le Git ultérieurement.

### 1 Première étape : les bases d'un jeu de carte

Vous devez modéliser la notion de carte.

Un jeu de cartes traditionnel comprend les cartes suivantes :

- Coeurs: As · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · Valet · Dame · Roi
- Carreaux: As · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · Valet · Dame · Roi
- Piques: As · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · Valet · Dame · Roi
- Trèfles: As · 2 · 3 · 4 · 5 · 6 · 7 · 8 · 9 · 10 · Valet · Dame · Roi
- Autres: 2 jokers
- Couleurs: coeur , carreau , pique , trèfle 

Le jeu du tarot comprend en plus :

- Le cavalier (entre le Valet et la Dame)
- Les atouts : de 1 à 21, avec en plus une carte spéciale, l'excuse.

Une carte à une valeur de 2 à 10, Valet, Dame, Roi, As, ( éventuellement Cavalier pour le jeu du tarot) et une couleur.

Vous devez modéliser la classe carte (avec des variables de classes et d'instance). Une carte est représentée par sa couleur, son rang et sa force.

1.1 Compléter des constructeurs de la classe. Le premier constructeur prend en argument la couleur et le rang de la carte, tant dis que le deuxième prend en argument la couleur, le rang et la force de la carte. Les constructeurs permettent de construire une carte valide.

```
public Carte (COULEUR couleur, int rang)
public Carte (COULEUR couleur, int rang, int force)
```

1.2 Compléter des méthodes retournant la couleur, le rang et la force de la carte.

```
public COULEUR getCouleur ()
public int getRang ()
public int getForce ()
```

1.3 Compléter des méthodes retournant le nom de la couleur (selon le tableau `NOMS_COULEURS`) et le nom du rang de la carte (selon le tableau `NOMS_RANGS`). Par exemple, le nom du rang 0 est "Joker", le nom du rang 2 est "2" et celui du rang 13 est "Roi". Ensuite, compléter la méthode retournant le nom complet associé à la carte (ex, "Dame de Coeur", "10 de Trefle").

```
public String getNomCouleur ()
public String getNomRang ()
public String getNomComplet ()
```

Le résultat d'une carte affiché à l'écran doit être comme suit :

```
9 de Coeur
Dame de Carreau
Joker
10 de Trefle
```

Les cartes ne changent jamais de valeur ni de couleur. Il faudra donc faire des objets java non mutables (immuables).

Votre classe Java aura des attributs privés permettant de représenter les informations de la carte. Enfin, La classe doit interdire par construction la création d'objets ne correspondant pas à des cartes réelles (par exemple le 0 de pique, le cavalier de coeur ou le 8 de plomb).

1.4 Ensuite, créer une classe `JeudeCarte` permettant de créer un jeu de carte. Un jeu standard fait 32 cartes (8 cartes par couleurs) ou 52 cartes (13 cartes par couleur), 54 (52 + 2 jokers), 56 (52 + 4 cavaliers du tarot), 78 pour le tarot. Cette classe hérite d'un tableau pour le moment.

Vous devez créer également une méthode `shuffle()` qui vous renvoie un jeu mélangé ainsi qu'une méthode `paquet()` qui vous renvoie un tableau de n jeu de cartes mélangées.

Vous devez faire un petit menu permettant à l'utilisateur d'avoir les fonctionnalités suivantes :

- Création d'un jeu de 32, 52, 54, 56 ou 78 cartes
- Mélange du jeu de cartes.
- Affichage du jeu en cours (mélangé ou non)
- Création de 2 mains (une main se compose d'un ensemble de 10 cartes)
- Affichage des mains triées par couleur et valeur.
- Création d'un paquet de n jeu de cartes mélangées,
- Affichage du paquet

## 2 Seconde étape : héritage, polymorphisme et concrétisation.

### Indications préliminaires qui seront fixées ultérieurement.

Nous allons enrichir la notion de cartes, jeu de carte, de main, de pli et des fonctionnalités. Vous devez considérer dans cette étape différent type de jeu de carte possibles. Vous devez réaliser une méthode type `compareTo()` permettant de comparer 2 cartes en fonction du jeu et de leur rang.

Vous devez implémenter concrètement le blackjack, dont les règles vous sont données en annexe.

On choisit le nombre de joueur au départ, avec la somme dont chacun dispose. 1 joueur IA doit être développé avec une règle toute simple : tant qu'il n'arrive pas à 15, il continue. Sa mise est fixe à chaque tour.

## 3 Troisième étape : Listes, exceptions et programmation fonctionnelle

### Indications préliminaires qui seront fixées ultérieurement.

Maintenant que la modélisation des cartes et leurs manipulations est optimisée, il s'agit dans cette étape de ne plus gérer les cartes, jeu ou sabots dans des tableaux mais dans des collections pertinentes de votre choix.

Choisissez la structure de données adaptée et modifier en conséquence votre code des étapes précédentes.

Vous devez ajouter un second joueur ordinateur avec une IA un peu plus perfectionnée. Cette IA suit la stratégie suivante pour jouer :

Carte de la banque										
	2	3	4	5	6	7	8	9	10	A
8 ou moins	T	T	T	T	T	T	T	T	T	T
9	T	D	D	D	D	T	T	T	T	T
10	D	D	D	D	D	D	D	D	T	T
11	D	D	D	D	D	D	D	D	D	T
12	T	T	R	R	R	T	T	T	T	T
13	R	R	R	R	R	T	T	T	T	T
14	R	R	R	R	R	T	T	T	T	T
15	R	R	R	R	R	T	T	T	A	T
16	R	R	R	R	R	T	T	A	A	A
17 ou plus	R	R	R	R	R	R	R	R	R	R
A8-A9-A10	R	R	R	R	R	R	R	R	R	R
A7	R	D	D	D	D	R	R	T	T	T
A6	T	D	D	D	D	T	T	T	T	T
A5	T	T	D	D	D	T	T	T	T	T
A4	T	T	D	D	D	T	T	T	T	T
A3	T	T	T	D	D	T	T	T	T	T
A2	T	T	T	D	D	T	T	T	T	T
AA-88	S	S	S	S	S	S	S	S	S	S
10-10	R	R	R	R	R	R	R	R	R	R
9-9	S	S	S	S	S	R	S	S	R	R
7-7	S	S	S	S	S	S	T	T	T	T
6-6	T	S	S	S	S	T	T	T	T	T
5-5	D	D	D	D	D	D	D	D	T	T
4-4	T	T	T	T	T	T	T	T	T	T
3-3/2-2	T	T	S	S	S	S	T	T	T	T

T

R

D

S

A

Tirer

Rester

Doubler

Split

Abandon

Le but du Blackjack est assez simple dans les faits, **il faut battre un croupier en se rapprochant le plus possible des 21 points**. Le problème, c'est que si vous allez tout faire pour être le plus proche, sans stratégie de Blackjack établie, vous risquez d'aller trop loin et de dépasser ce score ce qui signifie la défaite.

C'est là où ce fameux tableau de Blackjack pour le casino intervient. Il sera pour vous une véritable béquille sur laquelle vous allez pouvoir vous appuyer afin de prendre la meilleure décision en fonction de la situation dans laquelle vous vous trouvez. Ainsi, vous augmenterez grandement vos chances de remporter la victoire finale. Voici dans les faits la marche à suivre avec votre tableau des stratégies de Blackjack :

- Regardez votre main et reportez le résultat sur la première colonne.
- Regardez la carte de la banque et reportez-la sur la colonne correspondante.
- Faites alors l'action demandée qui peut être de tirer, rester, doubler, split ou bien abandonner.

## 4 Quatrième étape : GUI avec JavaFX et tests unitaires

**Indications préliminaires qui seront fixées ultérieurement.**

La quatrième étape du projet consiste à réaliser un jeu de blackjack interactif avec un GUI (JavaFX) permettant de jouer.

Vous devez également dans cette étape ajouter la persistance des données, à savoir la possibilité de sauvegarder / restaurer les données au début et à la fin de chaque utilisation. Le choix du support de stockage (BD, Fichier, ...) est libre.

# Annexe

## Les règles du Blackjack



Le Blackjack se joue avec six jeux de 52 cartes (sixain) : trois d'une couleur et trois d'une autre. Le nombre de joueurs assis est max de 7, mais des joueurs debout peuvent miser sur la main d'un joueur assis avec l'autorisation de ce dernier. La mise minimale est de **10 €**.

### Règles du Blackjack

Obtenir 21 points ou s'en approcher le plus possible de manière à totaliser un nombre de points supérieur à ceux du croupier. Le joueur ou le croupier (la banque) perd s'il dépasse les 21 points. Une fois que le joueur a misé dans sa case, le croupier commence la partie en distribuant deux cartes à chaque joueur et une à lui-même. Chaque joueur peut alors demander des cartes supplémentaires jusqu'à ce qu'il s'estime satisfait.

Lorsque tous les joueurs ont déterminé la situation de leur main, le croupier tire une ou plusieurs cartes pour lui, dans le but d'atteindre un minimum de 17 points et un maximum de 21 points. Il est interdit à tout joueur de tirer une nouvelle carte à 21.

Suite à ce tirage, il :

- Paie à égalité la mise des joueurs dont les points sont supérieurs aux siens. Le **Black Jack (As + 10)** est payé 1,5 pour 1.
- Ramasse la mise des joueurs dont les points sont inférieurs aux siens.
- En cas d'égalité des points, le coup est nul.

### Valeur des cartes

Valet, Dame, Roi comptent 10 points

As compte 1 ou 11, à la convenance du joueur ou du croupier

Les autres cartes sont comptées à leur valeur effective

## **L'Assurance**

Dans le cas où la première carte du croupier est un as, le joueur peut s'assurer contre le Black Jack du croupier en misant dans la case « assurance » une mise égale à la moitié de sa mise initiale. Si la banque fait Black Jack, cette assurance sera payée 2 pour 1 (coup nul pour le joueur). Dans le cas contraire, les assurances sont perdues.

## **La double mise**

Quel que soit le nombre de points obtenu avec ses deux premières cartes, le joueur peut doubler sa mise.

Dans ce cas, il n'a droit qu'à une seule carte.

## **Les paires**

Lorsqu'un joueur reçoit aux premiers coups deux cartes de même valeur, il peut les séparer et jouer sur deux mains différentes (jusqu'à trois maximums sur deux tirages) en rajoutant la même mise sur la deuxième carte. Il peut recevoir autant de cartes qu'il le désire, sauf s'il a séparé une paire d'As (une seule carte peut compléter l'As). Si la carte qui s'ajoute à l'As compte 10 points, le joueur ne fait pas Black Jack et n'est payé qu'à égalité de sa mise. Il en est de même si un As complète une carte de 10 points.