

UE Programmation Orientée Objet

TD Bataille

Exercice 1 : Bataille!

(Les classes/interfaces de cet exercice seront à ranger dans un paquetage `bataille`.)

Il s'agit de coder le déroulement (automatique) de parties du jeu de bataille.

Rappelons en brièvement les règles (au cas où...) : le jeu se joue à 2 joueurs qui disposent chacun d'un tas de cartes. Ce tas est une file (FIFO). À chaque tour de jeu, chacun des joueurs prend la première carte de son tas (début de sa file) et la pose sur la table. Si les valeurs des 2 cartes sont différentes (on ne tient pas compte des couleurs dans ce jeu), le joueur qui a mis la carte de plus forte valeur¹ ramasse (toutes) les cartes qui sont sur la table et les met (peu importe dans quel ordre) sous son tas (à la fin de sa file). Si les deux cartes sont de même valeur, il y a "bataille", dans ce cas, chacun des joueurs pose la première carte de son tas sur la table. On recommence alors un nouveau tour de jeu en laissant les cartes sur la table et en appliquant à nouveau les règles précédentes. Le prochain qui gagne un tour de jeu récupère toutes les cartes qui ont été posées sur la table (et les place sous son tas). Dès qu'un joueur ne peut plus jouer parce qu'il n'a plus de carte dans son tas, il a perdu.

Q 1 . Les cartes :

Q 1.1. Définissez le type énuméré `Couleur` permettant de gérer les quatre couleurs : *cœur*, *carreau*, *trèfle* et *pique*.

Q 1.2. Définissez de manière similaire le type énuméré `Valeur` gérant les valeurs des cartes (*sept*, *huit*, ... *as*).

Q 1.3. Définir la classe `Carte`, une carte est définie par une valeur et une couleur. Cette classe doit implémenter l'interface `java.lang.Comparable<T>`.

Q 2 . Définissez une classe `TasCartes` permettant de gérer les "tas" de cartes.

Ces "tas" de cartes se comportent comme une file "FIFO" (*first in first out*) de capacité limitée, fixée à 32. La classe fournira les méthodes :

- `public Carte enleveCarteEnTete() throws TasVideException`
- `public void ajouteCarteEnFin(Carte carte) throws TasPleinException`
- `public void ajouteDesCartesEnFin(TasCartes tas) throws TasPleinException` qui ajoute toutes les cartes de `tas` à la fin de "*this*" (du moins tant que celui-ci n'est pas plein)
- `public boolean estVide()`
- `public boolean estPlein()`

Les classes d'exceptions `TasPleinException` et `TasVideException` étant supposées définies.

Q 3 . Donnez l'algorithme puis le code d'une méthode `joue`, d'une classe `Bataille`, qui permet d'exécuter le déroulement d'une partie du jeu de bataille en supposant que les tas des deux joueurs ont été initialisés.

On pourra représenter la table sur laquelle les joueurs posent leur carte par un "tas de cartes".

¹L'ordre des valeurs des cartes de la plus forte à la moins forte est : As, Roi, Dame, Valet, 10, 9, 8, 7