TP1

Olivier Marchetti

octobre 2023

1 Le jeu de carte de la bataille

L'objectif de ce TP est de réaliser en Java le jeu de carte de la bataille. Le déroulement de la partie sera entièrement contrôlé par la machine.

Pour jouer à la bataille, on utilisera un jeu classique de 52 cartes :
quatre couleurs : pique, coeur, carreau et trèfle.
les cartes portent les noms : As, Deux, Trois,..., Dix, Valet, Dame, Roi. La valeur de la carte est déterminée par son nom. Si c'est une tête, la valeur est déterminée par sa hiérarchie. On considère que l'As est la carte de plus forte valeur.

Dans le jeu de la bataille, après avoir mélangé le jeu de carte, on distribue 26 cartes aux deux joueurs. Chaque joueur dispose alors d'un paquet retourné devant lui. Ensuite, chaque joueur retourne la première carte du sommet de son paquet et la pose sur la table. Le joueur ayant posé la carte de plus forte valeur ramasse les cartes posées. Si les cartes posées par les deux joueurs ont la même valeur, il y a bataille. Chaque joueur retire alors la première carte de son paquet et la pose sur la carte précédemment posée puis il retire de nouveau la première carte de son paquet et la pose sur la carte précédemment posée. Le joueur ayant posé la carte de plus forte valeur remporte la bataille et ramasse toutes les cartes posées sur la table. Si les dernières cartes posées sont de même valeur, il y a de nouveau une bataille. Le jeu prend fin lorsque l'un des deux joueurs se retrouve sans carte.

Afin de respecter l'esprit de la POO, vous devrez dans un premier temps analyser le problème et proposer une modélisation pertinente du jeu (sous forme de classes et de méthodes). Une fois ce travail validé par l'encadrant, vous pourrez passer à la programmation en Java. Nous vous conseillons de prendre environ 45 minutes à 1 heure de votre début de TP pour réfléchir sur la modélisation du problème.

2 Modélisation objet

Lors de la conception de tout programme objet, il convient de passer un certain temps pour définir les classes d'objets et les relations entre ces classes. La démarche objet suggère au concepteur de se mettre dans la peau de chaque objet de voir le monde tel que l'objet le verrait. Il vous faut donc définir les classes d'objets qui sont mises en jeu dans ce TP, définir les relations que ces objets

vont avoir et déterminer quels seront les niveaux d'accessibilité des champs de ces classes.

Vous pourrez par exemple commencer par définir une classe `Carte` pour modéliser les cartes. Cette classe sera munie de trois attributs privés (le nom (pour stocker sous forme de *String* les noms As, Deux,..., Dix, Valet,..., Roi), la couleur (pour stocker sous forme de *String* la couleur pique, coeur, carreau ou trèfle de la carte) et une valeur symbolique entière pour définir simplement les règles de la bataille de façon algorithmique). La classe contiendra un constructeur qui construira en mémoire un objet de type Carte en fonction d'un entier n passé en paramètre. Cet entier représentera le rang de la carte dans un jeu complet trié (par exemple, un 7 passé au constructeur engendrera un sept de pique, un 15 engendrera un deux de coeur). Pour les attributs étant définis privés, on leur prévoira une méthode d'accès. On commentera également de façon judicieuse le code avec les commentaires *javadoc* de sorte à générer une documentation conviviale au format *html* (voir fin du sujet pour *javadoc*).

Cependant, il ne faut pas vous perdre dans les détails de l'implémentation de votre modélisation objet. Vous pourrez faire une description sous forme de diagramme UML. Les détails de l'implémentation seront réglés lors de la phase de codage en Java.

3 Programmation en Java

Comme pour tous les langages de programmation, l'écriture d'un programme se fait par touches successives permettant de tester la robustesse et de valider les portions de codes précédemment écrites. Ne cherchez pas à faire des opérations complexes dès le début. Allez-y progressivement, quitte à simplifier le problème avant de le résoudre dans sa généralité.

Afin de vous aider à developper, pensez à munir chacune de vos classes d'une méthode d'affichage permettant d'afficher l'état des objets implémentés en mémoire si nécessaire. Pour valider le bon fonctionnement de chaque classe, vous écrirez un méthode `main` testant les méthodes de votre classe.

Lors de ce TP, nous vous demandons de manipuler la classe `ArrayList` (liste doublement chaînée circulaire). Pour manipuler les instances de cette classe, consultez la documentation en ligne de *Java - Sun*.

4 Trichons un peu...

On souhaite à présent modéliser un tricheur. Un tricheur est un joueur qui dispose secrètement d'un paquet de quatre As qu'il utilise dès lors qu'il s'apprête à jouer une carte de valeur inférieure ou égale à quatre (*i.e.* un deux, un trois ou un quatre). Lorsque cette situation se produit, le tricheur fait disparaître de son jeu la carte de faible valeur et joue un as de son paquet secret. Pour cela, nous vous conseillons de modifier au minimum vos classes. On fera également en sorte que le tricheur n'utilise pas un de ses As lorsqu'il faut poser la première carte d'une phase de bataille.

Exemple d'utilisation de Javadoc

L'extrait de code Java suivant définit la classe `Carte` (constructeur à compléter). Remarquez les commentaires *javadoc* et l'emploi des *tags javadoc* :

@param + nomDuParametre, tag permettant de documenter les paramètres d'appel d'une méthode.

@return + votreTexte, tag permettant de documenter la valeur retournée par une méthode retournant une valeur.

@see [NomDeClasse]#NomDeChamp, tag permettant de faire un lien *html* dans la documentation vers un autre champ. Si le tag est précédé d'un nom de classe alors le lien est établi vers la page *html* documentant le champ de cette classe.

La première phrase d'un commentaire *javadoc* d'un champ doit être une description succincte de ce champ. Elle sera mise en exergue dans la documentation associée à ce champs. Seuls les champs publics feront l'objet d'une documentation *javadoc*.

Pour utiliser *javadoc*, vous créerez un répertoire `Doc` puis vous invoquerez la commande suivante `javadoc -d Doc votreFichier_1.java ... votreFichier_n.java`.

```
/**
 * La classe Carte permet de représenter en mémoire un objet carte.
 */
public class Carte {
    private int valeurSymbolique; // valeur numérique pour traiter la bataille
    private String couleur;  // Pique, Coeur, Carreau, Trèfle
    private String nom;  // deux, trois,..., Valet,..., As

    /**
     * Constructeur de la classe Carte. Prend un entier n en paramètre
     * et retourne une carte correctement initialisée.
     *
     * @param Le numéro n de la carte dans le jeu complet triée.
     */
    public Carte (int n) {
        ...
    }

    /**
     *  Retourne la valeur symbolique de la carte contenue dans l'attribut
     *  valeurSymbolique.
     *
     *  @return L'entier figurant dans le champs private valeurSymbolique.
     */
    public int valeurDeCarte () {
         return valeurSymbolique;
    }
    .
    .
    .
    /**
     *  Affiche une carte.
     *
     *  @see Carte#Carte()
     */
    public void afficheCarte () {
        System.out.println (nom + " de " + couleur +
        " et de valeur symbolique " + valeurSymbolique);
    }
}
```