POO – Programmation Orientée Objet en Java

Fiche de TP numéro 3

Exercice 1 : Cartes à jouer

On s'intéresse à un jeu de 52 cartes avec quatre couleurs (coeur, carreau, trèfle, pique), et 13 hauteurs de rangs (de 2 à 9, Valet, Dame, Rois, As).

1. Écrivez une énumération pour les couleurs qu'on appellera `Couleur`. On associera un symbole pour chaque couleur grâce à sa valeur unicode. Le symbole sera renvoyé par la fonction `toString` de l'énumération. Voici la liste des caractères unicode :
   — ♠ :"\u2660"
   — ♥ :"\u2665"
   — ♦ :"\u2666"
   — ♣ :"\u2663"
2. Écrivez une énumération pour les hauteurs. Au rang est associé un nombre de points : de 2 à 10 points pour les rangs de 2 à 10, 11 points pour un valet, 12 points pour une dame, 13 pour un roi, et 14 pour un As. Vous implémenterez les méthodes :
   — `public int getRang()`
   — `public String toString()`
3. Écrivez une classe `Carte` avec deux attributs non modifiables, rang et couleur, un constructeur, des accesseurs et une méthode `toString()`. Cette fonction doit renvoyer un format "hauteur symbole". Par exemple, l'as de pique sera représenté par "As♠", la dame de coeur "D♥" et le 5 de carreau "5♦".
4. Écrivez une classe `Jeu` qui contient un jeu de cartes. Le jeu devra donc contenir les 52 cartes. Déclarez son constructeur et une méthode `afficheJeu()` (qui affiche sur la console chaque carte du jeu).
5. Écrivez une classe `DemoJeu` qui instancie un jeu et affiche chaque carte du jeu.

Exercice 2 : Pile et File

Vous allez écrire deux classes, `FileDeCartes` et `PileDeCartes`. Le principe d'une file et d'une pile est rappelé ci-dessous.

Les deux classes doivent contenir les méthodes `ajouterCarte()` et `retirerCarte()` qui permettent d'ajouter et de retirer une carte. La capacité maximale des files et piles sera fixée à 52 par défaut étant donné qu'ils ne peuvent pas contenir plus de 52 cartes avec un seul jeu. L'emplacement où ajouter et retirer la carte doit correspondre au fonctionnement d'une file ou d'une pile (détaillé ci-dessous). La classe pile est plus simple à implémenter que la classe file. Pour faciliter l'implémentation de la file, vous pouvez utiliser un buffer circulaire.

Comme dans une file d'attente, on retire le premier élément a avoir été ajouté

Exemple : on ajoute deux éléments (C et D) et on en retire un (A)

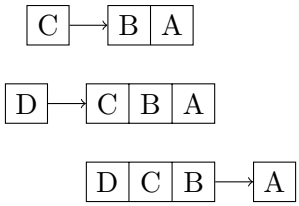

FIGURE 1 – Fonctionnement d'une file

Comme dans une pile d'assiette, on retire le dernier élément qui a été ajouté

Exemple : on ajoute un élément (C) et on en retire deux (C et B)

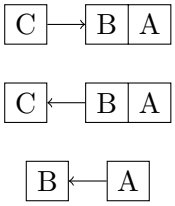

FIGURE 2 – Fonctionnement d'une pile

— Écrivez la classe `FileDeCartes`
— Écrivez la classe `PileDeCartes`

Exercice 3 : Bataille

Écrivez une classe `Bataille` qui permet de jouer à la bataille avec un jeu de cartes. Dans ce jeu, deux joueurs ont chacun un tas de cartes face cachée. Pour chaque tour de jeu, les deux joueurs retournent une carte de leurs paquets simultanément pour les dévoiler et le joueur qui possède la plus haute carte prend toutes les cartes retournées pour les mettre en dessous de son paquet. En cas d'égalité, personne ne prend les cartes et elles sont laissées retournées tant que personne ne les a récupérées. On joue le tour suivant en déposant les nouvelles cartes au-dessus de celles qui sont retournées. Toutes les cartes doivent être récupérées par un des joueurs pour gagner.

Vous utiliserez la classe `FileDeCartes` pour représenter les cartes qu'un joueur à en main et la classe `PileDeCartes` pour représenter les cartes retournées. Une fonction de mélange peut être ajoutée à la classe `Jeu` pour rendre les parties plus intéressantes. Les parties doivent se dérouler automatiquement avec deux méthodes de la classe `Bataille` : une méthode qui implémente le déroulement d'un tour de jeu et une méthode qui enchaîne les tours tant qu'il n'y a pas de joueur gagnant.