

# Quatorze

Prénom \_\_\_\_\_  
NOM \_\_\_\_\_

20

## Examen de laboratoire Java

( septembre 2006)

### Remarques :

- ↪ vous travaillerez dans un répertoire evaluations/sept
- ↪ vous pouvez utiliser **toutes les notes** que vous voulez ainsi que les programmes qui se trouvent dans votre répertoire,
- ↪ l'examen se fait sur linux1,
- ↪ vous disposez de 4 heures,
- ↪ vous travaillez sur machine et remettez votre programme au terme du temps imparti.
- ↪ soyez attentif au **style**, à la **clarté** et à l' **indentation** du code

**E**NONCÉ. Nous allons implémenter une partie de *Quatorze* qui est un jeu de cartes de type solitaire. *Quatorze* se joue avec un jeu de 52 cartes, les cartes sont disposées en un tableau de piles de cartes de 3 lignes par 4 colonnes. Les piles des 2 dernières lignes ont 4 cartes, et celles de la première en ont 5.

## But du jeu

Le but est de sortir toutes les cartes du tableau en faisant des paires de cartes dont la somme fait 14 (as = 1, valet = 11, dame = 12, roi = 13), soit as+roi ou 6+8, ... S'il n'est plus possible de faire une paire, la partie est perdue.

## 1 Classe Couleur

(1 pts)

Vous écrirez une classe `Couleur` représentant les couleurs, Pique, Coeur, Carreau et Trèfle. Cette classe est une énumération.

## 2 Classe Carte

(2 pts)

Une carte possède trois attributs :

↪ couleur de type `Couleur`,

↪ valeur de type `int`, au sens AS=1, deux=2, ... valet=11, ...

↪ nom de type `String`, son nom, AS, Valet, ...

et les méthodes ; constructeur, assesseurs (uniquement les *getters*, les cartes ne changent pas en cours de jeu), `toString`.

## 3 Classe JeuDeCartes

(3 pts)

Un jeu de 52 cartes. L'attribut est laissé au choix, il représente l'ensemble des cartes ... à vous de voir.

Les méthodes, constructeur (construit un jeu de 52 cartes et le mélange) et une méthode 'donneCarte' qui donne une carte (et la supprime du jeu).

## Javadoc

(2 pts)

Pour cette classe, vous écrirez la **javadoc**.

## 4 Classe Tas

(3 pts)

Un tas représente un tas de 4 ou 5 cartes, le plateau de jeu comportera 12 tas (4\*3). Un tas possède un attribut de type `Stack` ou `ArrayList` (au choix) et propose les méthodes

↪ constructeur d'un tas vide,

↪ `void push(Carte c)` , place une carte sur le tas

↪ `Carte pop()` , retire la carte au sommet du tas,

↪ `Carte peek()` , jette un oeil au sommet du tas (retourne la carte sans la retirer du tas ou lance une exception si le tas est vide)

↪ `boolean empty()` , dit si le tas est vide

## 5 Classe Plateau

(3 pts)

Cette classe représente le plateau de jeu, celui-ci est composé d'un tableau 3\*4 Tas.

Les méthodes,

- ↪ constructeur crée un nouveau plateau et remplit les tas de 4 ou 5 cartes (cfr règle ci-dessus),
- ↪ affiche, permet d'afficher le plateau de jeu. Dans chaque "case", on affichera la Carte qui se trouve au sommet du tas (pour ce faire vous pouvez utiliser les méthodes toString que vous auriez écrites).
- ↪ boolean estVide(), retourne vrai si le plateau est vide ... ce qui signifie que la partie de solitaire est terminée et gagnée ... sinon le joueur aura abandonné,
- ↪ Tas getTas(ligne, colonne) pour les tas

## 6 Classe Quatorze

(3 pts)

Mise en place du jeu proprement dit. Une structure répétitive du type suivant devrait suffire

```
tant que (plateau non vide)
    regarder deux cartes au hasard
    si (les deux cartes ont pour somme 14)
        les supprimer des tas respectifs
fin tant que
```

## 7 Test unitaires

(3 pts)

Ecrivez des tests pour la classe Tas. Vous testerez les méthodes

- ↪ Tas,
- ↪ push,
- ↪ pop,
- ↪ peek et
- ↪ empty