TP no. 2 sur les collections « Jeu de cartes»

(*equals, hashcode, Set, itérateurs, tri*)

S. Rosmorduc

September 16, 2022

Exercice 1: jeu de cartes

Dans cet exercice nous allons implanter les classes nécessaires à un jeu de cartes. Dans le site du cours, récuperez les sources pour ce Tp. Vous trouverez 3 classes: `Carte`, `TestCarte0` et un type *énuméré* donné par la classe `Couleur`.

Question 1: type énuméré Couleur

Un type énuméré est formé d'un ensemble d'identifiants considerés comme des constantes. Dans le type `enum Couleur` chacune des constantes correspond à une couleur de carte: *Pique, Carreau, Coeur, Trefle*. Ces constantes sont de type `Couleur` et peuvent donc être mises dans des variables de ce type, passées en paramètre, etc.

```
package cartes;
public enum Couleur {
    Pique,
    Carreau,
    Coeur,
    Trefle;
}
```

Le classes enum ont des caractéristiques intéressantes qui facilitent l'écriture de programmes. Toute classe enum E, possède une méthode statique implicite `static E [] values()` qui retourne un tableau composé des constantes déclarées dans l'enum. Par ailleurs, ces classes implantent `Comparable`. Il est donc possible de comparer deux constantes enum via `couleur1.compareTo(couleur2)`.

Dans la classe `TestCarte0` il y une boucle qui affiche toutes les constantes du type Couleur: expliquez son code. Déclarez ensuite un ArrayList de type Couleur et ajoutez y quelques constantes. Ecrivez une boucle qui trouve et affiche la plus petite des constantes de la liste au sens de l'ordre de leur déclaration dans l'enum.

Question 2 : cartes

Considérez la classe `Carte` fournie. Notez comment le type couleur est utilisé dans la classe `Carte`.

```
package cartes;
public class Carte {

        private int valeur;
        private Couleur couleur;

        public Carte(int valeur, Couleur c) {
                this.valeur = valeur;
```

```
                this.couleur = c;
        }
        public int getValeur() {
                return valeur;
        }
        public Couleur getCouleur() {
                return couleur;
        }
        @Override
        public String toString() {
                return "" + valeur + "␣de␣" + couleur;
        }
}
```

Testez cette classe avec le programme `TestCarte0` fourni sur le site et expliquez les affichages obtenus.

```
package cartes;
import java.util.ArrayList;

 // Expliquez le comportement de ce programme

public class TestCartes0 {

        public static void main(String[] args) {

                // A quoi correspond l'appel Couleur.values() ?
                for (Couleur c: Couleur.values()) {
                        System.out.print(c + "␣");
                }
                // Question 1: creez une liste avec quelques couleurs puis
                // via un boucle trouvez et affichez la plus petite dans l'ordre
                // de l'enum.
                ArrayList<Couleur> cols; // a completer
                ArrayList<Carte> cartes = new ArrayList<Carte>();

                cartes.add(new Carte(10, Couleur.Carreau));
                cartes.add(new Carte(1, Couleur.Coeur));

                if (cartes.contains(new Carte(10, Couleur.Carreau))) {
                        System.out.println("Le␣jeu␣contient␣le␣10␣de␣carreau");
                } else {
                        System.out.println("Le␣jeu␣ne␣contient␣pas␣le␣10␣de␣carreau");
                }
        }
}
```

Question 3 : `hashCode()` et `equals()`

Ajoutez des méthodes `hashCode()` et `equals()` correctes dans la classe `Carte`. Testez le bon fonctionnement de ces deux méthodes:

1. Pour vérifier que ces deux méthodes fonctionnent bien, remplacez l'`ArrayList` par un `HashSet` en y ajoutant des doublons. Ajoutez l'affichage du nombre d'éléments dans la collection.

2. recommencez l'exécution du programme `TestCarte0`.

Question 4 : MainJoueur1

On veut représenter la *main* d'un joueur contenant plusieurs cartes (pas la méthode main!). Pour cela, on écrira la classe `MainJoueur1`, où vous utiliserez un `HashSet` pour stocker les cartes de la main. Dans un premier temps on souhaite doter cette classe des méthodes suivantes:

- `add(Carte)`: ajoute une carte à la main.
- `contient(Carte)`: renvoie true si la main contient une carte.
- `toString()`: renvoie une représentation de la main sous forme de chaîne.

Vérifiez le bon fonctionnement de votre classe avec un petit programme qui créé une main, y ajoute plusieurs cartes et affiche le résultat.

Question 5 : paquet de cartes

On représente un paquet de cartes par la classe `Paquet`. Quand on crée un paquet, il est complet (cartes de valeurs 1 à 13 pour chaque couleur) et à priori mélangé (les cartes sont dans un ordre aléatoire). On dote la classe `Paquet` d'une méthode `toString` et d'une méthode `size()`. Une méthode `creerMainDeJoueur1` permettra de tirer 5 cartes du paquet (les 5 premières, par exemple), pour constituer une main, puis de retourner cette main. Sa signature est `public MainDeJoueur1 creerMain1()`. Notez que l'on doit retirer du paquet les cartes tirées et mises dans le résultat de cette méthode. Pour tester votr classe, écrivez un programme qui crée un paquet, affcihe le paquet, crée une main à partir de ce paquet, puis la visualise.

Question 6 : intégration

Testez vos classes avec le programme suivant:

```
package cartes;
import java.util.Scanner;
public class TestPaquet {
   public static void main(String[] args) {
        // Creation d'un paquet
        Paquet paquet = new Paquet();
        System.out.println("taille␣du␣paquet:␣"+ paquet.size());

        // Creation d'une main1
        MainJoueur1 maMain = paquet.creerMain1();
        System.out.println("mon␣jeu:␣");
        System.out.println(maMain);

        // Les cartes restant dans le paquet
        System.out.println("Reste␣dans␣le␣paquet␣␣"+paquet.size()+ "␣cartes:");
        System.out.println(paquet);

        Scanner scanner = new Scanner(System.in);
        System.out.print("Entrez␣une␣valeur␣de␣carte␣:");
        int valeur = scanner.nextInt();
        System.out.print("Entrez␣une␣couleur␣de␣carte␣:");
        String nomcouleur = scanner.next();

        Carte c = new Carte(valeur, Couleur.valueOf(nomcouleur));
        if (maMain.contient(c)) {
                System.out.println("La␣main␣contient␣la␣carte");
        } else
```

```
                System.out.println("La main ne contient pas la carte");
  }
}
```

Question 7 : itération sur MainJoueur1

On veut disposer d'un moyen simple de parcourir les cartes d'une `MainJouer1`. Pour cela, modifiez cette classe de sorte qu'elle implante l'interface `Iterable<Carte>`. Testez le résultat en avec le programma suivant :

```
public static void main(String[] args) {
        MainJoueur1 mainj1 = new MainJoueur1();

        mainj1.add(new Carte(10, Couleur.Carreau));
        mainj1.add(new Carte(1, Couleur.Coeur));
        mainj1.add(new Carte(10, Couleur.Trefle));

        // Test d'iteration sur une MainJoeur1
        for(Carte c: mainj1){
                System.out.println(c.toString());
        }
}
```

Question 8 : MainJoueur2

Pour cette question vous allez créer une nouvelle classe `MainJoueur2` avec les mêmes fonctionnalités que `MainJoueur1`, mais qui stocke les cartes de la main dans un `TreeSet` de Cartes. Un TreeSet est un ensemble ordonné. L'ordre des cartes sera testé selon leur valeur puis selon leur coleur. Comme pour la question précédente, votre classe doit implanter l'interface `Iterable<Carte>`. Vous testerez votre classe avec le programme donné dans la question précédente (n'oubliez pas de modifier le type de la variable `mainj1` en `MainJoueur2`).

Question 9 : nombre de couleurs

On veut implémenter la méthode `getNombreCouleurs` (qui renvoie le nombre de couleurs qui apparaissent dans une main). Pour cela, on désire travailler sur un jeu trié par couleur. En utilisant un Comparator, et en travaillant sur un TreeSet temporaire, écrire la methode getNombreCouleurs dans la classe MainDeJoueur.

Question 10 : tester les suites

Dans `MainJoueur2` ajoutez la méthode `estSuite()` qui teste si une main est une suite de cartes. Une suite de cartes est une main triée composée de cartes de même couleur avec valeurs consécutives.