NFA035 – Bibliothèques et patterns : TP n°7 (3ème sur les collections)

S. Rosmorduc, V. Aponte

1er avril 2016

Exercice 1 : suite du livre des recettes

Dans cet exercice on fera évoluer votre solution pour l'exercice 2 du Tp 5. Si ce n'est pas fait :

1. terminez l'exercice 2 du Tp5 (livre de recettes).
2. Testez votre code à l'aide des vos tests unitaires ou de ceux de la correction du tp5 (site du cours).

Vous pouvez également utiliser le code et tests de la correction. Vous devez vous assurer que votre code (ou celui de la correction) passe bien touts les tests ! Ne sautez pas cette étape.

Question 1 : le livre de recettes

On veut s'assurer qu'un livre de recettes ne contienne pas de recette en double (même nom de recette). Vous utiliserez une structure `Map` pour garder sans doublons les titres (String) des recettes. Das votre Map les clés seront les titres des recettes, chacune associée à un objet Recette. Attention : la logique de l'application change avec cet implantation. Lors du Tp5 on identifiait une recette (p.e., pour la noter) en donnant son rang dans le livre. Cela n'a pas de sens dans une Map (pourquoi ?). Quelles sont les opérations qui changent et comment ? Modifiez votre classe de manière à faire fonctionner votre application avec cette nouvelle implantation.

Par ailleurs, on voudrait que les recettes du livre soient classés en ordre alphabétique. A cette fin, les clés de recherche de votre Map doivent implanter Comparable. Mais cela ne suffit pas : vous devez utiliser un TreeMap pour stocker les recettes du livre. A ce stade, *utilisez à nouveau* les tests mis en place lors du tp précédent afin de vous assurer que tout fonctionne comme prévu : ce sont des *tests de non regréssion !*

1. Votre programme main devrait bien fonctionner avec peu de modifications (lesquelles ?)
2. Vos test Junit devraient réussir avec peu de modifications.

Question 2 (s'il vous reste du temps)

On souhaite enrichir la définition d'une recette avec la liste de ses ingrédients (sans doublons) et la quantité nécessaire pour chaque ingrédient. Un ingrédient sera représenté par un String et une quantité par un nombre d'unités (Integer). Comment représenter les ingrédients et quantités d'une recette ? Modifiez la classe Recette afin d'y stocker les ingrédients et quantités et ajoutez les méthodes appropriées dnas cette classe et dans LivreRecettes. Ajoutez des tests et testez.

Exercice 2 : suite jeu de cartes

Vous reprendrez le code de l'exercice « jeu de cartes » du tp précédent, là où vous l'avez laissé. Terminez toutes les questions demandées alors avant d'entâmer cette suite.

Question 6 : 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 mélangé. 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 réusltat de cette méthode.

Question 7 : intégration

Testez vos classes avec le programme suivant :

```
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 couleur = scanner.next();

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

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.

Question 9 : tester les suites

Dans `MainJoueur2` ajoutez la méthode `estSuite()` qui teste si une main est une suite de cartes.

Question 10 (optionnelle) : nombre de couleurs d'une main

On veut implémenter `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 cette méthode dans une des classes MainJoueur.