Travaux Dirigés n°9

Java Avancé

—M1 Apprentissage—

Enum, Set

Enum, Enum abstraite, Set

▸ Exercice 1. Calculette

1. Créer une énumération constituée des 4 opérations courantes (addition, soustraction, multiplication, division).
2. On souhaite pouvoir récupérer le symbole associé à l'opération. Pour cela, utiliser un constructeur privé associé à l'énumération.
3. On souhaite pouvoir afficher ce symbole. Par exemple, `System.out.println(Operation.PLUS);` affichera +. Indice : une énumération est aussi un objet.
4. On souhaite maintenant effectuer les calculs liés à ces opérations. Écrire une méthode `compute` prenant deux doubles et retournant le résultat de l'opération. Réfléchir à l'intérêt d'une énumération abstraite ici plutôt qu'une méthode utilisant un switch.
5. Tester avec ces tests JUnit : OperationTest

▸ Exercice 2. Swag

On veut définir une méthode `String swag(String txt, int style);` dans une classe `Swag` qui renvoie le texte txt avec un certain style passé également en argument (on n'utilise pas d'énumération pour l'instant). L'entier style indique l'effet à appliquer. Par exemple, pour le texte "miage" :

— CROSS affiche m+i+a+g+e
— KIKOO affiche MiAgE
— CROSS|KIKOO affiche M+i+A+g+E

1. Fixer CROSS et KIKOO comme des constantes entières et écrire la méthode swag. Des combinaisons sont possibles, par exemple `swag(‘‘miage’’,CROSS|KIKOO)` doit fonctionner.
2. Tester avec SwagTest1

3. Utiliser une énumération (qu'on appelle `STYLE`) plutôt que des constantes. Pour passer plusieurs éléments, vous pouvez utiliser un `Set`. Modifier votre méthode swag pour qu'elle prenne un `Set` en second argument.
4. Faire appel à votre méthode dans un main. Pour créer facilement un Set, utiliser `EnumSet.of()`.
5. Modifier votre énumération pour qu'elle contienne une méthode abstraite `applyStyle` appliquant la transformation associée. Modifier votre méthode swag pour qu'elle l'utilise.
6. Si vous avez placé CROSS avant KIKOO dans votre énumération, que remarquez-vous sur ce qui est renvoyé si les 2 styles sont demandés ? Pour remédier à cela, utilisez un Set qui préserve l'ordre d'insertion, comme `LinkedHashSet`.
7. Tester avec ces tests JUnit : SwagTest2

▸ Exercice 3. Sac

Une structure données stockant plusieurs fois le même élément (au sens du equals) et le nombre de fois où cet élément apparait est appelé Bag ou MultiSet. Elle n'existe pas dans la JDK, mais est souvent utile.

1. Écrire une classe (paramétrée) pour une telle structure de données, avec les méthodes d'ajout, suppression (d'une occurrence) et de comptage (nombre d'occurrences d'un objet). Ces opérations doivent s'effectuer avec la meilleure complexité possible.
2. Ajouter une méthode `Iterator<Map.Entry<T,Integer>> iterator()` qui renvoie un itérateur sur les couples stockés dans votre structure. La solution s'effectue en une seule ligne !
3. Ajouter une énumération permettant de choisir l'ordre pour l'itération (au moment de la construction du Bag (vous garderez un constructeur sans argument qui utilise aucun ordre spécifique). Les choix possibles doivent être : ordre d'insertion, ordre naturel (au sens compareTo) et aucun ordre spécifique. Est-ce que cela impose des contraintes sur le type d'objets stockés ? (Normalement, il aurait fallu faire une interface commune avec différentes classes concrètes selon le type, mais le but est de vous faire utiliser des énumérations !).
4. Utiliser une méthode abstraite dans votre énumération pour simplifier le code du constructeur.
5. Tester avec ces tests JUnit : BagTest1
6. On veut pouvoir exécuter le code suivant :

```
Bag<String> bag=new Bag<>(BagOrder.ANY);
bag.add("toto");
bag.add("toto");
bag.add("titi");
for(Map.Entry<String,Integer> entry:bag)
   System.out.println(entry.getKey()+'='+entry.getValue());
```

. Que faut-il faire pour que cela fonctionne ?

7. Réfléchir à quelle interface de Collection votre implémentation pourrait coller. Vous avez le droit d'utiliser `AbstractCollection`. Est-ce que cela pose un problème pour l'itérateur ? Tester avec le code suivant :

```
Collection<String> bag = new Bag<>(BagOrder.INSERT);
bag.add("denis");
bag.add("cornaz");
bag.add("denis");
for(String s:bag) {
  System.out.println(s);
  //affiche denis denis cornaz
}
```

8. Tester avec ces tests JUnit : BagTest2