Sokoban : étape 4

Nous allons maintenant rendre génériques nos séquences créées précédemment. Le problème est qu'elles ne manipulent que des entiers: nous aimerions disposer de séquences nous permettant de manipuler toutes sortes d'objets. En particulier, par la suite, nous allons utiliser des séquences de coups (pour gérer l'historique), d'observateurs (pour le *pattern* éponyme), d'animations (pour gérer les éléments en mouvement dans la scène) et de marques (pour visualiser le travail de l'IA).

1 - Des séquences génériques

Comme vu en cours, en java, toute classe ou interface peut être paramétrée par un type de référence inconnu lors de sa compilation. Cela se fait en ajoutant un nom générique, par exemple `T`, entre < et > après le nom de l'interface ou de la classe. Le reste de l'interface ou de la classe peut alors utiliser `T` pour déclarer des variables contenant des références à des objets de ce type ou réaliser des affectations entres références de ce type. En revanche, il n'est pas possible de créer un objet de la classe `T`, qui n'est pas une classe mais un type générique. Il n'est pas non plus possible d'appeler n'importe quelle méthode à partir d'une référence à un objet de type `T` car, pour garantir la correction de l'ensemble, le compilateur considère que, au pire, les objets référencés par `T` sont de la classe `Object`. Cette classe contient, entre autres, les méthodes `toString`, `equals` ou `clone`. Comme nos séquences ne manipulent que des références avec, éventuellement, des appels à `toString` lors de l'affichage, nous nous contenterons de cela pour l'instant et verrons un peu plus loin comment repousser cette limite.

Question :

- modifiez vos séquences pour en faire des versions génériques. Pour la cohérence de l'ensemble il vous faudra aussi rendre vos itérateurs génériques et adapter vos programmes de test en conséquence. Faites cela pour les deux implémentations de séquences réalisées jusqu'à maintenant.

2 - Des FAP génériques

Un peu plus tard, vous aurez besoin de files d'attente à priorité (FAP) pour implémenter l'algorithme de Djikstra, que vous verrez en cours d'algorithmique. Cet algorithme servira de base à la résolution automatique du Sokoban et sera donc indispensable pour la partie IA. Vous connaissez déjà les FAP, mais il s'agit ici d'en faire une version générique.

Questions :

- dans un premier temps votre FAP va manipuler de simples valeurs pour lesquelles la priorité est égale à la valeur elle-même. Cela veut dire qu'il suffira de la paramétrer avec un seul type générique et cela implique que ce type, utilisé pour les valeurs et les priorités, est totalement ordonné, afin de pouvoir comparer des priorités entre elles. Ceci se fait, comme vu en cours, en imposant une contrainte au type générique. Ici il faut qu'il implémente l'interface `Comparable` de la bibliothèque standard, contenant une méthode `compareTo` qui vous permet de comparer les éléments entre eux.

  Vous pouvez réutiliser les séquences pour stocker les éléments, en les utilisant plutôt par compostion (c'est à dire que la FAP contient une séquence) que par héritage (car les opérations sur la FAP et la séquence ne sont pas les mêmes). Quant au détail d'implémentation, vous avez le choix :

  - maintenir votre séquence triée, en insérant toujours à la bonne position ;
  - ou bien extraire d'une séquence non triée en recherchant l'élément maximal ;
  - ou encore stocker les éléments dans un tas, mais nous éviterons cette solution qui permettra difficilement de réutiliser nos séquences déjà écrites.

  Pensez, bien entendu, à tester vos implémentations.

- dans un second temps, pour pouvoir utiliser des couples (valeur, priorité) dans votre FAP, il vous suffit de créer une classe nommée `Couple` dont les éléments sont totalement ordonnés selon la valeur de leur priorité.

3 - Un peu de rangement

Avant de passer à la prochaine étape, il est maintenant temps de ranger toutes nos classes et interfaces (qui commencent à être nombreuses).

Question :

- créez un paquetage nommé `Structures` et rangez-y toutes nos classes gérant des séquences et des FAPs.