TP Tours de Hanoi

Il s'agit d'implémenter le problème des tours de Hanoï étudié en TD.

Rappel méthodologique

Il faut travailler une méthode à la fois en appliquant la démarche suivante :

1. **écrire la signature de la méthode,**
2. **écrire la documentation (javadoc) de la méthode,**
3. **écrire les tests qui permettront de vérifier que le code produit pour la méthode est correct,**
4. **écrire le code,**
5. **exécuter les tests prévus à l'étape 3, en vérifiant que les tests des méthodes précédemment écrites (et testées) restent réussis**[a]**,**
6. **si les tests sont réussis passer à la méthode suivante (étape 1) sinon recommencer à l'étape 4.**

[a] On s'assure que le nouveau code écrit ne remet pas en cause les codes précédents.

Packages. Pour vous familiariser avec l'utilisation des paquetages, il vous est demandé de définir un paquetage `hanoi` qui contiendra la classe qui représente le problème des tours de Hanoï (par exemple `Hanoi`) et un paquetage `hanoi.util` qui regroupera les classes modélisant les disques et les tours (par exemple `Disc` et `Tower`).

Travail à réaliser

Q 1 . Codez les 3 classes nécessaires : `hanoi.Hanoi`, `hanoi.util.Disc` et `hanoi.util.Tower`, en écrivant la documentation et exécutant les tests au fur et à mesure.

La classe `hanoi.Hanoi` devra posséder une méthode permettant la résolution complète du problème de Hanoï.

Q 2 . . Ajoutez dans la classe `hanoi.Hanoi` une (ou des) méthode(s) nécessaire(s) pour permettre une manipulation interactive des tours de Hanoï, en respectant bien sûr les contraintes du problème (pas de grand disque sur un plus petit).

Pour vous permettre de réaliser ce travail, une classe permettant la saisie d'information au clavier vous est fournie (cf. `fichiers-hanoi.tar.gz` sur le portail à décompresser dans votre répertoire de travail). Il s'agit de la classe `io.HanoiInput` (voir ci-dessous).

Pour avoir un aperçu du travail demandé et de l'utilisation de cette classe, vous pouvez tester l'archive hanoi.jar proposée pour démonstration :

```
java -jar hanoi.jar
```

Dans votre travail vous pouvez commencer par une version avec un affichage basique des tours de hanoi (par exemple une simple énumération des tailles des disques sur chaque tour), puis dans un second temps réalisez un affichage plus visuel comme celui proposé dans le `.jar` ci-dessus.

Dans la réalisation de ce second affichage, vous chercherez à bien décomposer le problème pour en faciliter la réalisation...

Q 3 . Créez une archive exécutable avec votre programme. Le programme exécuté par votre archive correspondra au mode de saisie interactive. Il prendra en paramètre le nombre de disques. Comme présenté dans le TP 4, cette archive devra contenir en plus les sources, les tests (pour `Disc` et `Tower`) et la documentation de votre projet. Cette archive sera à rendre via PROF.

Saisie interactive. `HanoiInput`
La saisie interactive utilisée dans cette archive utilise la classe io.HanoiInput qui vous est founie.
Lisez la documentation de cette classe (générée avec javadoc) fournie dans l'archive.
Comme vous pouvez le constater en consultant cette documentation, cette classe propose 4 méthodes :

- `readInput()` : cette méthode crée une attente d'une saisie valide de l'utilisateur (au clavier). Pour être valide une saisie doit être :
  - soit `"quit"`,
  - soit composée de 2 caractères pris parmi `g`, `c` ou `d` correspondant respectivement aux tours gauche, centre et droite. Une telle saisie représente un déplacement de disque : le premier caractère correspond à la tour de départ le second à la tour d'arrivée.

    La saisie `gc` aura donc pour effet de déplacer un disque de la tour de gauche vers celle du centre.
- les 3 autres méthodes (`isMove`, `getFrom`, `getTo`) permettent d'analyser et d'exploiter la saisie de l'utilisateur (voir la javadoc fournie).

Le principe est donc de :

- créer un objet `HanoiInput`,
- de réaliser la saisie en invoquant la méthode `readInput` sur cet objet
- d'analyser la réponse à l'aide des 3 méthodes supplémentaires fournies et si c'est un déplacement (`isMove`) d'identifier les tours de départ (`getFrom`) et d'arrivée (`getTo`) de ce déplacement et exécuter le déplacement correspondant.

Voir le petit code illustratif (que vous pouvez tester dans une méthode `main`) dans la javadoc fournie.