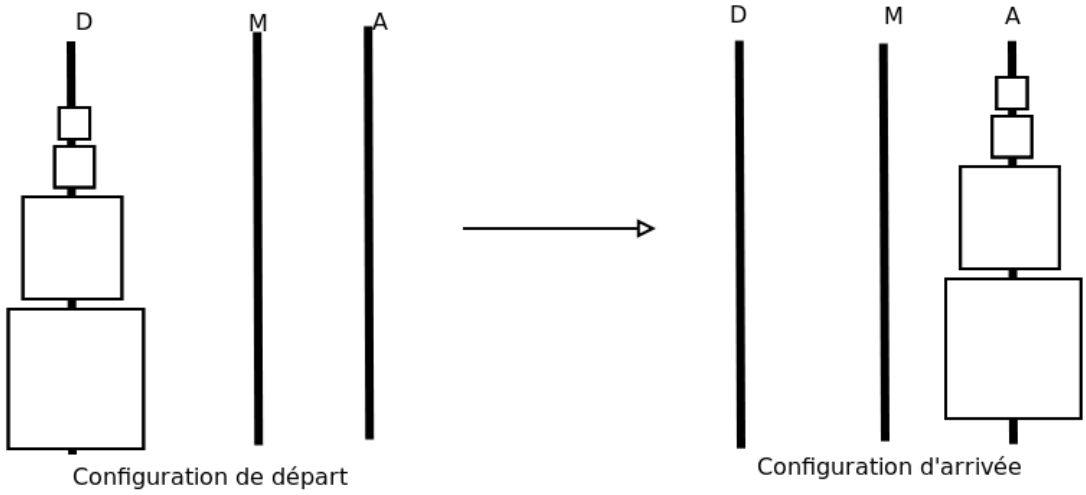
Les questions Q0-Q1, Q2 et Q3 doivent être faites dans des répertoires (ou projets sous Eclipse) différents, afin de pouvoir rendre les solutions séparées aux questions.

On veut créer différentes versions du célèbre problème des « Tours de Hanoi » en jouant sur les aspects génériques ou d'introspection de Java. On rappelle le problème général :



Règles de manipulation:
- On ne peut déplacer qu'un disque à la fois
- On ne peut pas poser un disque sur un disque plus "petit" pour une certaine notion de "taille" d'un "disque".

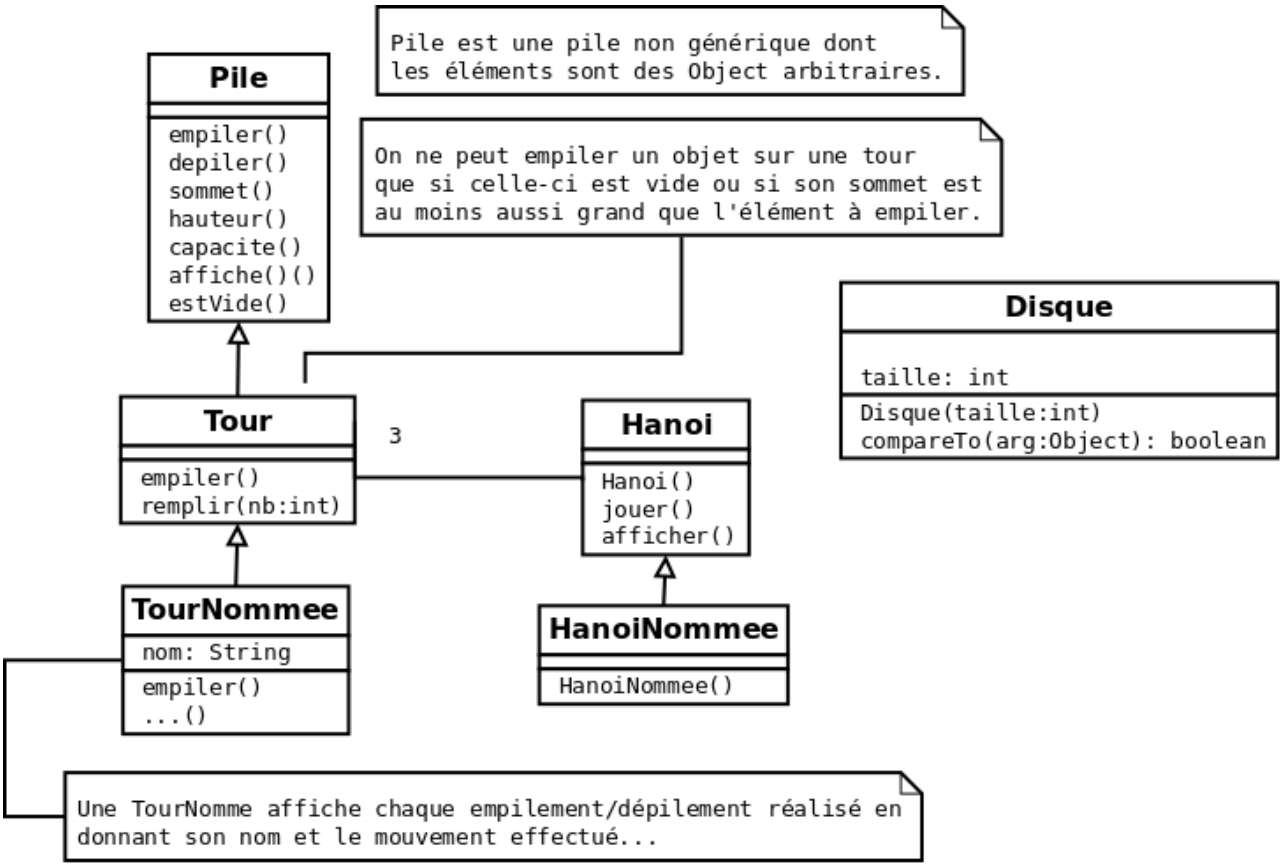
Une solution simple consiste à définir une fonction récursive basée sur le principe suivant, où `D`, `M`, `A` représentent respectivement les tours de départ, du milieu et d'arrivée. La procédure ci-dessous suppose les disques empilés sur la tour de départ et se contente de lancer une procédure récursive de déplacement des disques.

```
void jouer() { oneStep(D.capacite(), D, A, M) }
```

```
// Empiler nb Disque de D vers A en se servant de M comme Tour auxiliaire
void oneStep(int nb, Tour D, Tour A, Tour M) {
  if (nb > 0) {
      /* faire passer nb-1 disques de D vers M en se servant de A.
       * A la fin les disques sont ranges sur M, donc A est vide
       * et D contient l'unique disque de taille maximale.
       */
    oneStep(nb-1, D, M, A);
    A.empiler(D.sommet()); /* deplacer le disque de D vers A */
    D.depiler();
    /* deplacer nb-1 disques de M vers A en se servant de D
     * A n'est pas vide mais contient le plus gros disque, sur
     * lequel on peut empiler n'importe quel autre disque.
     */
    oneStep(nb-1, M, A, D);
  }
}
```

On a défini une première version non générique (i.e. Java 1.4) dans laquelle les « disques » sont modélisés par une classe spécifique `Disque` qui définit un constructeur et une fonction de comparaison. La hiérarchie de classes à implémenter est la suivante (`Pile` représente une pile non

générique dont les éléments sont typés comme `Object`. La capacité de la pile est fixée à la création de l'instance t n'est pas modifiable) :



Remarque : dans cette question et dans la suivante, on utilise les versions **non génériques** des classes et interfaces de la librairie standard (`ArrayList`, `Comparable`, …). Il est donc normal qu'on ait des « warnings » à la compilation à ce sujet.

Q0 : Complétez les classes `Tour` et `ErreurTour` de façon à obtenir une implémentation correcte des Tours de Hanoi telles que décrites ci-dessus. Les classes `Hanoi` et `HanoiNommée` contiennent des méthode `main()` pour lancer l'exécution.

Q1 : Plutôt que de travailler avec une classe `Disque` spécifique, puisque la seule notion dont on a besoin est de pouvoir comparer des « tailles » de « disques », on prévoit une version plus générale dans laquelle la notion de « disque » est décrite par une interface «`Empilable`» qui permet d'une part de comparer (au sens d'une relation d'ordre) deux éléments, d'autre part définit une méthode `init(int taille)` pour fixer la taille d'un élément (remplacez alors dans **Disque** le constructeur présent par un constructeur sans argument ; on fixera la taille d'un disque par un appel à `init`). On suppose que dans une même tour on n'empilera jamais des instances d'implémentations différentes de `Empilable`: les tours auront toujours un contenu homogène.

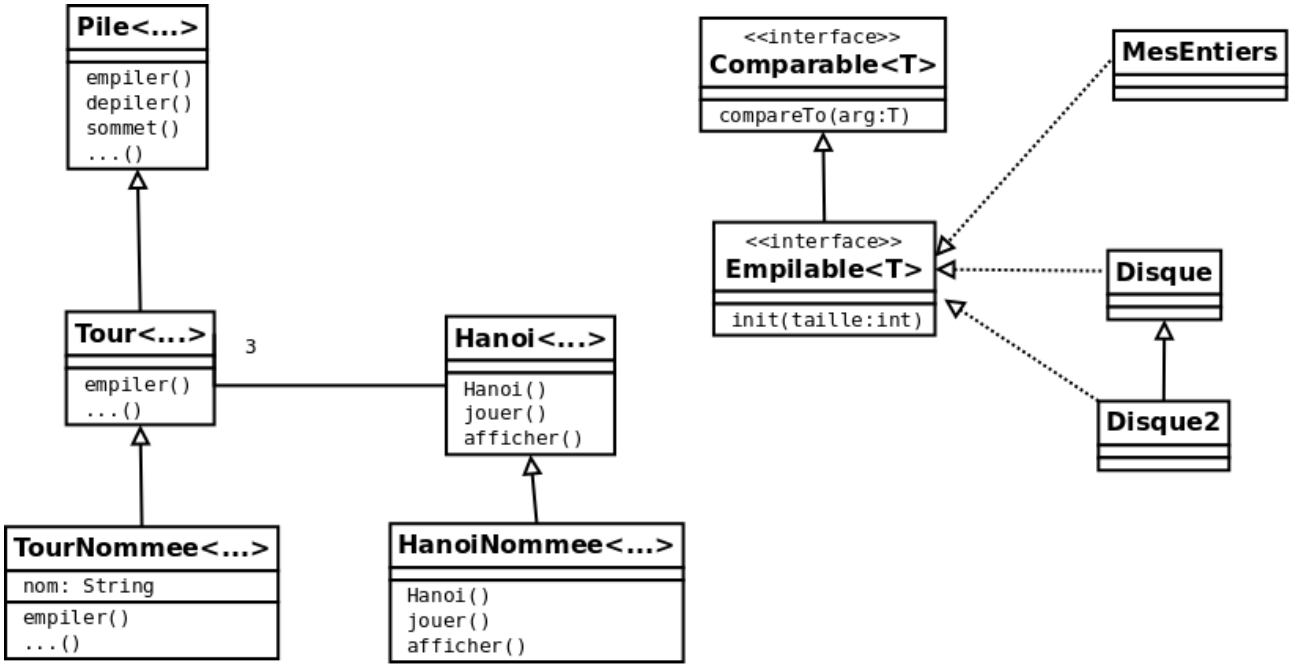
Modifiez la version précédente de manière à ce que les méthodes de `Tour` (**pas** de `Pile` qui reste une classe générale !) ne soient basées que sur cette notion d'objet empilable. Testez votre solution à l'aide d'un programme qui illustre le fonctionnement avec **deux classes d'objets qui implémentent l'interface** `Empilable`**.**

Q2 : Copiez votre solution dans le paquetage hanoi2 et modifiez ensuite modifiez-la de manière à ce que le nom de la classe d'objets à empiler ne soient plus écrite dans le programme de test, mais

demandée dynamiquement à l'utilisateur. On devra vérifier qu'une telle classe existe et a les bonne propriétés et procéder dynamiquement aux instanciations nécessaires.

Remarque : on utilisera les possibilités d'introspection de Java et notamment les méthodes de la classe `Class` et on gérera les exceptions associées. Testez votre programme notamment avec les différentes classes de disques mises à disposition : `Disque`, `MesEntiers`, `MesEntiers2` et `MesEntiers3`[1] ainsi qu'avec un nom de classe inexistante. **Avant d'exécuter vos tests, assurez-vous que les classes jouant le rôle de disque soient bien compilées** (i.e. le compilateur ne les compile pas automatiquement puisque leurs noms ne sont pas référencés statiquement dans le programme de test). Dans cette question on se préoccupe pas de des classes `TourNommee` et `HanoiNommee`. Le programme de test se trouve dans `TestHanoi`.

Q3. Plutôt que de jouer avec les aspects « introspection » on considère maintenant une version générique (i.e. Java 1.5+) des Tours de Hanoi, en définissant des classes `Pile` et `Tour` génériques. La hiérarchie de classes générale est la suivante :



Implémentez les différentes classes demandées de manière à obtenir une **solution générique et souple d'utilisation en se servant correctement des mécanismes de généricité**. Il vous revient notamment de définir les méthodes nécessaires, avec leurs paramètres, de fixer les paramètres de généricité des classes et de prévoir les instantiations. Votre solution doit permettre d'utiliser les Tours de Hanoi avec les trois classes qui apparaissent sur la droite du diagramme. La classe `Disque2` dérivant de `Disque` on veut pouvoir empiler des instances des deux classes dans une même tour ! On suppose que les 3 classes ont un constructeur vide et une méthode `init` qui permet de définir la « taille » d'une instance (pour les comparaisons à faire avant les empilements). Les classes d'exceptions nécessaires n'apparaissent pas sur le diagramme mais doivent être prévues.

Dans cette question, la compilation des différentes classes ne doit émettre aucun « warnings » et vous ne devez donc pas utilisez les versions non génériques (par exemple `ArrayList`) au lieu de la version générique (`ArrayList<QuelqueChose>`) tant pour les classes de la librairie que pour vos propres classes. Le programme de test se trouve dans `TestHanoi`

1 Les deux dernières classes doivent être modifiées pour pouvoir être acceptable comme type de disques empilables.