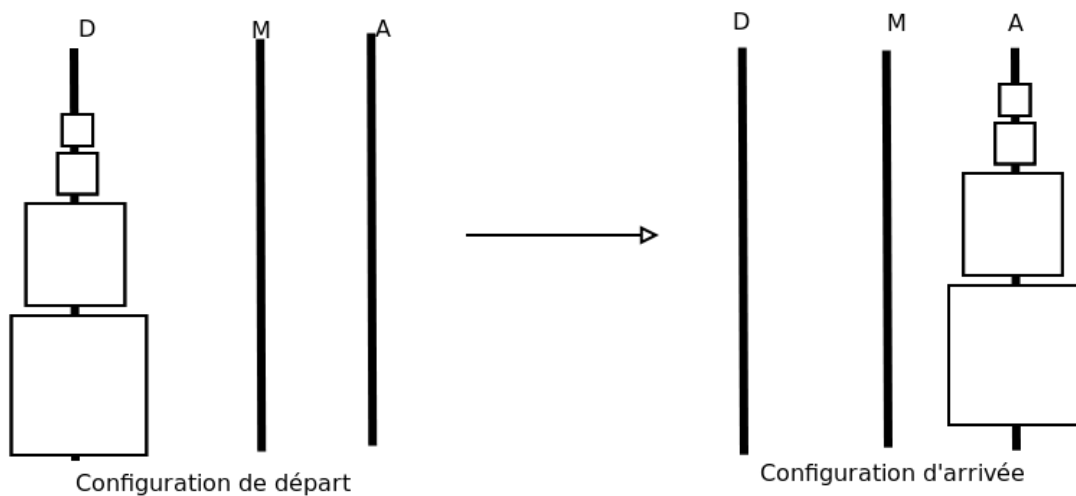


On veut créer différentes versions du célèbre problème des « Tours de Hanoi » en jouant sur les aspects génériques de Java. On rappelle le problème général :



Règles de manipulation:

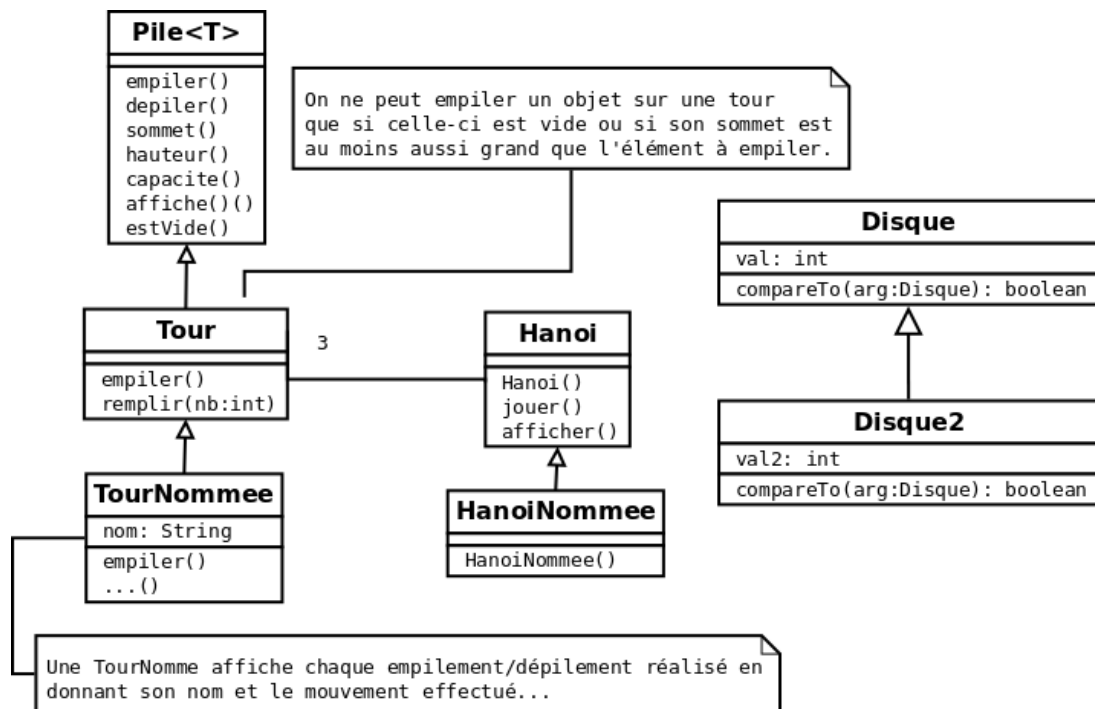
- On ne peut déplacer qu'un disque à la fois
- On ne peut pas poser un disque sur un disque plus "petit" pour une certaine notion de "taille" d'un "disque".

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 rangés 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()); /* déplacer le disque de D vers A */
        D.depiler();
        /* déplacer 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);
    }
}
```

Q1 : On veut une première version dans laquelle les « disques » sont modélisés par une classe `Disque` qui définit un constructeur et une fonction de comparaison. La hiérarchie de classes à implémenter est la suivante (`Pile` représente une pile générique. La capacité de la pile est fixée à la création de l'instance et n'est pas modifiable). `Tour` est une spécialisation de `Pile` dans laquelle on ne peut empiler que des instances de `Disque`. La méthode `empiler` de `Tour` garantit qu'on n'empile jamais un disque sur une instance plus petite, sinon on lève l'exception `ErreurTour`. La classe `Disque2` est une sous-classe de `Disque`. On doit pouvoir construire des instances de `Tour` qui contiennent aussi bien des instances de `Disque` que de `Disque2`.



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.

Q2 : On veut maintenant une version générique des Tours de Hanoi, dans laquelle les classes `Tour` et `TourNommée` sont paramétrées par le type de disque à empiler. On doit pouvoir aussi bien construire des Tours de Hanoi avec uniquement des instances de `Disque`, ou de `Disque2`, ou un mélange des deux. On peut aussi construire des tours de Hanoi avec des instances de `MesEntiers`, mais on ne doit pas pouvoir mélanger des instances de `Disque` et de `MesEntiers` sur une même tour. Implémentez les différentes classes demandées de manière à obtenir une **solution générique et souple d'utilisation en se servant correctement des mécanismes de généricité**.

