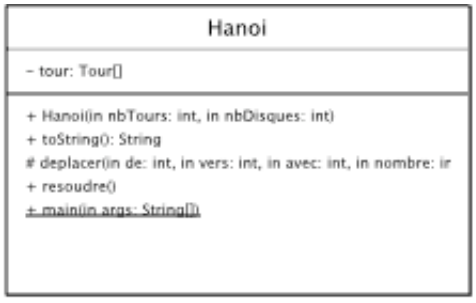
Hanoi
- tour: Tour[]
+ Hanoi(in nbTours: int, in nbDisques: int)
+ toString(): String
# deplacer(in de: int, in vers: int, in avec: int, in nombre: ir
+ resoudre()
+ main(in args: String[])


Tour
- disques: Disque[]
- contenu: int
+ Tour(in hauteur: int)
+ deplacerSur(in t: Tour)
+ deplacableSur(in t: Tour): boolean
+ getSommet(): Disque
+ depiler()
+ empiler(in d: Disque)
+ estVide(): boolean
+ estPlein(): boolean
+ getHauteurMaximale(): int
+ getHauteurActuelle(): int
+ toString(): String


Disque
- taille: int
+ Disque(in taille: int)
+ equals(in o: Object): boolean
+ compareTo(in o: Object): int
+ empilableSur(in d: Disque): boolean
+ getTaille(): int
+ toString(): String

```java
package hanoi;

/**
 * Définit la classe Hanoi
 * @author yroos
 *
 */
public class Hanoi {
    private Tour[] tour;

    /**
     * Construit une nouvelle classe pour la résolution des tours
         de Hanoi
     *
     * @param nbTours le nombre de tours
     * @param nbDisques le nombre de disques par tour
     * @throws IllegalArgumentException si un de ces paramètres est
          inférieur à 1
     */
    public Hanoi(int nbTours, int nbDisques) {
        tour = new Tour[nbTours];

        for (int i = 0; i < nbTours; i++) {
            tour[i] = new Tour(nbDisques);
        }

        for (int i = nbDisques; i > 0; i--) {
            tour[0].empiler(new Disque(i));
        }
    }

    /**
     * Retourne une représentation textuelle d'une étape de
     * résolution des tours de Hanoi
     * @return une représentation textuelle d'une étape de
     * résolution des tours de Hanoi
     */
    public String toString() {
        String s = "";

        for (int i = 0; i < tour.length; i++) {
            s += "|" + tour[i].toString() + "\n";
        }

        return s;
    }

    /**
     * Déplace un certain nombre de disques d'une tour donnée
     * vers une autre tour en
     * utilisant une troisième tour.
     * @param de la tour de départ
     * @param vers la tour de destination
     * @param avec la tour supplémentaire
     * @param nombre le nombre de disques à déplacer
     */
    protected void deplacer(int de, int vers, int avec, int nombre)
            {
        if (nombre > 1) {
            this.deplacer(de, avec, vers, nombre - 1);
        }

        this.tour[de].deplacerSur(this.tour[vers]);
        System.out.println(this);

        if (nombre > 1) {
            this.deplacer(avec, vers, de, nombre - 1);
        }
    }

    /**
     * Résoud le problème des tours de Hanoi pour cette instance.
     */
    public void resoudre() {
        System.out.println(this);
        this.deplacer(0, 1, 2, tour[0].getHauteurActuelle());
    }

    /**
     * Résoud le problème des tours de Hanoi classique.
     */
    public static void main(String args[]) {
        new Hanoi(3, 8).resoudre();
    }
}
```