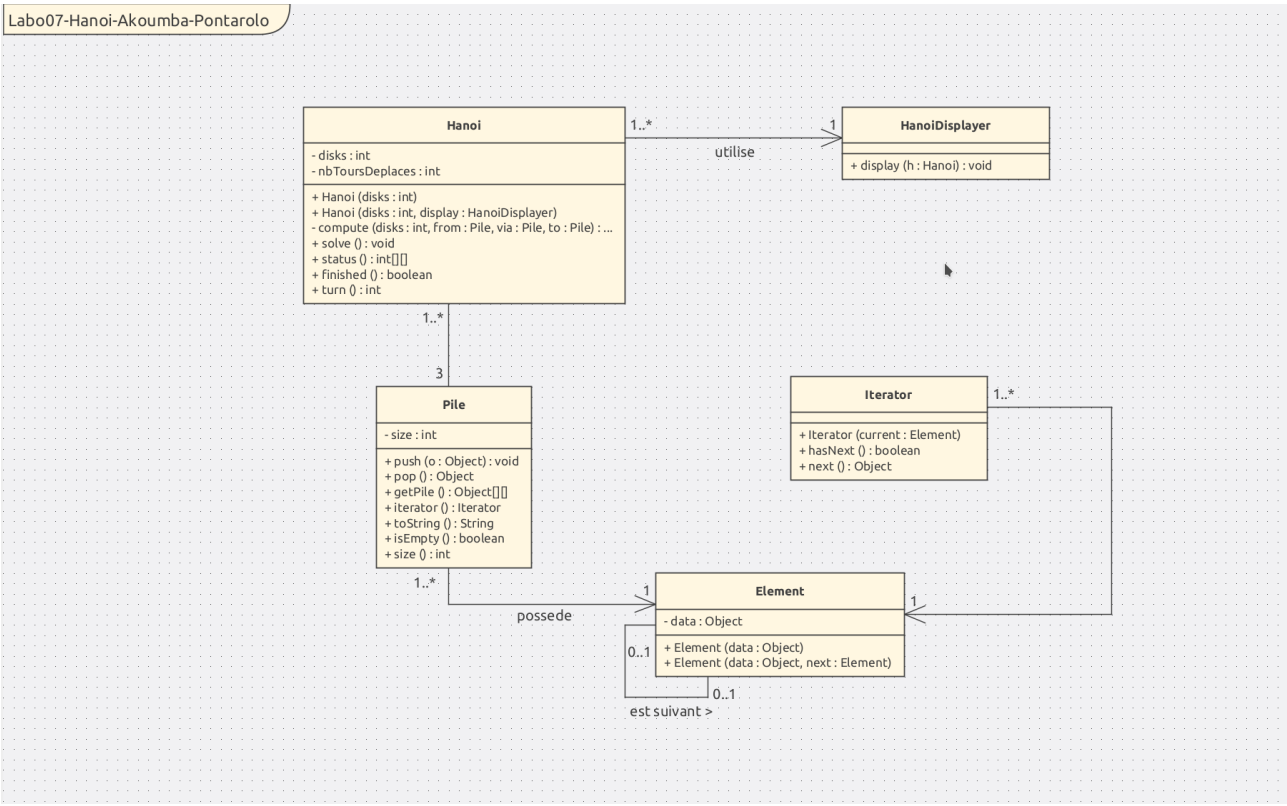
Auteurs: Akoumba Erica, Stefano Pontarolo

RAPPORT POO : LABO-07-HANOI

Diagramme des classes



Algorithme utilisé

Dans notre solution des Tours de Hanoi, nous faisons une récursion sur le plus grand disque à déplacer. Autrement dit, nous écrivons une fonction récursive qui prend comme paramètre le disque se trouvant au sommet, c’est à dire celui que nous voulons déplacer. Notre fonction prendra également trois paramètres indiquant à partir de quelle aiguille la tour doit être déplacée (from), vers quelle aiguille elle doit aller (to), et l'aiguille intermédiaire utilisée temporairement pour faciliter le fait que le plus grand disque se trouve toujours vers le bas dans tous les déplacement cela se produise (via).

Description des classes.

Pour la réalisation du 7 ieme labo de POO intitulé Hanoi, nous avons implémenté differentes classes réparties dans 2 packages distincts:

- ➔ Package util : Dans lequel se trouve toutes les classes qui nous serviront d'outils pour la modélisation de notre problème. Dans celui ci nous trouverons les classes:

  - Element : utilisée pour modéliser les éléments que nous allons manipuler. Elle dispose d'un attribut de type Object contenant la donnée à sauvegarder ,dans le cadre de ce labo ils seront de type int. Et une référence vers l'élément suivant. Elle dispose de 2 constructeurs , un pour créer un élément sans référence sur son suivant et le second pour initialiser l'élément et son suivant.
  - Iterator : Utilisée pour modéliser le comportement d'un itérateur , elle dispose d'un attribut de type Element , d'un constructeur pour initialiser l'élément sur lequel on va itérer, et 2 methodes next() qui renvoie une référence sur l'élément suivant,et hasNext() qui retourne un booleen à true si l'élément a un suivant et false dans le cas inverse.
  - Pile : Pour modéliser le comportement d'une pile générique c'est à dire que l'insertion et la suppression des éléments de font en tête. Elle dispose d'un attribut qui est la tête de pile. Et de plusieurs méthodes utilisées pour insérer supprimer afficher et créer un tableau contenant les éléments de la pile.

- ➔ package hanoi : Dans lequel nous trouverons l'implémentation proprement dite du problème. Elle contient les classes:

  - HanoiDisplayer : qui dispose uniquement d'une méthode display prenant en paramètre un objet de type hanoi. Elle sert à afficher toutes les étapes de la résolution du hanoi sur la console lorsqu'elle est appelée.
  - Hanoi utilise la classe pile grâce à un import , celle ci lui sert à modéliser le hanoi. Elle dispose d'attributs privée : les 3 piles représentant les 3 tours sur lesquelles les disques vont être déplacés, d'une référence vers HanoiDisplayer pour l'affichage du nombre de disques, et d'un attribut pour savoir savoir à tout moment le nombre de disques déplacés . 2 constructeurs pour l'initialisation de ces éléments de méthodes publiques et privées pour matérialiser le déplacement des disques de la tour 1 à la tour 3 en passant

par la tour 2 en respectant le fait que, considérant 2 disques, le disque le plus grand se trouvera toujours en dessous du plus petit.

- App : utilisée pour exécuter le jeu hanoi. Dans celle ci , si un élément est passé en paramètre, celui ci représentera le nombre de disques que nous allons devoir déplacer et l'affichage se fera dans la console. Tandis que si aucun argument n'est entré. Le jeu se fera à l'aide d'une interface graphique.

Bien entendu des tests ont été mis en œuvre pour tester chacune des fonctionnalités de ces classes.

Réponse à la question de complexité

La complexité pour résoudre le problème de la tour de hanoi est O(2^n). Pour 64 disques, le nombre d'opérations sera 2^64. Étant donné qu'on déplace un disque à la seconde, il faut 1,84467441 * 10^19 secondes ce qui correspond à 584'942'417'808,22 d'années.
Donc, il nous reste 584'942'417'808,22 - 13,7 * 10^9 = 571,2 milliards d'années