Quentin Forestier, Herzig Melvyn

POO1 – 19.11.2020

Tours de Hanoï

Laboratoire 7



Table des matières

[Algorithme utilisé 2](#_Toc56689679)

[Description des classes 2](#_Toc56689680)

[Stack 2](#_Toc56689681)

[Element 2](#_Toc56689682)

[Examinator 2](#_Toc56689683)

[Hanoi 2](#_Toc56689684)

[HanoiDisplayer 2](#_Toc56689685)

[Diagramme de classe 0](#_Toc56689686)

[Tests de la pile (Stack.java) 0](#_Toc56689687)

# Algorithme utilisé

Afin de résoudre le problème des tours de Hanoï, nous avons opté pour une solution itérative afin de ne pas être limité par la taille limitée de la pile de récursion.

Nous utilisons deux fonctions :

* Public void solve()
* Private void transfer(Stack s1, Stack s2)

La fonction transfer nécessite deux piles en paramètre. Elle cherche quelle pile a l’élément le plus petit pour le déplacer sur l’autre.

La fonction solve est l’élément central. En fonction du numéro du tour de résolution, elle détermine entre quelles piles nous devons déplacer un disque pour avancer la résolution. Il est inspiré de la solution proposée sur le site GeeksForGeeks[[1]](#footnote-1) et du cours d’ASD1. L’algorithme s’arrête dès que nous avons effectués le nombre maximal de déplacement pour N disques. Le nombre maximal est obtenu avec la formule (2^N)-1.

# Description des classes

## Stack

Package : util Visibilité : publique

Implémente le concept de pile LIFO. Elle emplie des objets Element qui encapsule des objets Object.

## Element

Package : util Visibilité : package

Eléments d’une pile. Un ensemble d’Elements forment une liste simplement chaînée.

## Examinator

Package : util Visibilité : publique

Itérateur permettant de parcourir une pile de haut en bas.

## Hanoi

Package : hanoi Visibilité : publique

Classe modélisant le problème des tours de Hanoï avec un nombre de disques quelconque. Les aiguilles sont modélisées par trois Stack.

## HanoiDisplayer

Package : hanoi Visibilité : publique

Par défaut, permet de représenter un objet Hanoï dans la console.

# Diagramme de classe

Contrainte :

Un élément ne peut pas être suivi par lui même

# Tests de la pile (Stack.java)

Pour tester l’implémentation de la classe Stack nous avons écrit un programme TestStack.java. Les résultats ont été vérifiés manuellement et tous ont été validés. Pour afficher le tableau nous avons créé une fonction spécifique.

|  |  |
| --- | --- |
|  | Doit afficher [ ] |
|  | Comme la pile est vide la taille vaut 0 et est isEmpty true |
|  | Top sur pile vide lance une RuntimeException |
|  | Pop sur pile vide lance une RuntimeException |
|  | Sur une pile vide, la création de l’examinator fonctionne mais son utilisation lance une NullPointerExcepton |
|  | Affiche { } |
|  | Affiche  [ <abc> <4> <a> ] |
|  | Le contenu du tableau est similaire à la pile. |
|  | La pile n’est plus vide et la taille vaut 3 |
|  | Le top retourne abc |
|  | Dépiler le top résulte en une pile  [ <4 <a> ] non vide de taille 2 |
|  | Le reste de la pile est parcouru avec l’examinator jusqu’à la fin |

1. <https://www.geeksforgeeks.org/iterative-tower-of-hanoi/> [↑](#footnote-ref-1)