Rapport de laboratoire

**HEIG-VD**

TIC

Laboratoire POO

Labo 7

**Réalisé par :**

Ivan Vecerina

Thibault Seem

**A l’attention de :**

M. Donini

M. Decorvet

**Dates :**

Début du laboratoire : 17 novembre 2021

Fin du laboratoire : 2 décembre 2021

Table des matières

[1 Introduction 3](#_Toc89272763)

[2 Diagramme de classe 3](#_Toc89272764)

[3 Choix de modélisation 3](#_Toc89272765)

[3.1 Description des classes 3](#_Toc89272766)

[3.2 Description algorithme 3](#_Toc89272767)

[4 Réponse à la question 3](#_Toc89272768)

[5 Tests 3](#_Toc89272769)

[6 Sources 3](#_Toc89272770)

# Introduction

Dans ce laboratoire, nous devons déveloper les classes nécéssaire à la representation du problème des tours de Hanoi ainsi que les fonctions requisent pour ça resolution. Les tours de Hanoi sont représentées par 3 stacks

# Diagramme de classe

# Choix de modélisation

## Description des classes

### Element

La classe element représente les disques. Elle contient une valeur et une référence sur l’élément (le disque) en dessous.

Elle n’a que des getters comme méthodes.

### Stack

La classe stack représente une aiguille. Elle n’a que deux attributs, soit une hauteur et une référence sur l’objet au sommet.

Comme une stack classique, elle proposent les méthodes push(), pop(), toArray() et toString() en plus des getters.

### ElementIterator

ElementIterator est une classe fonctionnelle nous permettant de parcourir les éléments de la stacks en partant du sommet. Elle n’a pour attribut qu’une référence sur un élément initialisé au sommet de la stack.

Elle propose les méthodes classiques, next() et hasNext(). En plus de ces deux méthodes, elle propose la méthode getElementValue() qui retourne la valeur de l’élément référencé.

## Description algorithme

L'algorithme utilisé ici est l'algorithme de résolution des tours de Hanoi récursif. Pour pouvoir résoudre ce problème, il faut déplacer tous les disques de la première tour à la dernière, en ne pouvant pas mettre un disque sur un autre disque de taille inférieur.

Pour résoudre ce problème, l'algorithme récursif va en quelque sorte transformer ce problème complexe en une multitude de problème, identique, mais plus simple. Il va donc faire en sorte de résoudre ce problème non pas avec des tours de taille n, mais de n-1, mais avec comme cible non plus le 3ème pilier, mais le pilier central, afin de pouvoir déplacer le plus grand des disques sur le 3ème pilier.

Cependant, il est possible qu'il y ait encore plus d'un disque à déplacer. Et bien l'algorithme va essayer de résoudre ca en regardant le problème, mais avec une taille de tour de n-2, et en utilisant non plus le pilier central comme objectif, mais le 3ème pilier, puisque le 2ème pilier doit accueillir la base de la tour de n-1. Et on continue comme ça, en inversant les piliers qui doivent recevoir la tour, jusqu'à n'avoir plus qu'une tour de taille n-(n-1), soit 1.

À ce moment-là, l'algorithme peut commencer à déplacer les disques. Une fois le disque le plus petit déplacé, on remonte la récursivité, on déplace le disque de taille 2 sur la tour qui n'a pas servit juste avant (d’où aucun disque n'est parti et aucun n'est arrivé). À ce moment, le nouveau but est de déplacer la tour faite des disques plus petits que celui que l'on vient de déplacer, et de la placer par-dessus ce "grand disque". Pour faire cela, l'algorithme recommence du début, mais en ne prenant en compte que la tour de "petit disque", et en ayant comme objectif le pilier ou se trouve le "grand disque" qui vient d'être placé.

L'algorithme continue ainsi jusqu'à quitter la récursion, ce qui signifie qu'il a déplacer l'entièreté des disque sur le pilier cible, et que le problème des tours de Hanoi est donc résolu.

# Réponse à la question

Nous avons 64 disques.

Le nombre de coups minimal pour résoudre le problème des Tours de Hanoi et de:

**2n – 1** ; “*n” étant le nombre de disques*.

Pour 64 disques nous obtenons donc **18’446’744’073’709’551’615** mouvements.

Les moines effectuant un mouvement par seconde, il leur faudra le même nombre en secondes.

Transformons ses secondes en années en divisant par (3600 \* 24 \* 365.25).

Nous obtenons à peu près **584.54 milliards d’années.**

A ceci nous pouvons donc soustraire l’âge actuel de l’univers (13.8 milliards d’années).

Selon la légende hindoue, l’univers disparaitra dans **570.74** **milliards d’années.**

# Sources

* Pour la stack, ses éléments et l’itérateur génériques, nous nous sommes inspires de l’implémentation proposée par :   
  <https://levelup.gitconnected.com/selfmade-stack-class-in-java-d401dc7d68f0> f
* Algorithme de résolution des tours de Hanoi :  
  <https://www.geeksforgeeks.org/c-program-for-tower-of-hanoi/>