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 développer les classes nécessaires à la représentation du problème des tours de Hanoi ainsi que les fonctions requises pour sa résolution. Les tours de Hanoi sont représentées par 3 aiguilles, l’une d’entre elles à un certain nombre de disques croissant de haut en bas. Le but du problème et des transférer tous ces disques de l’aiguille de départ à une autre en respectant deux contraintes. La première et que l’on ne peut déplacer qu’un disque à la fois. La seconde, un disque ne peut être déplacé que sur un disque plus grand que lui ou sur une aiguille vide. Nous avions quelques contraintes pour ce projet. Nous devons créer notre propre classe Stack afin de représenter les aiguilles ainsi que notre propre itérateur pour parcourir ses éléments. Nous avons aussi quelques fonctions imposées afin de permettre l’outil graphique de fonctionner.

# 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.

### Hanoi

## Description algorithme

# 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>