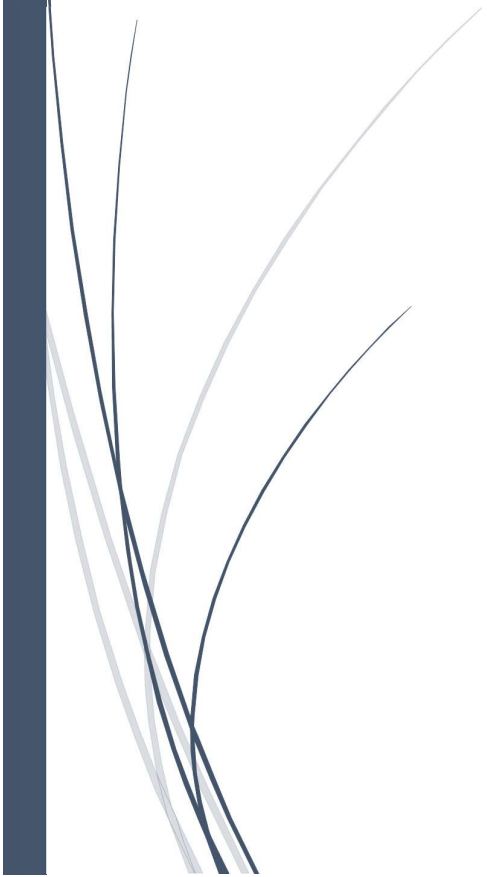


A dark blue vertical bar runs down the left side of the page. A blue arrow points to the right from this bar, containing the date.

02/12/2021

Rapport Laboratoire 7 : Hanoi

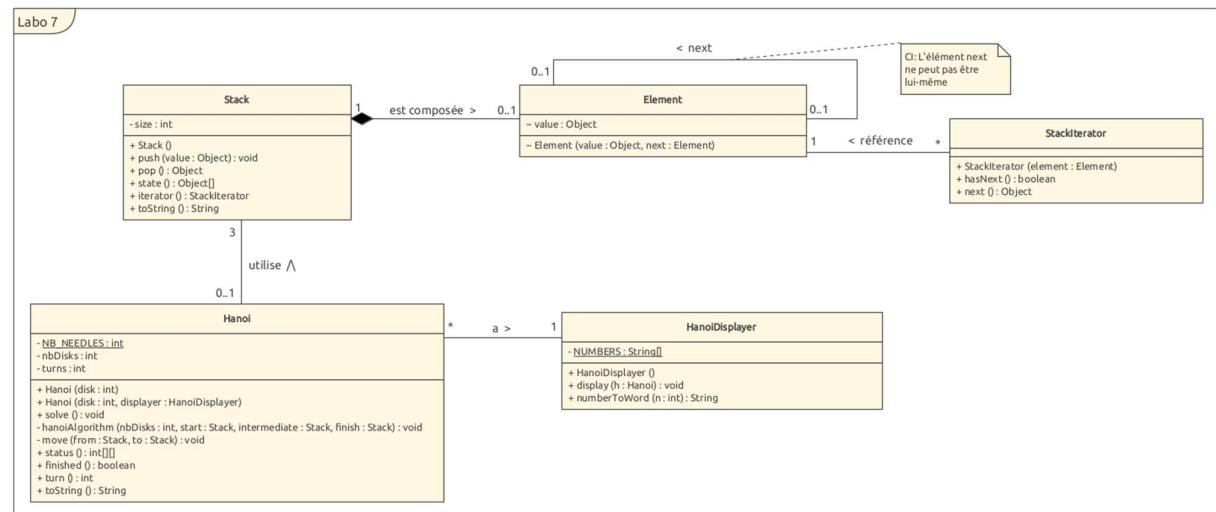
Several thin, curved lines in shades of blue and grey originate from the bottom left corner and sweep upwards and to the right.

Valentin Kaelin et Jonathan Friedli

Introduction

Dans le cadre de ce laboratoire n°7, nous avons dû implémenter un programme permettant de résoudre le problème des tours d'Hanoi de deux façons différentes : soit directement dans le terminal ou alors via une interface graphique. Dans les deux cas l'utilisateur peut choisir la taille de la tour initiale. L'implémentation utilise une pile que nous devons créer à la main.

Diagramme des classes



Algorithme utilisé

Nous avons utilisé l'algorithme récursif, présenté notamment sur Wikipédia : https://fr.wikipedia.org/wiki/Tours_de_Hanoï_-_Solution_récurrente

Cette solution a l'avantage d'être simple à implémenter et très compréhensible. L'idée de cet algorithme est la suivante :

Pour déplacer une tour de n disques de A vers C, on effectue ces trois étapes :

1. Déplacer la tour des $n-1$ premiers disques de A vers B (appel récursif).
2. Déplacer le plus grand disque de A vers C.
3. Déplacer la tour des $n-1$ premiers disques de B vers C (appel récursif).

Réponse à la question 1

Le nombre de déplacements pour résoudre la tour de Hanoï vaut $2^n - 1$, avec n le nombre de disque de la tour. Dans notre cas, n vaut 64 donc il faut $2^{64} - 1$ déplacements. Ce qui vaut 18'446'744'073'709'552'000 déplacements.

Les moines faisant un déplacement par seconde il faudrait 18'446'744'073'709'552'000 secondes. L'univers ayant environ 13.8 milliards d'année mais les moines n'existaient sûrement pas (à ma connaissance) au début de l'univers. Cependant, nous allons supposer que le premier déplacement a eu lieu il y a exactement 13.8 milliards d'années. Nous soustrayons donc 13'800'000'000 * 365,25 * 24 * 3600 à 18'446'744'073'709'552'000 ce qui donne :

18'446'744'073'709'552'000 - 435'494'880'000'000'000 = 18'011'249'193'709'552'000 secondes restantes ce qui représente environ 570'742'046'090 années. Donc environ **570,7 milliards d'années**.

Choix de modélisation et d'implémentation

Hanoi

Nous avons représenté les trois aiguilles sous forme d'un tableau pour plus d'évolutivité, même si l'algorithme de résolution serait à revoir dans le cas où le nombre d'aiguilles serait plus grand que trois.

La représentation graphique du tour a été implémentée dans la classe Hanoi afin d'éviter de devoir faire des accesseurs sur les différentes aiguilles pour le réaliser dans la classe HanoiDisplayer. Cette dernière se contente donc d'avoir une fonction helper et appelle le toString de l'instance de la classe Hanoi reçue en paramètre.

Stack

Nous avons tout d'abord essayé de faire une vraie version générique de la Stack mais il n'est pas aisé de travailler avec des tableaux génériques (à la place d'ArrayList). Nous nous sommes donc rabattus sur la version « générique » vue en cours grâce à la classe Object.

Afin de suivre les conventions Java, la méthode next() dans l'itérateur retourne l'élément courant tout en le remplaçant par le prochain élément par après. De même, la méthode hasNext() vérifie que l'élément courant existe bien et pas un élément plus loin. Cela nous permet de boucler facilement sur tous les éléments de la pile.

Nous avons choisi la visibilité package pour tous les éléments de la classe Element car nous l'utilisons plus comme une structure utile à l'interne dans l'implémentation de la pile. En effet, il n'est pas utile d'avoir accès à l'implémentation de l'élément en dehors du package util, dans le code métier.

La CI comme quoi le next d'un élément ne doit pas être lui-même n'a pas été implémentée directement dans le code étant donné que la classe est en visibilité package. L'utilisateur utilisant notre librairie ne pourra faire cette bêtise et nous partons du principe que nous ne faisons pas ça non plus dans notre code.

Tests effectués

Nous avons tout d'abord réalisé le test demandé dans la donnée permettant de lancer la résolution des tours d'Hanoi via une interface graphique ou la console.

Par la suite, nous avons testé l'implémentation de notre classe Stack avec via les tests suivants :

Test	Résultat attendu et observé
Ajouter une valeur dans la pile	La valeur est ajoutée tout au haut de la pile
Retirer une valeur dans la pile	L'élément tout en haut de la pile est supprimé et sa valeur est retournée
Création d'une pile vide et affichage de celle-ci	Les opérations se déroulent sans soucis
Affichage des éléments d'une pile grâce à sa représentation sous forme de chaîne de caractères	Les éléments sont affichés sous la forme demandée, c'est-à-dire : [<a> <c>]

L'itérateur de la pile fonctionne correctement	Il est possible d'itérer sur toutes les valeurs de la pile sans problème et dans le bon ordre
Récupération des données de la pile via sa méthode state retournant un tableau	Le tableau contient toutes les valeurs de la pile dans le bon ordre
Récupérer les données de la pile via la méthode state ne brise pas l'encapsulation	Modifier le tableau retourné ne modifie pas le tableau dans la classe de la pile
Retirer un élément d'une pile vide	Exception lancée lors de l'opération
Ajouter une valeur d'un type différent (String dans notre cas) dans la pile	Les opérations se déroulent sans soucis