Une image contenant symbole, Emblème, logo, Graphique

Description générée automatiquement

**Département de mathématiques et informatique**

Travail Pratique 1

**Cours**

INF-1004

**Présenté à**

M. Adam Joly

**Par**

David Gauvreau

David Descôteaux

Félix Lefebvre

Erik Ayotte

**Hiver 2025**

# Table des matières

[Table des matières 1](#_Toc72700752)

[Guide utilisateur 2](#_Toc1134160485)

[Distribution des tâches 3](#_Toc28622281)

[Cas de tests 4](#_Toc1323922431)

[Explications des stratégies algorithmiques principales 4](#_Toc336088107)

[Problèmes, difficultés ou commentaires sur le travail, s’il y a lieu 4](#_Toc738876046)

# Guide utilisateur

Dans ce jeu l’utilisateur sera appelé à compléter le casse-tête communément appelé les Tour d’Hanoi. Il y a trois tours et on doit déplacer une à une les anneaux empilés dans la première tour, afin de les replacer dans l’ordre de grandeur dans la dernière tour. La seule règle : l’anneau en dessous d’un autre anneau ne peut être plus petite que celle du dessus, c’est-à-dire que les anneaux doivent toujours être empilé par ordre de grandeur, des plus grandes vers le bas aux plus petites vers le haut.

Au lancement de l’application, le menu principal est affiché et l’utilisateur doit entrer le chiffre correspondant à son choix d’action.

Menu principal :

#1 – Déterminer le nombre d’anneaux (3 par défaut) :

Permet à l’utilisateur de changer le degré de difficulté du jeu en augmentant ou diminuant le nombre d’anneaux à déplacer. Le minimum possible est de 1 anneau et un maximum de 9 anneaux. Compte tenu de la difficulté d’un tel défi, nous avons limité le nombre d’anneaux entre 1 et 9.

#2 – Réinitialiser les tours

Réinitialise le jeu au départ et conserve le même nombre d’anneaux choisis par l’utilisateur.

#3 – Lancer une partie :

L’utilisateur démarrera une partie avec le nombre d’anneaux sélectionnés à l’option #1 ou trois (par défaut). Il sera appelé ensuite à entrer la lettre de la tour (A, B ou C) de l’anneau qu’il désire bouger. Il devra ensuite entrer la lettre de la tour (A, B ou C) où il désire déposer l’anneau précédemment sélectionné. Et répéter ainsi jusqu’à la réussite. En tout temps, l’utilisateur peut entrer le chiffre zéro (0) pour quitter la partie et retourner un menu principal.

#4 – Montrer la solution :

En choisissant cette option du menu, l’utilisateur pourra voir la solution au problème, résolue automatiquement par l’application, coup par coup avec une représentation visuelle et textuelle du mouvement des anneaux sur les tours.

#5 – Quitter :

Permet à l’utilisateur de quitter l’application normalement.

# Distribution des tâches

David Descôteaux : Test cases, révision

Félix Lefebvre : Structure général, le toString

Erik Ayotte : Résolution automatique et son affichage (récursion) – Guide utilisateur

David Gauvreau : Class Main, structure générale

# Cas de tests

\*Voir annexe Test-Case-TP1.xlsx

# Explications des stratégies algorithmiques principales

- Nous avons opté pour une résolution par Switch Case pour identifier les inputs de l’utilisateur au lieu de “IF” imbriqués.

- Afin de simplifier notre approche, nous avons été plus “verbose” qu’il aurait été possible de faire, mais nous voulions que ce soit le plus clair possible vu l’approche par équipe. Par exemple, nous avons mis des “break” dans notre Switch case ce qui aurait pu être résolu d’une autre manière.

- Nous avons conservé la base de la fonction récursive du laboratoire 4.2 en convertissant le code et modifiant les paramètres et variables à celles propres à notre code.

- Nous avons aussi limité le nombre maximum d’anneau à 9 pour éviter des débordements (résolution automatique) et aussi éviter une difficulté trop élevée du jeu pour l’utilisateur.

# Problèmes, difficultés ou commentaires sur le travail, s’il y a lieu

* Identifier les tours par A, B, C nous a causé quelques problèmes. Nous avions utilisé au départ le (“) au lieu d’un (‘) pour identifier un Char dans notre Switch Case d’action de l’utilisateur. Devant le problème, nous avons convenu d’utiliser 1, 2 ou 3, même après avoir compris le bloquant.
* Nous avons butté surtout sur des erreurs de syntaxe ou plutôt des particularités liées au langage Java.
* La table des matières... ;-)