INF1004 – Structures de données et algorithmes

Session HIV-25

TRAVAIL #1 : TOURS D'HANOI REVISITÉES

Indications principales	
Date de présentation de l'énoncé	11 février 2025
Date de remise du travail	18 mars 2025, 11h55 (sur le portail de cours)
Politique sur remises tardives	-25% par jour de retard, à partir du mars 2025 à 11h56
Éléments et format de remise	Rapport et fichiers .java ou .NET remis sous forme électronique sur le portail de cours dans un seul fichier compressé (.zip); aucun rapport imprimé nécessaire. Aucune autre famille de langage acceptée sauf .NET et Java.
Pondération	15% de la note finale
Nb d'étudiants par équipe	2 à 5

Énoncé du travail

Dans le cadre de ce premier travail pratique, vous devez implémenter le jeu des tours d'Hanoi sous la forme de piles. Plus précisément**, en vous basant sur le laboratoire 4.2,** vous devez faire en sorte qu'un jeu de tours d'Hanoï soit composé de 3 tours (cylindres) exactement, une tour étant une pile d'anneaux. Un utilisateur, via un menu en console, doit pouvoir déplacer des anneaux d'une pile à une autre et, évidemment, les algorithmes doivent permettre d'assurer que le déplacement est « légal », c'est-à-dire qu'un anneau ne peut être sur un autre anneau de diamètre inférieur. Aussi, toujours via ce menu, l'utilisateur doit pouvoir choisir de voir la solution et, dans ce cas, un algorithme récursif montre la solution. En résumé, il s'agit d'une version « améliorée » du laboratoire 4.2 dans laquelle l'utilisateur peut en plus manuellement faire des déplacements, le tout agrémenté d'un menu.

Encore plus concrètement, vous devez :

- Créer une classe `Anneau`, qui possède un diamètre fourni dans le constructeur et qui ne peut être modifié par la suite;
- Créer une classe `Tour`, qui implémente une pile dont la taille maximum dépend du nombre d'anneaux avec lequel on joue et qui manipule donc un tableau d'anneaux :

- Vous devez ajouter les opérations primaires et secondaires à la manière de ce que vous retrouvez dans les notes de cours (`push()`, `pop()`, `peek()`, `isEmpty()` et `isFull()`);
- Ajoutez aussi une méthode qui permet de vider la pile « d'un seul coup », que vous pouvez appelée `vider()` ou `clear()`, par exemple;
- Chaque tour a aussi un alias (une variable d'instance de type `char`) qui sert à identifier la tour (A, B ou C) lors de l'affichage;
- Spécifiez une redéfinition de la méthode `toString()` de façon à afficher l'alias et le diamètre de chacun des anneaux dans la tour (chaque emplacement libre dans la tour doit afficher un tiret (-); par exemple, pour un jeu à 5 anneaux, la représentation textuelle doit être :

  C: 543--

- Créer une classe `ToursHanoi`, qui possède comme attributs un tableau contenant exactement 3 tours et un second attribut qui indique le nombre d'anneaux dans le jeu :
  - Vous devez avoir un constructeur ayant comme paramètre le nombre d'anneaux dans le jeu et dans lequel vous créez chacune des tours en donnant les alias A, B et C, puis vous appelez la méthode `reinitialiser();`
  - Ajoutez une méthode `reinitialiser()` qui sert à vider chacune des tours et à remplir la tour A. Les anneaux ont comme diamètre 1 à n anneaux (p. ex. s'il y a 3 anneaux, alors les diamètres des anneaux sont de 1, 2 et 3);
  - Ajoutez une méthode `deplacer()` qui prend en paramètre deux caractères (`de` et `vers`) et qui sert à effectuer le déplacer de l'anneau au sommet de la tour dont l'alias est celui donné par le paramètre `de` vers la tour dont l'alias correspond à `vers` :
    - Vous devez vérifier que le déplacement est légal avant qu'il soit effectué; pensez donc aux situations qui pourraient faire en sorte que le déplacement soit impossible/possible; si le déplacement est illégal, vous devez afficher à la console ceci :

      Déplacement illégal

      Puis, vous retournez **faux**; si le déplacement est légal, suite au déplacement, vous retournez **vrai**;
    - Cela doit fonctionner pour 'A' comme pour 'a' (donc la case ne doit pas importer), et seulement pour les lettres A, B ou C;

        - Vous pouvez, si vous le voulez, avoir une méthode privée prenant en entrée l'alias et donnant en sortie l'indice dans le tableau de tour correspondant (A ou a —> 0; B ou b —> 1; C ou c —> 2);

        - Lorsque le déplacement est jugé valide, affichez à la console une trace du déplacement effectué, tel que :

            L'anneau de diamètre 3 est déplacé de la tour A vers la tour C

        - Faites en sorte que cette méthode soit aussi claire et concise (simple et efficace) que possible;

    - Ajoutez une méthode `resoudre()` qui réinitialise le jeu d'abord, puis qui fait l'appel récursif initial de la méthode `deplacerAuto()`;

    - Ajoutez la méthode `deplacerAuto()` qui implémente l'algorithme récursif tel que présenté dans le livre ou le laboratoire 4.2, mais qui cette fois-ci utilise la méthode `deplacer()` pour faire de véritables déplacements dans les tours (piles); affichez à la console le jeu des tours d'Hanoi après chaque appel de la méthode `deplacer()` afin de montrer l'évolution des coups;

    - Enfin, ajoutez une méthode `toString()` qui retourne le contenu de chacune des tours et qui devrait donner quelque chose tel que :

        A: 1----
        B: 2----
        C: 543--

- Dans la classe `Main`, lorsque le programme est exécuté, vous devez faire afficher le menu suivant à la console :

    Tours:
    A: 321
    B: ---
    C: ---

    [MENU]

    1: Déterminer le nombre d'anneaux (3 par défaut)
    2: Réinitialiser les tours
    3: Jouer un coup
    4: Montrer la solution
    5: Quitter
    Faites votre choix et appuyez sur ENTER

Suite à la réalisation de chacun des choix, le contenu des tours et les options du menu sont toujours réaffichés;

Pour chacune des options :

1) Déterminer le nombre d'anneaux : l'utilisateur indique un nombre entre 1 et 9 inclusivement pour indiquer le nombre d'anneaux que doit posséder le jeu des tours d'Hanoi :

    - Un nouveau jeu est recréé en fonction de ce nombre (par défaut lors de l'exécution, le nombre d'anneaux pour les tours est de 3);

    - Si l'utilisateur tape 0, alors il revient au menu principal;

    - S'il tape autre chose qu'un nombre entre 0 et 9, le menu doit être réaffiché et on doit indiquer à l'utilisateur que le nombre précédemment entré était incorrect.

2) Réinitialiser le jeu : le jeu est réinitialisé;

3) Jouer un coup : l'utilisateur est invité à entrer l'alias de la tour dont l'anneau au sommet doit être déplacé, puis l'alias de la tour qui recevra l'anneau. Tant et aussi longtemps que le déplacement est jugé illégal, l'utilisateur doit entrer de nouveau ces informations; par exemple :

    À partir de quelle tour souhaitez-vous déplacer un anneau?
    A

    Vers quelle tour?
    C

    L'anneau de diamètre 1 est déplacé de la tour A vers la tour C.

4) Montrer la solution : la méthode *resoudre()* du jeu est appelée;

5) Quitter : le programme se termine.

Faites en sorte que la présentation en console soit soignée, qu'il y ait des commentaires pertinents dans le code lorsque nécessaire et de respecter l'encapsulation.

Contenu du rapport

- Page couverture;
- Distribution des tâches dans l'équipe;
- Court guide utilisateur / cas de tests;
- Explications des stratégies algorithmiques principales (déplacement, méthode récursive, etc.);
- Problèmes, difficultés ou commentaires sur le travail, s'il y a lieu;

Cas de tests à inclure dans le rapport

Pour chacun des cas de tests qui suivront, vous devez :

- Nommer le cas de test
- Ajouter une ou des captures d'écran démontrant que le résultat attendu est atteint (ou si vous n'arrivez pas, mettez quand même une capture d'écran de ce que vous obtenez)
- S'il y a lieu, décrivez les résultats (si vous n'y êtes pas arrivés, décrivez ce que vous avez tenté et/ou la nature de l'échec)

Les cas de tests sont les suivants :

1) Le menu doit s'afficher conformément à ce qui est attendu

2) L'utilisateur fait plusieurs déplacements manuels, dont le dernier tenté est illégal. La console affiche l'ensemble des déplacements, le coup illégal, et la représentation visuelle adéquate des tours d'Hanoï

3) Après avoir joué plusieurs coups, un changement de nombre d'anneaux (> 5) est fait et la représentation des tours d'Hanoï est affichée à la console, avec les anneaux en position de départ / réinitialisée

4) La console affiche toutes les étapes de déplacements pour résoudre les tours d'Hanoï pour 3 et 6 anneaux.

Format de remise

Vous devez remettre le rapport et la solution complète dans un seul format compressé sur le portail de cours dans la zone Travaux prévue, avant la date de tombée donnée en entête du présent énoncé.

Par solution complète est entendue l'ensemble du répertoire contenant le fichier de solution (.sln en .NET) et/ou de projet (.csproj en .NET, les fichiers sources (.java ou .cs), les fichiers compilés/exécutables (notamment les .exe/.jar). L'option d'ouvrir un projet ou une solution en NetBeans, Eclipse ou Visual Studio devrait toujours permettre de bien ouvrir ce que vous remettez

directement sans devoir reconstruire un nouveau projet, et tout devrait être compilable en plus d'être déjà exécutable sans compilation.

Aucun retard ne sera toléré à moins de raison exceptionnelle. Tout retard d'une journée supplémentaire entrainera automatiquement la perte de 25% de la note maximale. Comme vous devez remettre le travail via le portail de cours, si vous dépassez la date de tombée, vous ne pourrez plus le déposer; dans un tel cas, envoyez-le-moi par courriel, mais vous serez considéré comme en retard. Aucunes raisons telles que *« le dépôt n'a pas fonctionné* » ou « *je n'avais pas accès à Internet* » ne seront acceptées.

Grille d'évaluation

Sujet d'évaluation	***Pts (/15)***
Rapport complet (incl. cas de tests) et tous les fichiers .java (JAVA) ou .cs/.csproj/.sln (.NET)	2.5
Commentaires et forme du code (indentations, nomenclatures, …)	1
Classe `Anneau`	0.5
Classe `Tour` (pile) : - Implémentation de la pile : 2.5 - `vider()`, `toString()` et autres: 1.5	4
Classe `ToursHanoi` : - Déplacement : 2 - Résolution automatique et algorithme récursif :2 - (Ré)initialisation, `toString()` et autres : 1	5
Classe `Main` : - Menu et affichage conformes et esthétiques : 1.5 - Saisie adéquate des données utilisateur : 0.5	2

Pour chacun des éléments concernant directement des classes et des méthodes demandées, les fonctionnalités doivent être implémentées adéquatement et efficacement, c'est-à-dire en utilisant les structures de données/variables attendues ou raisonnables, ainsi que le principe de récursivité lorsqu'élégant ou exigé.