UNIVERSITÉ DU QUÉBEC À TROIS-RIVIÈRES

PROJET DES TOURS D’HANOÏ

TRAVAIL PRÉSENTÉ

COMME EXIGENCE PARTIELLE DU COURS

INF-1004 : STRUCTURES DE DONNÉES ET ALGORITHMES

PAR

JULIEN BOISVERT ET ALEXIS YOUNG

14 MARS 2023

**Listing du code**

***CLASSE ANNEAU:***

namespace ProjectTourHanoi  
{  
 public class Anneau  
 {  
 public int Diametre { get; }   
  
 */\*  
 \* \brief : Constructeur Anneau surchargé   
 \* \param[in] : Un int qui représente le diamètre de l'anneau  
 \*/* public Anneau(int diametre)  
 {  
 Diametre = diametre;  
 }  
  
 */\*  
 \* \brief : Afficher l'anneau  
 \* \param[in] : Aucun  
 \* \return : String représentant le diamètre de l'anneau  
 \*/* public override string ToString()  
 {  
 return Diametre.ToString();  
 }  
 }

***CLASSE TOUR:***

using System;  
using System.Security.Cryptography;  
  
namespace ProjectTourHanoi  
{  
 public class Tour  
 {  
  
 public char \_lettreTour; *//Variable représentant la lettre associée à la tour* private int \_nbAnneau; *//Variable du nombre d'anneau de la tour* public int \_top; *//Variable du sommet de la pile* private Anneau[] \_tours; *//Tableau d'anneaux  
   
   
 /\*  
 \* \brief : Constructeur Tour surchargé   
 \* \param[in] : Un char qui représente la lettre de la tour   
 \* \param[in] : Un int pour le nombre d'anneaux sur la tour  
 \* \param[in] : Un int pour le nombre d'anneaux maximal sur la tour  
 \*/* public Tour(char lettreTour, int nbAnneau, int maxAnneau)  
 {  
 \_lettreTour = lettreTour;  
 \_nbAnneau = nbAnneau;  
 \_tours = new Anneau[maxAnneau];  
 \_top = nbAnneau - 1;  
 int diametre = nbAnneau;  
   
 *//Ajout des anneau sur la tour* for (int i = 0; i < nbAnneau; i++)  
 {  
 \_tours[i] = new Anneau(diametre);  
 diametre--;  
 }  
 }   
   
 *//////////Fonctions primaires de la pile Tour[]//////////  
   
 /\*  
 \* \brief : Ajouter un anneau sur la pile  
 \* \param[in] : Un anneau qui sera ajouter sur la pile   
 \* \return : Aucun  
 \*/* public void push(Anneau nbAnneau)  
 {  
 \_tours[++\_top] = nbAnneau;  
 }  
   
 */\*  
 \* \brief : Enlever un anneau de la pile  
 \* \param[in] : Aucun  
 \* \return : L'anneau qui est retiré de la pile  
 \*/* public Anneau pop()  
 {  
 return \_tours[\_top--];  
 }  
  
 *//////////Fonctions secondaires de la pile Tour[]//////////  
   
 /\*  
 \* \brief : Retourne l'anneau sur le dessus de la pile  
 \* \param[in] : Aucun  
 \* \return : L'anneau du dessus de la pile  
 \*/* public Anneau peek()  
 {  
 if (\_top != -1)  
 {  
 return \_tours[\_top];  
 }  
 *//Si la tour est vide, retourne un diamètre plus grand que le maximum d'anneau pour assurer que le déplacment sera perçu comme valide* else  
 {  
 return (new Anneau(\_tours.Length + 1));  
 }  
 }  
  
   
 */\*  
 \* \brief : Vérifie si la pile est vide  
 \* \param[in] : Aucun  
 \* \return : Un bool représentant si la pile est vide  
 \*/* public bool isEmpty()  
 {  
 return (\_top == -1);  
 }  
   
 */\*  
 \* \brief : Vérifie si la pile est pleine  
 \* \param[in] : Aucun  
 \* \return : Un bool représentant si la pile est pleine  
 \*/* public bool isFull()  
 {  
 return (\_top == \_nbAnneau - 1);  
 }  
  
 */\*  
 \* \brief : Vider la pile  
 \* \param[in] : Aucun  
 \* \return : Aucun  
 \*/* public void clear()  
 {  
 while (!isEmpty())  
 {  
 pop();  
 }  
 }  
   
 */\*  
 \* \brief : Afficher la tour  
 \* \param[in] : Aucun  
 \* \return : String représentant la tour avec ses anneaux  
 \*/* public override string ToString()  
 {  
 string affichage = \_lettreTour + ": ";  
  
 *//Variable représentant la différence entre le top dela pile et le maximum d'anneaux* int dif = \_tours.Length - (\_top + 1);  
  
 *//Affichage des anneaux sur la tour* for (int i = 0; i < (\_top + 1); i++)  
 {  
 affichage += \_tours[i].ToString();  
 }  
  
 *//Affichage des espaces libres de la tour* for (int i = 0; i < dif; i++)  
 {  
 affichage += "-";  
 }  
   
   
 return affichage;  
 }  
 }  
}

***CLASSE TOURSHANOI:***

using System;  
using System.Threading.Tasks.Dataflow;  
  
namespace ProjectTourHanoi  
{  
 public class ToursHanoi  
 {  
 private Tour[] \_tours = new Tour[3];  
 private int \_nbAnneau;  
   
 */\*  
 \* \brief : Constructeur ToursHanoi surchargé   
 \* \param[in] : Un int qui représente le nombre d'anneaux  
 \*/* public ToursHanoi(int nbAnneau)  
 {  
 \_nbAnneau = nbAnneau;  
 \_tours[0] = new Tour('A',nbAnneau,nbAnneau);  
 \_tours[1] = new Tour('B',0,nbAnneau);  
 \_tours[2] = new Tour('C',0,nbAnneau);  
 reinitialiser();  
 }  
  
   
 */\*  
 \* \brief : Réinitialiser le jeu  
 \* \param[in] : Aucun  
 \* \return : Aucun  
 \*/* public void reinitialiser()  
 {  
 *//Vide les tours* \_tours[0].clear();  
 \_tours[1].clear();  
 \_tours[2].clear();  
  
 *//Ajout des anneaux sur la tour A* for (int i = \_nbAnneau; i > 0; i--)  
 {  
 \_tours[0].push(new Anneau(i));  
 }  
 }  
  
   
 */\*  
 \* \brief : Déplacer un anneau  
 \* \param[in] : int représentant l'indice de la tour de départ  
 \* \param[in] : int représentant l'indice de la tour de fin  
 \* \return : bool représentant si le déplacment est valide  
 \*/* public bool deplacer(int de, int vers)  
 {  
 Anneau nb = \_tours[de].peek();  
  
 *//Vérifie si l'anneau a déplacer est plus petit* if (nb.Diametre < \_tours[vers].peek().Diametre)  
 {  
 *//Déplacement de l'anneau* \_tours[de].pop();  
 \_tours[vers].push(nb);  
   
 *//Affichage du déplacement* Console.WriteLine("L'anneau de diamètre " + nb + " est déplacé de la tour " + de + " vers la tour " + vers);  
 return true;  
 }  
 else  
 {  
 *//Affichage du message d'erreur* Console.WriteLine("Déplacement illégal");  
 Console.WriteLine("Veuillez recommencer la sélection des tours");  
 return false;  
 }  
  
 }  
  
   
 */\*  
 \* \brief : Transformer les lettres des tour en indice int (a = 0, b = 1, c = 2)  
 \* \param[in] : Aucun  
 \* \return : int représentant l'indice de la tour  
 \*/* public int transform()  
 {  
 int nb = -1;  
 bool fin = false;  
 string de;  
  
 *//Boucle qui prend fin lors d'un résultat valide* while (!fin)  
 {  
 *//Récupération de la lettre de la tour* de = Console.ReadLine();  
   
 *//Si la tour est a, retourne 0 et sort de la boucle* if (de.Equals("a") || de.Equals("A"))  
 {  
 nb = 0;  
 fin = true;  
 }  
  
 *//Si la tour est b, retourne 1 et sort de la boucle* else if (de.Equals("b") || de.Equals("B"))  
 {  
 nb = 1;  
 fin = true;  
 }  
  
 *//Si la tour est c, retourne 2 et sort de la boucle* else if (de.Equals("c") || de.Equals("C"))  
 {  
 nb = 2;  
 fin = true;  
 }  
  
 *//Si la tour est non-valide, affiche le message d'erreur et reste dans la boucle* else  
 {  
 Console.WriteLine("Choix de tour non-valide");  
 }  
 }  
  
 return nb;  
 }  
  
   
 */\*  
 \* \brief : Résolution du jeu  
 \* \param[in] : Aucun  
 \* \return : Aucun  
 \*/* public void resoudre()  
 {  
 reinitialiser();  
 deplacerAuto(\_nbAnneau,0,1,2);  
 }  
   
 */\*  
 \* \brief : Déplacement automatique des anneaux par récursivité  
 \* \param[in] : int représentant le disque à déplacer  
 \* \param[in] : int représentant l'indice de la tour de départ  
 \* \param[in] : int représentant l'indice de la tour intermédiare  
 \* \param[in] : int représentant l'indice de la tour de fin  
 \* \return : Aucun  
 \*/* private void deplacerAuto(int disque,int de, int inter, int vers)  
 {  
 if (disque == 1)  
 {  
 deplacer(de, vers);  
 }  
  
 else  
 {  
 deplacerAuto(disque-1,de,vers,inter);  
 deplacer(de, vers);  
 deplacerAuto(disque - 1,inter,de,vers);  
 }  
 }  
   
 */\*  
 \* \brief : Afficher le jeu  
 \* \param[in] : Aucun  
 \* \return : String représentant le jeu avec ses tours et ses anneaux  
 \*/* public override string ToString()  
 {  
 string affiche = "";  
 for (int i = 0; i < 3; i++)  
 {  
 affiche += \_tours[i]+"\n";  
 }  
 return affiche;  
 }  
 }  
}

***CLASSE PROGRAM:***

using System;  
  
namespace ProjectTourHanoi  
{  
 class Program  
 {  
 private static ToursHanoi jeu;  
   
 static void Main(string[] args)  
 {  
 string choix;  
 bool fin = false;  
   
 jeu = new ToursHanoi(3);  
  
 while (!fin)  
 {  
 *//Affichage du menu* Console.WriteLine("\nTours:");  
 Console.WriteLine(jeu);  
 Console.WriteLine("MENU");  
 Console.WriteLine("1: Déterminer le nombre d'anneaux (3 par défaut)");  
 Console.WriteLine("2: Réinitialiser les tours");  
 Console.WriteLine("3: Jouer un coup");  
 Console.WriteLine("4: Montrer la solution");  
 Console.WriteLine("5: Quitter");  
 Console.WriteLine("Faites votre choix et appuyer sur ENTER");  
   
 *//Récupération du choix de l'utilisateur* choix = Console.ReadLine();  
  
 *//Appel de la fonction selon le choix* switch (choix)  
 {  
 case "1":  
 choix1();  
 break;  
   
 case "2":  
 jeu.reinitialiser();  
 break;  
   
 case "3":  
 choix3();  
 break;  
   
 *//Résolution du jeu* case "4":  
 jeu.resoudre();  
 break;  
   
 *//Quitte la boucle* case "5":  
 fin = true;  
 break;  
   
 *//Message d'erreur si choix non-valide* default:  
 Console.WriteLine("Choix non-valide");  
 break;  
 }  
   
 }  
 }  
  
 */\*  
 \* \brief : Déterminer le nombre d'anneaux (3 par défaut)  
 \* \param[in] : Aucun  
 \* \return : Aucun  
 \*/* static void choix1()  
 {  
 int nb;*//Variable du nombre d'anneaux* bool fin = false;*//Variable pour la boucle  
  
 //Boucle jusqu'à ce qu'un résultat valide soit entré* while (!fin)  
 {  
 *//Récupération du nombre d'anneau* Console.WriteLine("Veuiller entrer un nombre d'anneau entre 1 et 9");  
 nb = Convert.ToInt16(Console.ReadLine());  
  
 *//Validation du nombre entré* if (nb < 10 && nb > 0)  
 {  
 *//Création du nouveau jeu avec le nombre d'anneaux* jeu = new ToursHanoi(nb);  
   
 *//Termine la boucle* fin = true;  
 }  
   
 *//Si le nombre est , retour au menu* else if (nb == 0)  
 {  
 Console.WriteLine("Retour au menu principal");  
 fin = true;  
 }  
   
 *//Si le nombre est invalide, affichage du message d'erreur* else  
 {  
 Console.WriteLine("Choix non-valide");  
 }  
 }  
 }  
   
 */\*  
 \* \brief : Jouer un coup  
 \* \param[in] : Aucun  
 \* \return : Aucun  
 \*/* static void choix3()  
 {  
 bool estfin = false;  
   
 *//Boucle qui prend fin si le déplacement est valide* while (!estfin)  
 {  
 *//Récupération des tours* Console.WriteLine("De quelle tour prenez-vous l'anneau? (A - B - C)");  
 int \_base = jeu.transform();  
   
 Console.WriteLine("Vers quelle tour envoyez-vous l'anneau? (A - B - C)");  
 int fin = jeu.transform();  
  
 *//Déplacement de l'anneau* estfin = jeu.deplacer(\_base, fin);   
 }  
   
 }  
  
   
 }  
}

**Distribution des tâches dans l’équipe**

|  |  |
| --- | --- |
| **Tâche** | **Responsable** |
| Ébauche de classe Anneau | Alexis Young |
| Ébauche de classe Tour | Julien Boisvert |
| Ébauche de classe ToursHanoi | Alexis Young |
| Classe Programme | Julien Boisvert et Alexis Young |
| Révision des classes | Julien Boisvert et Alexis Young |
| Agencement et finalisation des classes | Julien Boisvert et Alexis Young |
| Documentation et rapport final | Julien Boisvert |

**Guide utilisateur**

**MENU :**

**Figure 1**

**Une image contenant texte

Description générée automatiquement**

Le premier menu offre les options de déterminer le nombre d’anneaux, de réinitialiser les tours, de jouer un coup, de montrer la solution ou de quitter. Les tours et l’emplacement actuel des anneaux – il y en a 3 par défaut – sont également affichés dès le premier menu (**Figure 1**). L’option de quitter met fin au programme.

**CHOIX DU NOMBRE D’ANNEAUX :**

**Figure 2.1**

**Une image contenant texte

Description générée automatiquement**

**Figure 2.2**

**Une image contenant texte

Description générée automatiquement**

En choisissant l’option de déterminer le nombre d’anneaux, il est possible de choisir un chiffre entre 1 et 9 (**Figure 2,1)**. L’affichage des tours est alors actualisé avec le nombre d’anneaux choisi (**Figure 2.2**).

**JOUER UN COUP :**

**Figure 3.1**

**Une image contenant texte

Description générée automatiquement**

**Figure 3.2**

**Une image contenant texte

Description générée automatiquement**

En choisissant l’option de jouer un coup, il est possible de choisir la tour d’origine et la tour d’arrivée (**Figure 3.1**), ce qui sélectionne automatiquement l’anneau du dessus de la tour d’origine. Si cet anneau est plus grand que celui sur le dessus de la tour d’arrivée, il est déplacé et un message confirme le déplacement effectué. Sinon, un message d’erreur indiquant « Déplacement illégal » apparaît (**Figure 3.2**). Dans tous les cas, l’affichage de l’emplacement des anneaux sur les tours est ensuite actualisé.

**RÉINITIALISER LE JEU :**

**Figure 4.1**

**Une image contenant texte

Description générée automatiquement**

**RÉSOLUTION AUTOMATIQUE DU JEU :**

**Figure 4.2**

**Une image contenant texte

Description générée automatiquement**

**Une image contenant texte

Description générée automatiquement**

En choisissant l’option de montrer la solution, le jeu est d’abord réinitialisé (**Figure 4.1**). Le programme utilise alors un jeu vierge au nombre d’anneaux donné et l’algorithme affiche la solution tout en l’appliquant en même temps. Chaque étape est affichée tandis que les déplacements se font automatiquement, et à la fin l’état des tours du jeu est affiché (**Figure 4.2**).

**QUITTER L’APPLICATION**

**Une image contenant texte

Description générée automatiquement**

**Stratégies algorithmiques :**

**Transform :**

Pour récupérer les indices des tours lorsque l’utilisateur sélectionne l’une d’entre elles avec la lettre qui lui est associée, nous avons créé la méthode tranform. Celle-ci récupère la lettre de la tour entrée par l’utilisateur, pour ensuite la comparer à l’aide de conditions if permettant de déterminer l’indice associé. Ainsi, si la lettre entrée est A, l’indice est 0, 1 pour B ou encore 2 pour C. Si la valeur entrée est autre que celle correspondante, un message d’erreur s'affiche et la méthode recommence grâce à une boucle while prenant fin lorsqu’un choix de tour valide est entré par l’utilisateur.

**Déplacement** :

En ce qui concerne le déplacement, la méthode utilise les différentes fonctions de la classe Tour pour l’effectuer. Nous commençons par récupérer les deux tours où l’on doit déplacer l’anneau, tout en vérifiant leur validité avec la méthode transform de la classe ToursHanoi. Par la suite, la méthode valide que l’anneau à déplacer est inférieur à celui se trouvant déjà sur la tour où l’on envoie celui-ci, grâce à une condition if. Si la condition est respectée, la tour de provenance appelle la fonction pop() pour enlever l’anneau de la tour puis la méthode push() pour ajouter l’anneau sur la tour agissant de destination.

**Afficher le jeu :**

Pour afficher les tours avec les anneaux se trouvant sur celle-ci, le programme fait appel à la méthode ToSring() de jeu. Celle-ci est une boucle qui affiche chaque tour se trouvant dans le tableau de Tour[]. Ensuite le ToString de Tour est constitué de deux boucles for. La première permet d’afficher les anneaux se trouvant sur la tour et prend fin lorsque le dessus de la pile est atteint. La seconde boucle permet d’afficher les espaces libres de la tour par des traits d’union. Elle représente ainsi la différence entre le dessus et la capacité maximale de la pile et prend donc fin lorsque celle-ci est atteinte.

**Résolution automatique :**

Pour commencer, la méthode résoudre appel la fonction réinitialiser permettant de remettre les anneaux sur la tour A. Elle appelle ensuite la méthode déplacement automatique pour résoudre le jeu. Celle-ci utilise la récursivité multiple et directe. En effet, si l’anneau est différent de 1, la méthode s’appelle elle-même avec un disque plus petit. Ensuite, elle effectue un déplacement simple, pour ensuite se rappeler à nouveau avec un disque plus petit, mais avec la tour de départ et celle intermédiaire interchangée.

**Problèmes et difficultés :**

Nous avons rencontré deux difficultés réelles. Pour commencer, bien que vu en classe, le fonctionnement des piles ne nous était pas extrêmement familier, il a donc été plus laborieux de trouver une façon intéressante d’afficher les espaces vides sur les tours. En effet, alors que pour les anneaux la boucle utilise le dessus de la pile comme contrainte, les espaces vides doivent varier selon les anneaux sur la tour. Ainsi, bien que notre méthode finale est adéquate, elle a pris un bon nombre de temps pour y arriver. Ensuite, le déplacement automatique représentait un bon défi, effectivement, bien que la base de l’algorithme était fournie, nous devions comprendre son fonctionnement pour réussir à trouver une façon d’effectuer les véritables déplacements sur nos tours.