420-W31-SF

ALGORITHMIQUE AVANCEE

TRAVAIL PRATIQUE #3

MASTERMIND

OBJECTIFS PEDAGOGIQUES

Ce travail vise à développer les capacités suivantes :

- la compréhension et l'application des concepts de base de la programmation orientée objet en C++;
- l'utilisation adéquate et la programmation d'une liste et d'un itérateur;
- la programmation d'algorithmes complexes;
- la clarté du code (un code clair n'a pas besoin de commentaires);
- l'application des standards de programmation;
- la programmation des tests unitaires.

MANDAT

Dans un premier temps, vous devez programmer en C++, en suivant la méthodologie TDD, les classes `ListeDouble` et `Iterateur` qui implémentent respectivement les interfaces `ListeBase` et `IterateurBase`. La liste double est votre liste développée en exercice à laquelle vous ajouterez un chaînage « double », c'est-à-dire qu'elle pourra être parcourue dans les 2 sens. Ce changement impacte également le Nœud car celui-devra dorénavant connaître son précédent.

Dans un deuxième temps, toujours en suivant la méthodologie TDD, vous devez compléter la classe Mastermind qui permet à l'ordinateur de deviner la combinaison d'une partie au Mastermind selon les indices fournis par l'utilisateur.

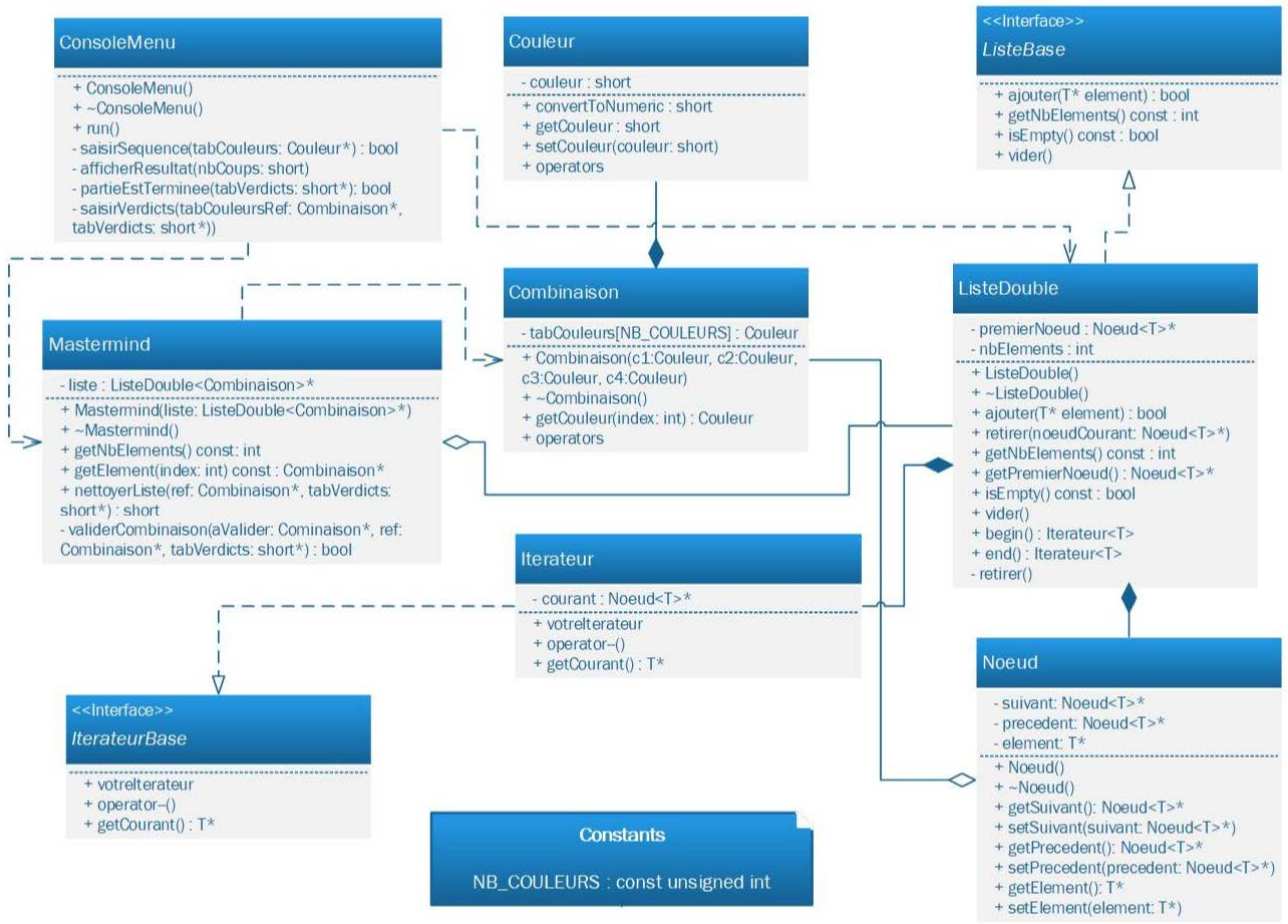
Afin d'apprendre à travailler en équipe de la même façon qu'en entreprise, vous devrez utiliser Git. Cela permettra aussi de s'assurer d'une participation équivalente des 2 coéquipiers.

Pour réaliser ce travail, vous devez compléter la solution fournie avec l'énoncé de ce travail. Les interfaces *ListeBase* et *IterateurBase* ne peuvent pas être modifiées.

Vous devrez commencer par modifier votre Liste pour :

- qu'elle devienne doublement chainée
    - ceci implique aussi une modification de la classe Nœud
- qu'elle soit un template de classe
- qu'elle utilise votre itérateur
    - Toutes les opérations de « positionnement » dans la liste devront utiliser un itérateur (exemple : se déplacer pour trouver le point d'insertion)
    - modifier la méthode retirer() de la liste afin qu'elle supprime un nœud plutôt qu'un contenu

DIAGRAMME DE CLASSES



SPECIFICATIONS DE L'APPLICATION

DESCRIPTION GENERALE

Le Mastermind est un jeu de déductions où un joueur doit analyser des combinaisons de couleurs possibles en fonction de contraintes qui lui sont soumises. Voici les règlements de la version du jeu Mastermind que nous suivrons pour ce travail pratique.

Le jeu Mastermind, mode débutant :

Le déroulement du jeu est simple. Avant de démarrer la partie, le maître du jeu doit choisir une combinaison secrète de 4 couleurs (parmi 8 proposées) et dévoiler de l'information sur cette dernière à chaque tour.

Le joueur doit essayer de deviner la combinaison de 4 couleurs que le maître du jeu a choisie. Pour ce faire, le joueur doit proposer au maître du jeu une combinaison de 4 couleurs qu'il a lui-même choisie.

Le maître du jeu doit ensuite dévoiler des indices sur la combinaison secrète en fonction de la combinaison proposée par le joueur.

Les indices suivants sont délivrés couleur par couleur à partir de la combinaison proposée par le joueur :

1) La couleur est présente dans la combinaison secrète et est à la bonne place dans la combinaison proposée;

2) La couleur est présente dans la combinaison secrète mais n'est pas à la bonne place dans la combinaison proposée;

3) La couleur n'est pas présente dans la combinaison secrète.

Le joueur propose une nouvelle combinaison en fonction des indices dévoilés. Le joueur dispose de 8 essais pour trouver la combinaison secrète.

CRITERES D'EVALUATION

Éléments	Pondération
Réalisation de la classe « Mastermind » et operator == de la classe Combinaison	20%
Réalisation des classes « ListeDouble » et « Iterateur » à partir des interfaces « ListeBase » et « IterateurBase »	35%
Conception de tests unitaires de « Mastermind » • Qualité des tests, pertinence et couverture de code	10%
Conception de tests unitaires de « ListeDouble », incluant l'itérateur • Qualité des tests, pertinence et couverture de code	15%
Utilisation appropriée de Git • Participation régulière et équilibrée des 2 membres de l'équipe • Un « push » par fonctionnalité	5%
Qualité du code (tests et application) • Pas de répétition • Noms significatifs (variables, classes, fonctions, etc.) • Respects d'un standard de nomenclature • Simplicité, découpage et séparation des responsabilités	15%

Conformément à la politique concernant la qualité du français écrit, le travail est sujet à une pénalité pouvant aller jusqu'à 20 % du total des points attribuables. Se référer à la section « Évaluation » du plan de cours pour plus de détails sur l'évaluation des travaux.

Si le professeur le juge à propos, tout étudiant(e) pourra être convoqué(e) à une rencontre d'évaluation pour vérifier son degré d'acquisition des connaissances et d'appréhension de la solution proposée.

SPECIFICATIONS DE REMISE

Sur LEA, vous devez remettre un fichier ZIP contenant :

- votre projet C++ épuré
- lien GitHub ou GitLab avec accès pour l'utilisateur Idan12

Date limite de remise : Lundi le 7 décembre 2020 à 8h.

Pondération du travail : 30% de la note finale

Ce travail est en équipe de deux.

Si le code du travail ne compile pas ou ne s'exécute pas, il ne sera pas corrigé.