Département de génie informatique et logiciel

POLYTECHNIQUE MONTRÉAL

LE GÉNIE EN PREMIÈRE CLASSE

INF1010 – Programmation orientée-objet
TP1

Aut 2022

Objectifs

Permettre à l'étudiant de se familiariser avec les concepts de classes, les pointeurs intelligents, de composition et d'agrégation.

Remise du travail

- Date : **22 septembre 2022 à 23h30**
- **Une note de 0 sera attribuée aux équipes qui remettent leur travail en retard.**
- Remettre **tous** les fichiers directement sous une archive **.zip** (sans dossier à l'intérieur et sans autres fichiers).
- ✂**Toute archive autre que zip ou remise ne respectant pas ces consignes sera refusée et se méritera une note de 0. Attention, une archive .7z ne compte pas comme une archive zip.** ✂
- Respectez le format de la remise! Nom de l'archive zip : **`matricule1_matricule2_groupe.zip`**
  Exemple : 1234567_1234568_1.zip

Références

- Notes de cours sur Moodle

Directives

- Les travaux s'effectuent obligatoirement en équipe de deux personnes faisant partie du même groupe.
- Les entêtes de fichiers sont obligatoires.
- Les fonctions que vous décidez d'ajouter au programme doivent être documentées.

Conseils

- Lisez les conseils, l'aperçu et les spécifications avant de commencer!
- Ayez lu vos notes de cours!
- Si vous avez des difficultés au cours du TP, rappelez-vous que de nombreux problèmes demandés sont assez couramment rencontrés en C++. Consulter la documentation sur cppreference, les notes de cours, et les forums sur Google peuvent vous donner de bonnes pistes de résolution!
- Veuillez utiliser la version v17 de C++.

- Si votre programme ne compile pas, veuillez mettre en commentaires les instructions qui ne compilent pas.

✂Tout cas de plagiat sera automatiquement reporté au Comité d'examen de fraude (CEF) ✂

Spécifications générales

✂ Le non-respect des spécifications générales entraînera des pénalités. ✂

- Tout warning à la compilation sera pénalisé. Si vous utilisez Visual Studio de Microsoft, vous devez activer l'option **/W4**. Sur Visual Studio, **assurez-vous de compiler avec x64**, certains warnings pourraient ne pas s'afficher sinon.
- Sur Visual Studio, vous devez activer C++17.
- **L'inclusion des fichiers d'en-tête (`.h`) doit être sensible aux majuscules et minuscules**. Malheureusement, sur Windows les fichiers ne sont pas sensibles aux majuscules et minuscule, donc des erreurs d'inclusions pourraient passer inaperçues. Relisez-vous bien.
- Toutes les classes, fonctions et entêtes de fichiers doivent être documentées. Suivez l'exemple de la documentation déjà présente dans le TP.
- Toutes les méthodes doivent être définies dans le fichier d'implémentation (`.cpp`) **dans le même ordre** que leur déclaration dans le fichier d'en-tête (`.h`). Le non-respect de cette règle entraînera une pénalité au niveau du style.
- Utilisez le plus possible la liste d'initialisation des constructeurs. L'utilisation du corps des constructeurs à la place de la liste d'initialisation entraînera une pénalité de style. L'ordre des variables (attributs) dans la liste d'initialisation doit être la même que celle dans la liste des attributs de la classe dans le fichier d'en-tête.
- Suivez le guide de codage sur Moodle.
- Modifications/ajouts au guide de codage à respecter :
    - Vous pouvez utiliser le style d'accolades que vous désirez, **tant que vous êtes uniformes**. Sachez cependant qu'il est généralement attendu d'un(e) développeur(se) suivre la convention établie par le projet, ou la convention établie par la langue (si elle existe) en créant un nouveau projet. Dans le cas du TP fourni, les accolades ouvrantes sont sur leur propre ligne (style Allman).
    - Mettez un espace avant la parenthèse ouvrante des énoncés de contrôle comme if, for, while et switch, mais pas avant les parenthèses d'un appel de fonction. Ne mettez jamais d'espace tout juste après une parenthèse ouvrante ou tout juste avant une parenthèse fermante.
    - Le deux-points (`:`) et le signe égal (`=`) devraient être entourés d'espaces, sauf dans le cas des étiquettes de `case` et les spécificateurs d'accès (`private`, `protected` et `public`) qui ne devraient pas avoir d'espace avant le deux-points. Un espace

devrait toujours suivre les éléments de ponctuation comme la virgule (`,`), le deux-points (`:`) et le point-virgule (`;`).
- N'utilisez pas `NULL` ou `0` pour les pointeurs, mais bien `nullptr`. Dans le cas de test de `nullptr` avec un pointeur, soyez explicite : préférez `if (p != nullptr)` à `if (p)`.
- N'utilisez pas de `else` après un `return`.

Mise en contexte

Ce travail consiste à implémenter les classes d'une équipe de sport avec des joueurs. Dans les tps suivants, on implémentera des matchs dans un tournoi.

Aperçu des classes du projet

- ***Date***: Classe qui représente une date ;
- ***Joueur*** : Classe qui représente un joueur d'une équipe;
- ***Équipe*** : Classe qui représente l'ensemble des joueurs de l'équipe;

Travail à réaliser

Implémenter les classes en complétant les TODO des fichiers .cpp fournis.

Classe Date

La classe représente une date ayant les attributs suivants : Année, Mois et jour

Classe Joueur

Classe qui représente un joueur d'une équipe.

Cette classe contient les attributs suivants : le nom du joueur, le nombre de match, le nombre de buts marqués, et le nombre d'assist.
Le constructeur par défaut à la valeur « inconnu » à son nom, le nombre de matchs, les buts et assits sont à zéro.
Le constructeur par paramètres initialise les attributs en fonction des paramètres.
Il faut implémenter les méthodes d'accès et de modification.
La méthode afficher() affiche à l'écran la valeur de tous les attributs.

Classe Équipe

Classe qui représente l'ensemble des joueurs de l'équipe.

Cette classe contient les attributs suivants :
- Un nom (string).
- Un nombre de victoire (entier).
- La capacité du tableau intelligent de pointeurs à un joueur.

- Le nombre de joueurs du tableau.
- Un tableau intelligent de pointeurs à des joueurs.
- La date de création de l'équipe

Il faut implémenter toutes les méthodes de la classe en particulier

- Constructeur par défaut à les valeurs des attributs : nom est « inconnue », le nombre de joueurs et le nombre de joueurs à zéro, la date de création est 2022, 01, 01, et le constructeur par paramètres.
- getNombreVictoire() retourne le nombre de victoires de l'équipe.
- getNombresJoueurs() retourne le nombre de joueurs dans l'équipe
- getJoueur() retourne le joueur dont le nom est passé en paramètre. Retourne un null pointer si le joueur n'existe pas dans l'équipe.
- ajouterJoueur() qui ajoute un joueur dans le tableau intelligent si le joueur n'existe pas. Lors de l'ajout de joueur dans le tableau, si la capacité du tableau est plus petite que le nombre de joueurs, alors on double la capacité du tableau.
- supprimerJoueur() qui supprime un joueur s'il existe dans l'équipe. Renvoie true si le joueur a été supprimé correctement, false sinon.
- ajouterVictoire() incrément le nombre de victoires de l'équipe.
- afficher () affiche le nom de l'équipe, les joueurs et le nombre de victoires

main.cpp

Le programme principal appelle la fonction testAll() qui roule une série de tests. L'affichage devrait correspondre à ce qui suit :

```
Tests pour Creation/Modifications des joueurs:
--------
Test  1: Getters de la classe Joueur                                          : OK
Test  2: Setters de la classe Joueur                                          : OK
--------
Total pour la section: 5/5  arrondi au dixieme

Tests pour Creation/Modification des equipes:
--------
Test  3: Constructeurs et Getters du nom de l equipe et du nombre de victoires          : OK
Test  4: Fonction ajouterVictoire() et obtenir le bon nombre de victoires               : OK
Test  5: Fonction ajouterJoueur() ajoute des joueurs                                    : OK
Test  6: AjouterJoueur() lorsque le nombre de joueurs > capactite du tableau            : OK
Test  7: Fonction supprimerJoueur() si le joueur n existe pas                           : OK
Test  8: Fonction supprimerJoueur() et get le bon nombre de joueurs apres suppression   : OK
Test  9: Test de getJoueur() et mix des autres fonctions                                : OK
--------
Total pour la section: 5/5  arrondi au dixieme

Total pour tous les tests: 10/10  arrondi au dixieme
```

Correction

La correction du TP se fait sur 20 points :

- [3 pt] Compilation du programme sans avertissements.
- [4 pt] Exécution du programme.
- [10 pt] Comportement exact de l'application
- [2 pt] Qualité et documentation du code.
- [1 pt] Absence de fuites de mémoire.

Relisez bien les spécifications générales et les consignes de remise au début du TP avant de remettre votre travail!