Hiver 2017

Programmation avancée en C++

GIF-1003

Thierry EUDE, Ph.D

Travail individuel

à rendre avant
jeudi 23 février 2017 14h
(Voir modalités de remise à la fin de l'énoncé)

Tout travail remis constitue une contribution originale et distincte de travaux remis par d'autres. Le plagiat est strictement défendu. Tout travail plagié obtiendra la note 0. De plus, les étudiants ou étudiantes ayant collaboré au plagiat seront soumis aux sanctions normalement prévues à cet effet par les règlements de l'Université.

Par ailleurs, vu l'importance des communications écrites dans le domaine de la programmation, il sera tenu compte autant de la présentation que de la qualité du français et ce, dans une limite de 10% des points accordés.

Travail Pratique #2
Développement de classes

UNIVERSITÉ LAVAL
Faculté des sciences et de génie
Département d'informatique
et de génie logiciel

Notre objectif final est de construire un outil afin de gérer l'annuaire téléphonique des membres d'un club de hockey. La série des travaux pratiques devrait permettre de tendre vers cet objectif. Chaque travail pratique constitue donc une des étapes de la construction de cet outil.

But du travail

- Apprivoiser le processus de développement d'une classe dans un environnement d'implémentation structuré,
- Respecter des normes de programmation et de documentation.

Classe Personne

Cette classe permet de modéliser des personnes Vous devez implanter les spécifications qui suivent dans les fichiers `Personne.h` et `Personne.cpp`.

Attributs de la classe

```
string m_nom;
string m_prenom;
```

Le nom et le prénom de la personne. Doivent être non vide et ne comporter que des lettres. Cette validation pourra être faite par l'intermédiaire de la fonction `util :validerFormatNom` (voir plus loin dans fonctions de validations)

```
util::Date m_dateNaissance;
```

La date de naissance de la personne.
La validité de la date doit être établie par la méthode statique `Date::validerDate(…). (`

```
string m_telephone;
```

Le numéro de téléphone de la personne.
Le numéro de téléphone doit être dans un format valide tel que déterminé par la fonction `util::validerTelephone` développée au TP1.

Méthodes de la classe

Constructeur de la classe. On construit un objet `Personne` à partir de valeurs passées en paramètre.

```
Personne (  const std::string& p_nom,
            const std::string& p_prenom,
            const util::Date& p_dateNaissance,
            const std::string& p_telephone);
```

Méthodes d'accès aux attributs de la classe :

```
reqNom()
reqPrenom()
reqDateNaissance()
reqTelephone()
```

Méthode qui permet d'assigner un nouveau numéro de téléphone à la personne :

```
void asgTelephone(const std ::string& p_telephone);
```

Le numéro de téléphone doit être dans un format valide selon la fonction `util::validerTelephone`.

```
std::string reqPersonneFormate ()
```

Méthode qui retourne dans un objet `std::string` les informations correspondant à une personne formatées sous le format suivant :

```
Nom               : Louis
Prenom            : Jean
Date de naissance : Samedi le 12 mai 1979
Telephone         : 418 656-2131
```
Utilisez la classe ostringstream du standard pour formater les informations sur la personne.

Opérateur de comparaison d'égalité. La comparaison se fait sur la base des nom, prénom et date de naissance.
```
bool operator==(const Personne& unePersonne)
```

Fonction de validation

Il s'agit de compléter le module de fonctions de validation développé dans le TP1.
Vous devez commencer par insérer l'ensemble des fonctions dans le namespace **`util`**.
Dans les fichiers `validationFormat.h` et `validationFormat.cpp`, vous devez implanter une nouvelle fonction de validation dont les spécifications sont les suivantes.
```
bool validerFormatNom(const std ::string& p_nom)
```
Permet de valider le format d'un nom; soit retourne vrai s'il n'est pas vide et ne comporte que des lettres (majuscules ou minuscules), faux sinon.

Documentation

La classe `Personne` ainsi que toutes les méthodes et fonctions devront être correctement documentées pour pouvoir générer une documentation complète à l'aide de l'extracteur DOXYGEN. Des précisions sont fournies sur le site Web pour vous permettre de l'utiliser (syntaxe et balises à respecter etc.).
Vous devez respecter les normes de programmation adoptée pour le cours. Ces normes de programmation sont à votre disposition dans la section "Notes de cours / Présentations utilisées en cours / Normes de programmation en C++ " du site Web du cours.
Aussi, comme indiqué dans ces normes, vous devez définir un espace de nom (namespace) qui inclura le code spécifique au développement de l'outil de gestion d'annuaire (pour le moment il n'y aura que la classe Personne). Il devra porter le nom suivant : `tp` (en minuscules).

Utilisation

Après avoir implanté la classe comme demandé, vous devez écrire un programme interactif minimaliste. Le but est simple, il s'agit simplement d'obtenir interactivement avec l'usager, les données nécessaires pour créer une "*`Personne`*" puis de modifier son numéro de téléphone.
Rappelons qu'un objet *`Personne`* ne peut être construit qu'avec des valeurs valides. C'est la responsabilité du programme principal qui l'utilise de s'assurer que ces valeurs sont valides.
Les critères de validité ont été énoncés dans la description des attributs des classes et dans l'énoncé des fonctions de validation.
Une fois toutes les données validées, vous créez un objet *`Personne`* et vous demandez à l'objet créé de retourner une chaîne formatée (*`reqPersonneFormate()`*) pour pouvoir l'afficher dans la console. Vous demandez ensuite à l'usager de saisir un nouveau numéro de téléphone que vous assignez à la personne précédemment construite. Un deuxième affichage permet de vérifier que la modification a bien été effective. Et c'est tout pour l'instant…
Voir l'exemple d'exécution fourni. Cependant, respectez strictement l'ordre et les données saisies (pensez au correcteur qui va devoir évaluer votre travail).

Modalités de remise, bien livrable

Le deuxième travail pratique pour le cours GIF-1003 Programmation avancée en C++ est un **travail individuel.** Vous devez remettre le code source dans un espace de travail (workspace) Eclipse mis dans une archive zip, en utilisant le dépôt de l'ENA :
https://sitescours.monportail.ulaval.ca/ena/site/evaluation?idSite=75147&idEvaluation=293749&onglet=boiteDepots
Ce travail est intitulé TP #2. **Aucune remise par courriel n'est acceptée**.

Vous pouvez remettre autant de versions que vous le désirez. Seul le dernier dépôt est conservé (pensez tout de même à numéroter vos versions). **Il est de votre responsabilité de vous assurer de ce que vous avez déposé sur le serveur**.

Attention, vérifiez qu'une fois déplacé et décompressé, votre workspace est toujours fonctionnel (il doit donc être « portable »). Pensez au correcteur ! Sachez qu'il utilisera la machine virtuelle fournie pour le cours.

Très important

Ne pas utiliser « windows » pour faire votre archive ; son outil natif présente des lacunes. Utilisez à la place 7-zip (gratuit, inclus dans la machine virtuelle fournie pour le cours).

Date de remise

Ce travail doit être rendu avant le jeudi 23 février 2017 à 14h00. Pour tout retard non motivé (voir plan de cours; motifs acceptables pour s'absenter à un examen), la note 0 sera attribuée.

Critères d'évaluation

1) Respect des biens livrables
2) Respect des normes de programmation, et de documentation
3) Structures, organisation du code (très important!)
4) Exactitude du code
5) Utilisation des classes, i.e. le programme principal

Particularités du barème

- *Si des pénalités sont appliquées, elles le sont sur l'ensemble des points.*
- *Si un travail comporte ne serait-ce qu'une erreur de compilation, il sera fortement pénalisé, et peut même se voir attribuer la note zéro systématiquement.*
- *Il est très important que votre travail respecte strictement les consignes indiquées dans l'énoncé, en particulier les noms des méthodes, les noms des fichiers et la structure de développement sous Eclipse sous peine de fortes pénalités*

Bon travail