IFT1169 – TRAVAIL PRATIQUE #1 – 20 janvier 2015

« Espace, frontière de l'infini ... »

Mohamed Lokbani

Équipes : le travail est à faire en monôme (une seule personne).
Remise : une seule remise est à effectuer par voie électronique le mardi 03 février 2015, 23h59 au plus tard, sans possibilités de prorogation.
Conseils : n'attendez pas le dernier jour avant la remise pour engager votre travail. Vous n'aurez pas le temps nécessaire pour le réaliser.

But : ce TP a pour but de vous faire pratiquer l'héritage, le polymorphisme et l'utilisation d'un « makefile ».

Énoncé : ce travail est subdivisé sur plusieurs étapes. Les fichiers de chaque étape sont rassemblés dans un répertoire unique. Chaque répertoire doit avoir un fichier « makefile » propre à lui pour compiler l'ensemble des fichiers nécessaires au bon fonctionnement de l'étape en question. L'organisation complète du répertoire à remettre est détaillée dans l'annexe -1-.

1re étape : soit les deux classes « Terre » et « Lune », qui chacune d'elles ne contient que la méthode « affiche », dont la signature est « void affiche() ». Lorsqu'elle est appelée par le programme, cette méthode affiche en sortie :

Classe	Affichage en sortie
Terre	Je suis sur la terre!
Lune	Je suis sur la lune!

- Écrire le contenu de « Terre.h/.cpp » et « Lune.h/.cpp ».
- Écrire le programme « main_01.cpp » qui doit contenir ce qui suit :
  - Une instance « t1 » de Terre, déclarée par valeur, puis un appel à la méthode « affiche » via cette instance.
  - Une instance « t2 » de Terre, déclarée par pointeur, puis un appel à la méthode « affiche » via cette instance.
  - Une instance « l1 » de Lune, déclarée par valeur, puis un appel à la méthode « affiche » via cette instance.
  - Une instance « l2 » de Lune, déclarée par pointeur, puis un appel à la méthode « affiche » via cette instance.
- L'affichage en sortie sera comme suit :

```
Je suis sur la terre!
Je suis sur la terre!
Je suis sur la lune!
Je suis sur la lune!
```

2e étape : vous gardez la même composition des classes. Vous allez apporter cette fois-ci des modifications au fichier « main_01.cpp ». Le nouveau fichier va porter le nom de « main_02.cpp » comme suit :

- Déclarer d'abord le tableau « Terre tab_Terre[50]; ».
- Initialiser les instances « 0 » et « 1 » du tableau « tab_Terre » avec les valeurs « t1 » et « t2 ».
- Remplacer les deux appels à la méthode « affiche » pour les deux instances de « Terre » par un appel à la même méthode, mais à travers le tableau.

Question -1- : est-ce que l'instance « tab_Terre[0] » a la même adresse que l'instance « t0 »? Comment peut-on le prouver?
Question -2- : est-ce que l'instance « tab_Terre[1] » a la même adresse que l'instance « t1 »? Comment peut-on le prouver?
Question -3- : est-il possible d'initialiser les instances « 2 » et « 3 » du tableau avec les instances de « Lune », « l1 » et « l2 »? Pourquoi?
Question -4- : est-il possible de faire appel à la méthode « affiche » pour les les 50 éléments du tableau « tab_Terre » ou uniquement les 4 premiers éléments du tableau qui représentent probablement les deux instances de « Terre » et de « Lune »?

3e étape : nous allons modifier l'organisation des classes. En réalité, comme chacun le sait, la terre et la lune sont des astres.

- Le mot clé « virtual » ne doit pas être utilisé à ce stade de l'exercice.
- Ajouter la classe « Astre.h/cpp ».
- La méthode « affiche » de la classe « Astre » va afficher en sortie ce qui suit :

```
Je suis sur un astre! Oups! Lequel?
```

- Ajuster en conséquence les classes « Terre » et « Lune ».
- Quel est le type du tableau qu'il faudra inclure dans la fonction « main », du programme « main_03.cpp » afin qu'il puisse contenir des instances de « Terre » et de « Lune »? Pourquoi?
- Déclarer un tel tableau « xxx un_tab[50]; » (xxx : représente le type prédéfini dans la précédente question).
- Initialiser les 4 premiers éléments d'un tel tableau avec les instances « t1 », « t2 », « l1 » et « l2 ».
- Appeler, la méthode « affiche » pour les 4 premiers éléments du tableau. Que va-t-on obtenir en sortie? Pourquoi?
- Est-ce que le fait d'ajouter le mot clé « virtual » aura une influence sur l'affichage en sortie? Pourquoi?

4e étape : modifier le code « main_03.cpp » de la précédente étape pour permettre un affichage correct! Vous n'allez pas ajouter de code. Vous allez reprendre exactement les mêmes lignes de code et les ajuster en conséquence. Le nouveau fichier va porter le nom « main_04.cpp ».

5e étape : nous allons réorganiser la hiérarchie des classes en introduisant un niveau intermédiaire afin de généraliser la représentation : la classe « Planete » pour nous permettre d'ajouter la classe « Mars »; la classe « Satellite naturel » pour nous permettre d'ajouter « Phobos » et « Deimos ».

- Ajuster en conséquence les différentes classes.
- Modifier le contenu du fichier « main_04.cpp » comme suit :
  - Le tableau va contenir cette fois-ci uniquement des instances allouées dynamiques.
  - Allouer dans le tableau une instance pour chaque classe : « Astre », « Planete », « SatNat », « Terre », « Mars », « Lune », « Deimos » et « Phobos ».
  - Allouer un nouveau tableau qui va contenir les 3 instances : « Planete », « Terre » et « Mars ».
  - Comparer, au niveau de la classe, les instances des deux tableaux et afficher un message en conséquence.
  - Les deux tableaux ont 3 instances en commun : « Planete », « Terre » et « Mars ».
  - Le fichier « main_04.cpp » va porter le nom « main_05.cpp ».

Fichiers à remettre : vous devez remettre le fichier « tp1H15.zip » tel que décrit en annexe -1-. Par ailleurs, vous devez inclure dans votre remise un rapport décrivant le travail effectué. Les réponses aux questions peuvent être rédigées dans un document à part clairement identifié.

Remise : vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI. La procédure a déjà été décrite dans le TP#0.

Le répertoire de remise sur le serveur est tp01.

Barème : ce TP1 est noté sur 11 points. Les points sont répartis comme suit :

1re étape(1.5) : classes (0.5), main (0.5), makefile (0.5).
2e étape (2.5) : main (1), Q1/Q2 (0.5), Q3 (0.5), Q4 (0.5).
3e étape (2) : classes (1), tab (0.25), affichage (0.5), virtual (0.25).
4e étape (1)
5e étape (2.5) : classe (1), tableau1 (0.5), tableau2 (0.5), comparaison (0.5).
Rapport : 1.5
Bonus (0.5) : makefile à la racine du répertoire qui va faire appel au makefile dans les sous-répertoires.

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ....
- un makefile manquant dans une des étapes (-0.5).
- La non-remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- Un programme qui ne compile pas : 0.
- Un programme qui compile, mais ne réalise pas les choses prévues dans la spécification : 0.
- Les avertissements (warnings) non corrigés : cela dépend de la quantité! À partir de -0.25 et plus.
- Aberration dans le codage : même si tous les chemins mènent à Rome, faites l'effort nécessaire pour éviter de prendre le plus long!

Des questions à propos de ce TP? Une seule adresse : dift1169@iro.umontreal.ca

Pour faciliter le traitement de votre requête, inclure dans le sujet de votre courriel, au moins la chaîne [IFT1169] et une référence au tp01.

Mise à jour
19-01-2015 diffusion de l'énoncé.

Annexe -1-

TP1_H15
- makefile_H15 (bonus)
- Etape_1
  - Terre.h, Terre.cpp, Lune.h, Lune.cpp, main_01.cpp, makefile_1
- Etape_2
  - Terre.h, Terre.cpp, Lune.h, Lune.cpp, main_02.cpp, makefile_2
- Etape_3
  - Astre.h, Astre.cpp, Terre.h, Terre.cpp, Lune.h, Lune.cpp, main_03.cpp, makefile_3
- Etape_4
  - Astre.h, Astre.cpp, Terre.h, Terre.cpp, Lune.h, Lune.cpp, main_04.cpp, makefile_4
- Etape_5
  - Astre.h, Astre.cpp, Mars.h, Mars.cpp, Terre.h, Terre.cpp, Lune.h, Lune.cpp, Phobos.h, Phobos.cpp, Deimos.h, Deimos.cpp, main_05.cpp, makefile_5

En rouge, sont signalés les répertoires; en vert, les fichiers « makefile » et en noir les fichiers « .h/.cpp ».

Il vous est possible d'omettre un fichier (sauf le « makefile », ce dernier est obligatoire), si vous jugez qu'il n'est pas utile. Vous devez mentionner les raisons de ce retrait dans votre rapport.