

**IFT1169 – TRAVAIL PRATIQUE #1 – 18 septembre 2020****« 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 lundi 05 octobre 2020, 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 de « codelite ».

**Énoncé :** ce travail est subdivisé sur plusieurs étapes. Les fichiers de chaque étape sont rassemblés dans un répertoire unique. L'organisation de l'espace de travail à remettre est détaillée dans « l'annexe - 1- ».

**Assurez-vous de n'afficher en sortie que ce qui est demandé.**

**1<sup>re</sup> étape :** soit les deux classes « Terre » et « Lune », 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!
--

**2<sup>e</sup> 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- : l'instance « tab\_Terre[0] » a-t-elle la même adresse que l'instance « t0 »? Comment peut-on le prouver?

Question -2- : l'instance « tab\_Terre[1] » a-t-elle la même adresse que l'instance « t1 »? Comment peut-on le prouver?

Question -3- : est-ce qu'il est possible d'initialiser les instances « 2 » et « 3 » du tableau avec les instances de « Lune », « l1 » et « l2 »? Pourquoi?

Question -4- : est-ce qu'il est possible de faire appel à la méthode « affiche » pour 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 »?

**3<sup>e</sup> 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?

**4<sup>e</sup> 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 ».

**5<sup>e</sup> 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 ».

Codelite : assurez-vous d'avoir activé dans codelite ces options : « Wall », « pedantic » et « c++17 ».

### **Fichiers à remettre :**

« TP1\_A20.workspace » : tout le travail doit-être réalisé dans codelite dans l'espace de travail (« workspace ») « TP1\_A20 ». Chaque étape de ce travail pratique est matérialisée dans un projet. Ainsi l'espace de travail va contenir au final 5 projets.

« TP1\_A20 \_Rapport.pdf » : le rapport doit décrire le travail effectué.

« TP1\_A20 \_QetR.pdf » : les réponses aux différentes questions posées dans ce travail pratique.

« TP1\_A20 \_feuille2route.pdf » : une auto-évaluation du travail réalisé.

Vous devez regrouper l'ensemble de ces fichiers dans « TP1\_A20.zip » et le remettre par l'entremise du système Studium, dans le dépôt « tp01 ».

La correction va se faire sur un poste de la DESI en utilisant le compilateur « g++ » (9.2). Vérifiez que le code que vous avez produit (devant normalement fonctionner correctement chez vous) fonctionne aussi bien sur les ordinateurs de la DESI.

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

1<sup>re</sup> étape(1.5) : classes (1), main (0.5).

2<sup>e</sup> étape (2.5) : main (1), Q1/Q2 (0.5), Q3 (0.5), Q4 (0.5).

3<sup>e</sup> étape (2) : classes (1), tab (0.25), affichage (0.5), virtual (0.25).

4<sup>e</sup> étape (1)

5<sup>e</sup> étape (2.5) : classe (1), tableau1 (0.5), tableau2 (0.5), comparaison (0.5).

Rapport : 1.5

En plus du précédent barème, vous risquez de perdre des points dans les cas suivants ...

- La non-remise électronique (volontaire ou par erreur) est sanctionnée par la note 0.
- La non-remise du fichier « TP1\_A20 \_feuille2route.pdf » : -1.
- 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](mailto: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**

18-09-2020 diffusion de l'énoncé.

---

### **Annexe -1-**

#### TP1\_A20

##### Etape\_1

Terre.h, Terre.cpp, Lune.h, Lune.cpp, main\_01.cpp

##### Etape\_2

Terre.h, Terre.cpp, Lune.h, Lune.cpp, main\_02.cpp

##### Etape\_3

Astre.h, Astre.cpp, Terre.h, Terre.cpp, Lune.h, Lune.cpp, main\_03.cpp

##### Etape\_4

Astre.h, Astre.cpp, Terre.h, Terre.cpp, Lune.h, Lune.cpp, main\_04.cpp

##### 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

En bleu, le nom de l'espace de travail, en rouge, les noms des projets, et en noir les différents fichiers « .h/.cpp ».

Il vous est possible d'omettre un fichier, si vous jugez qu'il n'est pas utile. Vous devez mentionner les raisons de ce retrait dans votre rapport.

### **Rapport -2-**

Vous devez remettre un rapport complet avec ce travail. Ce rapport va expliquer votre démarche. Quelles sont les difficultés rencontrées et comment les avez-vous résolues? Le rapport ne doit pas contenir votre programme! Telle ligne fait telle opération. Les commentaires doivent être intégrés au code.

Le rapport a deux objectifs :

1- aider une personne qui cherche à faire la même chose que vous, à mieux comprendre votre programme. En lisant votre code, bien commenté, et votre rapport, bien rédigé, elle aura une vision globale de votre travail. Elle peut mieux comprendre votre programme pour l'utiliser tel quel ou lui intégrer plus facilement de nouvelles fonctionnalités.

2- aider le correcteur à mieux comprendre votre démarche. Vous avez décidé d'une démarche, à première vue ce choix peut paraître étrange. Votre rapport va permettre de mieux appuyer le pourquoi de ce choix. Ce dernier sera ainsi mieux compris.