

	La programmation orientée objet (POO) en C++		BTS SN-IR
	Nom, Prénom :	Cours	

Quelques programmes simples en langage POO

Application 1 :

Soit le programme ci-dessous devant permettre d'associer un nom de sportif à sa spécialité, un étudiant a déjà écrit un morceau de code, mais celui-ci produit des erreurs à la compilation

1. Saisissez ce morceau de code et observez le résultat obtenu, quels types d'erreurs obtenez-vous ? Les expliquez.
2. Corrigez et complétez ce programme de façon à obtenir le résultat suivant en console :

```
K Mbappé est âgé de 19 ans et le sport qu'il pratique est le Football
```

3. Copiez et commentez votre programme dans votre compte rendu

```
#include <iostream>
using namespace std;
class Joueur
{
    string mNom;
    int mAge;
public:
    string Typejeu;
};
int main()
{
    Joueur p1;
    p1.mNom = "K M";
    p1.mAge = 19;
    p1.Typejeu = "Football";
    return 0;
}
```

Application 2

Créer un programme simple contenant une classe joueur et permettant d'afficher les postes de joueurs d'une équipe de basket ball (utilisation de constructeur)

Poste 1 : **meneur**

Poste 2 : **arrière**

Poste 3 : **ailier**

Poste 4 : **ailier fort**

Poste 5 : **pivot**

Vous choisirez les 5 joueurs que vous voudrez pour votre équipe (nom et numéros)

1. Avant d'écrire votre programme, vous définirez une classe joueur qui aura 3 attributs :

1. Nom du joueur (String)
2. Poste du joueur (String)
3. Numéro du joueur (int)

et 3 méthodes :

- constructeur par défaut : demande à l'utilisateur de saisir le nom, le poste et le numéro
 - constructeur avec 3 paramètres : permet de créer directement un objet joueur en indiquant son nom, poste et numéro
 - accesseur (get) de la variable poste : renvoie le poste du joueur
 - accesseur (get) de la variable nom : renvoie le nom du joueur
 - accesseur (get) de la variable poste : renvoie le numéro du joueur
2. Dans le « main », vous créerez autant d'objets que de joueurs (5 sur le terrain dans une équipe)
 3. Votre programme contiendra un fichier d'en tête joueur.h, un fichier joueur.cpp
 4. Réalisez et tester votre programme, et copiez le dans votre compte rendu, vous le commenterez un maximum
 5. Améliorez votre programme à votre goût pour un affichage « agréable » de l'équipe de basket.

Héritage simple

Questions liées à l'héritage :

1. Qu'est-ce que l'héritage et qu'est-ce qu'une classe mère et une classe fille ?
2. Comment déclare-t-on qu'une classe B dérive d'une classe A ?

3. De quoi hérite la classe fille ?

Application 3 :

Soit le programme ci-dessous :

```
#include <iostream>
using namespace std;
class Personne
{
    public:
        string profession;
        int age;
        void affiche()
        {
            cout << "Ma profession est: " << profession << endl;
            cout << "Mon age est de " << age << " ans" << endl;
            etude();
            travaille();
        }
        void travaille() { cout << "Je peux travailler." << endl; }
        void etude() { cout << "J'ai terminé ma formation." << endl; }
};

class MathsProfesseur : public Personne
{
    public:
        void enseignemaths() { cout << "Je peux enseigner les maths." << endl; }
};

int main()
{
    MathsProfesseur prof1;
    prof1.profession = "Professeur";
    prof1.age = 23;
    prof1.affiche();
    prof1.enseignemaths()
    return 0;
}
```

1. Observez le programme ci-dessous, combien de classes contient-il ? Nommez les.
2. Quelles sont les méthodes de la classe mère, de la classe fille ?

3. Est-ce que les attributs et méthodes de la classe mère sont accessibles par la classe fille ?
4. Saisir ce programme et observez le résultat obtenu en console
5. Déclarez les méthodes « travail() » et « etude() » de la classe mère comme protected, quel est l'avantage à procéder de cette façon ?
6. Représentez graphiquement le diagramme de classes de ce programme
7. Organisez ce programme en créant un fichier d'en tête .h, un fichier source .cpp
8. Enrichir ce programme en créant deux autres classes filles de profession de votre choix héritant de la classe Personne.

Héritage multiple et hiérarchique

L'héritage peut être de plusieurs types :

- Héritage unique, nous l'avons abordé précédemment
 - Héritage multiple
 - Héritage multiniveaux
- etc.

Héritage multiple

Lorsqu'une classe dérivée hérite des propriétés et des comportements de plusieurs classes de base, elle est appelée héritage multiple

Application 4 :

Soit le programme ci-dessous :

```
#include <iostream>
using namespace std;

class USBDevice
{
private:
    int m_id;

public:
    USBDevice(int id) : m_id(id) { }
    int getID() { return m_id; }
};
```

```

class NetworkDevice
{
private:
    int m_id;

public:
    NetworkDevice(int id) : m_id(id){}
    int getID() { return m_id; }
};

class USBWifi: public USBDevice, public NetworkDevice
{
public:
    USBWifi(int usbId, int networkId)
        : USBDevice(usbId), NetworkDevice(networkId) {}
};

int main()
{
    USBWifi f54 (5442, 181742);
    cout<< "Ce type de clé USB sans fil a une fréquence de "<< f54.getID()<<" Hz"<<endl;

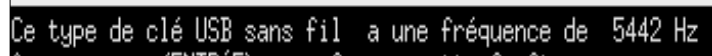
    return 0;
}

```

1. Identifiez et nommez les classes de ce programme
2. La classe USBWifi hérite de quelles classes ? justifiez votre réponse, réaliser le diagramme de classes de ce programme
3. Saisir ce programme et observez les messages du compilateur.

Effectivement lorsque f54.getID () est compilé, le compilateur cherche à savoir si USBWifi contient une fonction nommée getID () et ce n'est pas le cas. Le compilateur recherche ensuite si l'une des classes parentes possède une fonction nommée getID (). Le problème est que f54 contient en réalité DEUX fonctions getID () : une héritée de USBDevice et une héritée de NetworkDevice. Par conséquent, cet appel de fonction est ambigu.

4. Proposez une solution pour contourner cette ambiguïté de façon à produire en console le résultat suivant :



```

Ce type de clé USB sans fil a une fréquence de 5442 Hz

```

Vous pouvez imaginer comment les choses peuvent devenir complexes lorsque votre classe hérite de quatre ou six classes de base, qui héritent des autres classes elles-mêmes. Le potentiel de conflits de noms augmente de façon exponentielle à mesure que vous héritez de classes, et chacun de ces conflits de noms doit être résolu explicitement.

Héritage Hiérarchique

Lorsque plusieurs propriétés d'une classe de base héritent des propriétés et des comportements d'une classe de base, on parle alors d'héritage hiérarchique.

Personne

(Cf : application 3)

Boulangier

Mathsprofesseur

Héritages multiniveaux :

Dans ce type d'héritage, une classe dérivée est créée à partir d'une autre classe dérivée.

1. Proposez un exemple illustrant ce type d'héritage, (trouver un programme sur le net illustrant cela ou le créer, le tester et le sauvegarder dans votre compte rendu)

Application 5

- Réalisez et tester le programme C++ proposé sur Openclassrooms, sur l'héritage qui est la suite du TP que vous avez réalisé au TP précédent

<https://openclassrooms.com/fr/courses/1894236-programmez-avec-le-langage-c/1898475-lheritage>