TP3: Programmation Orientée Objet_0

Introduction :

Comme expliqué dans le cours, la programmation orientée objet avec le C++ sous Eclipse est faite de manière modulaire. Un projet orienté objet est composé de trois parties (DDU):

1. Partie Déclaration dans un fichier header: nom_classe.h,
2. Partie Définition : programmation des méthodes de la classe, dans un fichier : nom_classe.cpp. Ce fichier doit inclure le fichier : nom_classe.h
3. Partie Usage : c'est le programme principal : main.cpp. Ici on inclut aussi : nom_classe.h et ici on créer les instances de la classe programmée et on peut utiliser leurs méthode.

Pour faire cette création, on procède comme expliqué ci-dessous : Créer un projet comme indiqué dans les figure 1, 2, 3

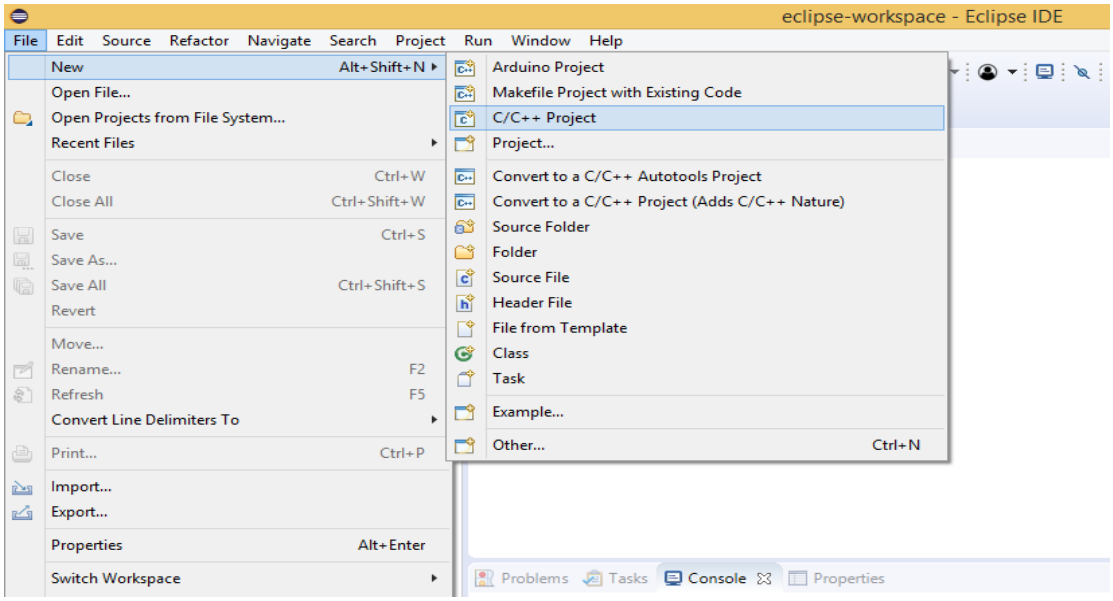

Figure 1 : création de projet C++

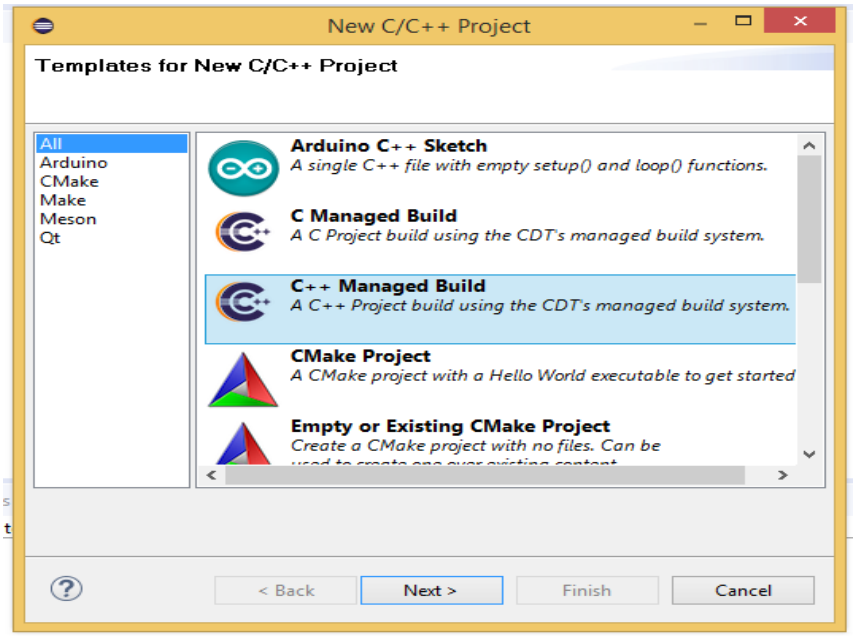

Figure 2 : création de projet

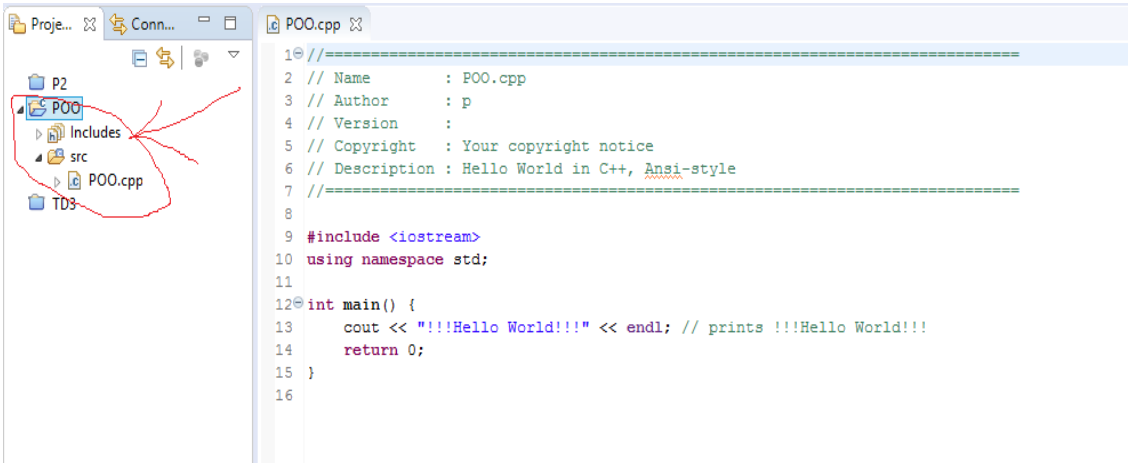

Figure 3 : création de projet POO

Après ça, il faut ajouter au projet POO déjà créé une classe comme indiqué sur la figure 4: Cliquer droite POO->New->class

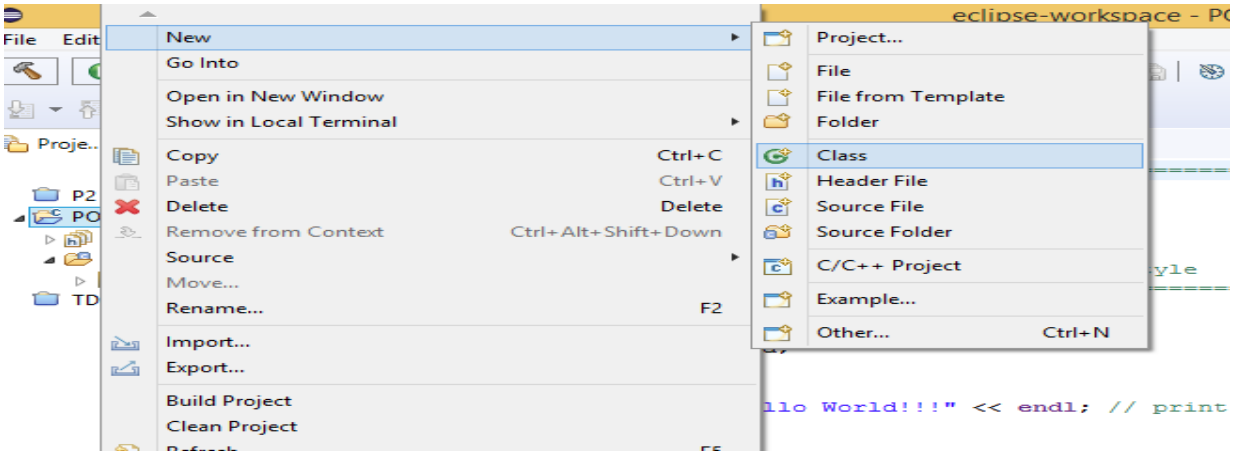

Figure 4 : ajouter une classe à un projet

Il faut donner le nom de la classe à crée comme sur figure 5 :

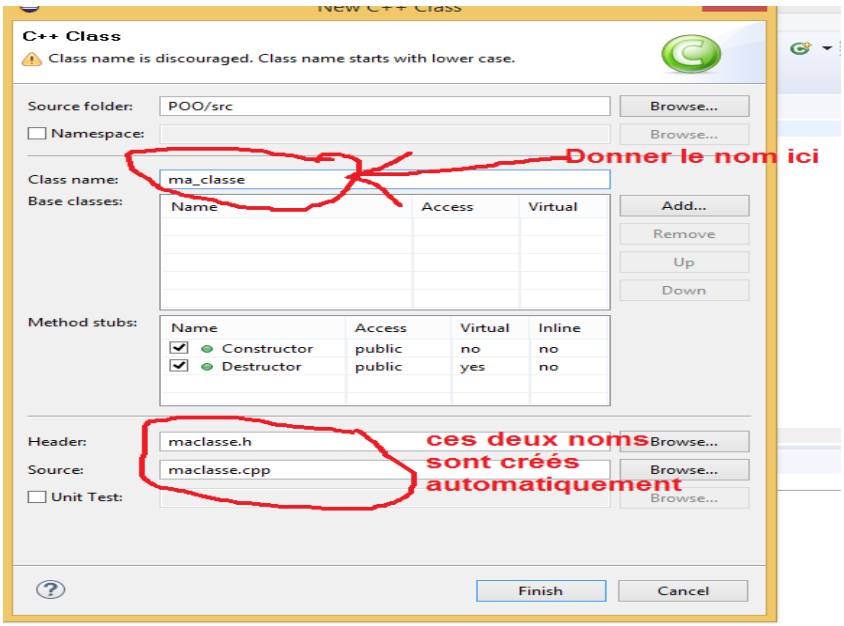

Figure 5 : donner le nom de la classe créée

Une fois, vous pressez « finish », vous aurez les deux fichiers ma_classe.h et ma_classe.cpp dans le projet POO. Il reste maintenant à ajouter les attributs et signatures des méthodes dans le fichier ma_classe.h et de faire la programmation des méthodes dans le fichier ma_classe.cpp. Comme indiqué sur la figure 6

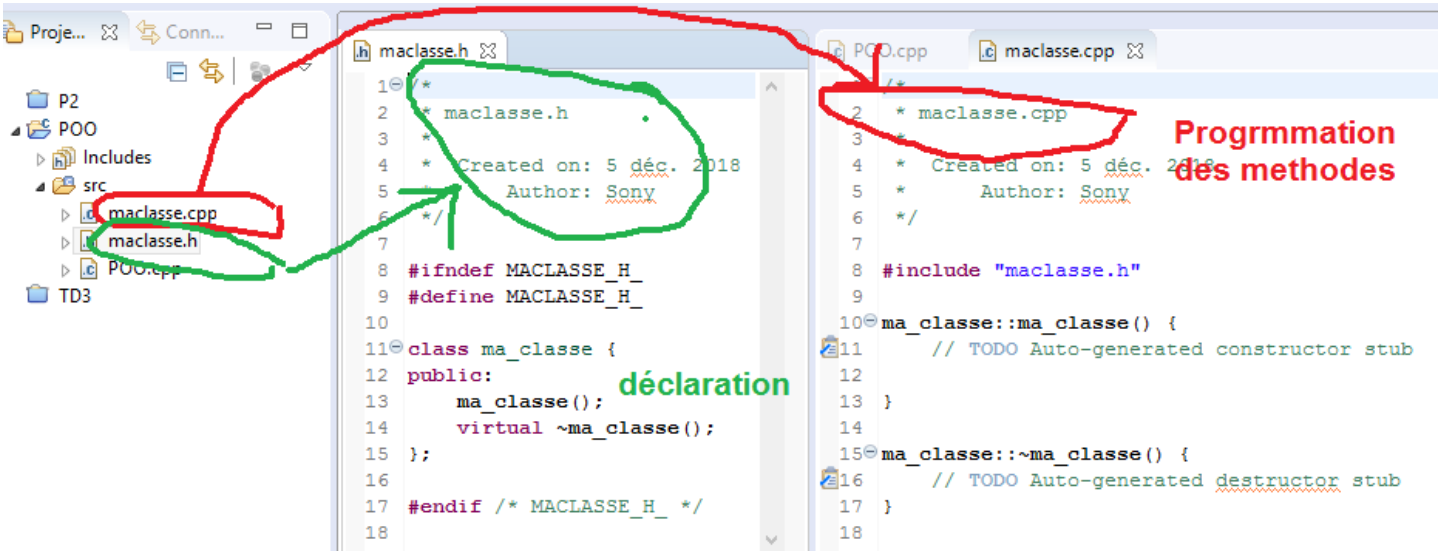

Figure 6 : classe h et classe cpp

Exercice 1 :

Utiliser la démarche ci-dessus, pour programmer les codes suivants :

```
class A{
    int x;
public:
    int get_x();
};
int A::get_x(){
return x;
}
int main() {
    A a;
    cout<<"a.x="<<a.get_x()<<endl;
    return 0;
}
```

==========

```
class A{
    int x;
public:
    int get_x();
};
int A::get_x(){
return x;
}
int main() {
    A a;
    cout<<"a.x="<<a.get_x()<<endl;
    return 0;
}
```

==========

```
class Complexe
{
```

```
public:
      Complexe(float x, float y); // premier constructeur de la classe :
                                  // fixe la partie réelle à x, la partie imaginaire à y
      void Lis(); // lit un nombre complexe entré au clavier
      void Affiche(); // affiche un nombre complexe
private:
      float re, im; // parties réelle et imaginaire
}

Complexe::Complexe(float x, float y) // constructeur avec paramètres
{
re = x;
im = y;
}
void Complexe::Lis() // lecture d'un complexe
{
cout << "Partie réelle ? ";
cin >> re;
cout << "Partie imaginaire ? ";
cin >> im;
}
void Complexe::Affiche() // affichage d'un complexe
{
cout << re << " + i " << im;
}
```

- **Dans le program « main », on veut faire :**

```
void main() // traitement principal
{
Complexe z1(0.0, 1.0); // appel implicite du constructeur paramétré
z1.Affiche(); // affichage de z1
cout << "\nEntrer un nombre complexe : ";
z2.Lis(); // saisie de z2
cout << "\nVous avez entré : ";
z2.Affiche(); // affichage de z2
}
```

==========

```
class Personne {
private:
      string my_prenom;
      string my_nom;
public:
      Personne( string prenom, string nom );
      void quiSuisJe();
};
// Personne.cxx
// Constructeur
Personne::Personne( string prenom, string nom )
{
      my_prenom = prenom;
      my_nom = nom;
}
void Personne::quiSuisJe()
{
      cout << "Je suis " << my_prenom << " " << my_nom << endl;
}
```

- Dans le programme principal, créer des instances de la classe personne et fais usage des méthodes de ces objets créés.

Exercice 2 :

Utiliser la démarche expliquée ci-dessus, pour faire le programme des exercices de TD3 : exo2-exo6