

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 crée 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

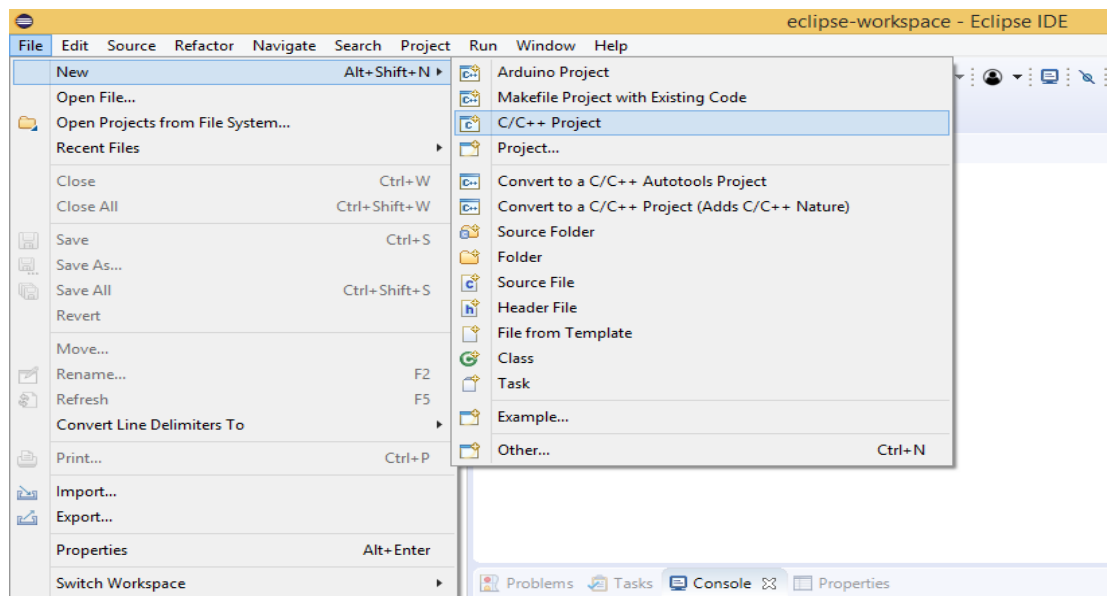


Figure 1 : création de projet C++

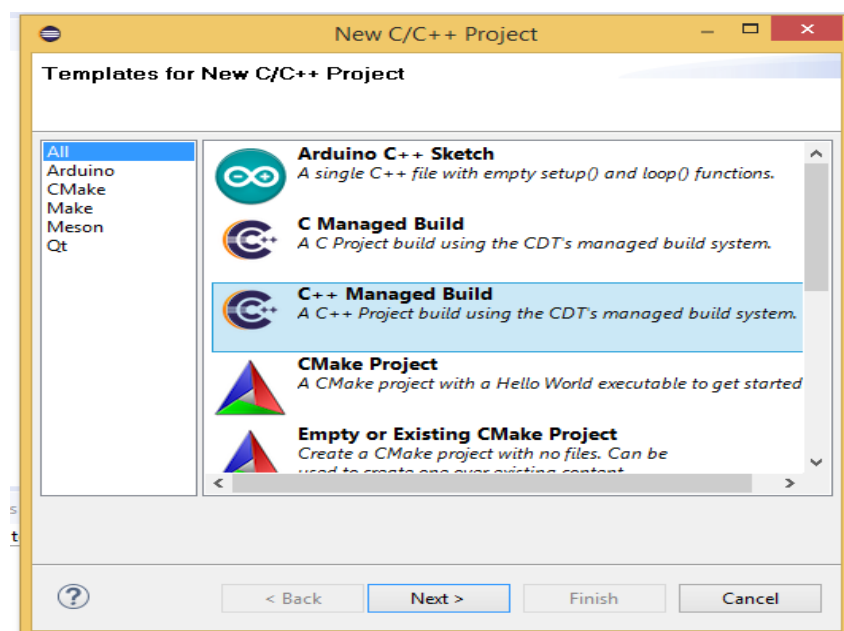


Figure 2 : création de projet

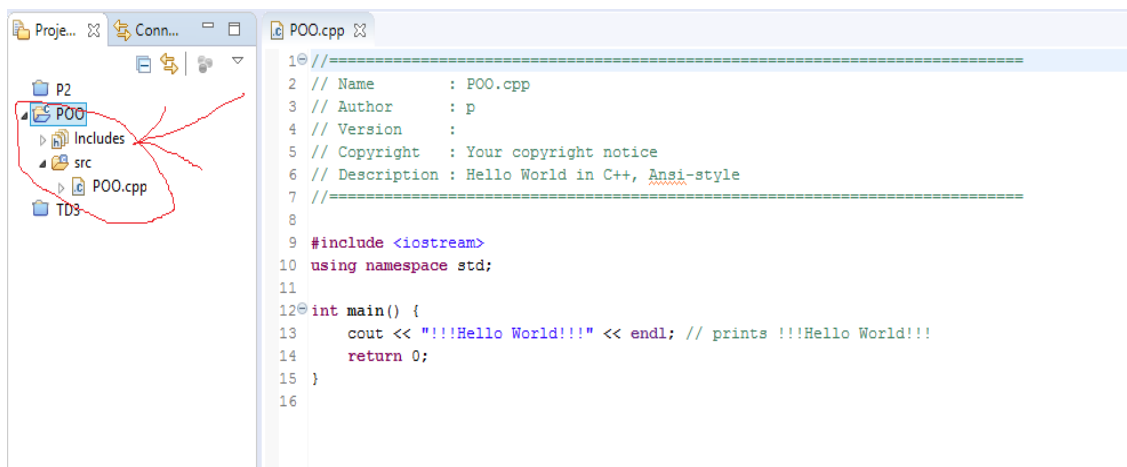


Figure 3 : création de projet POO

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

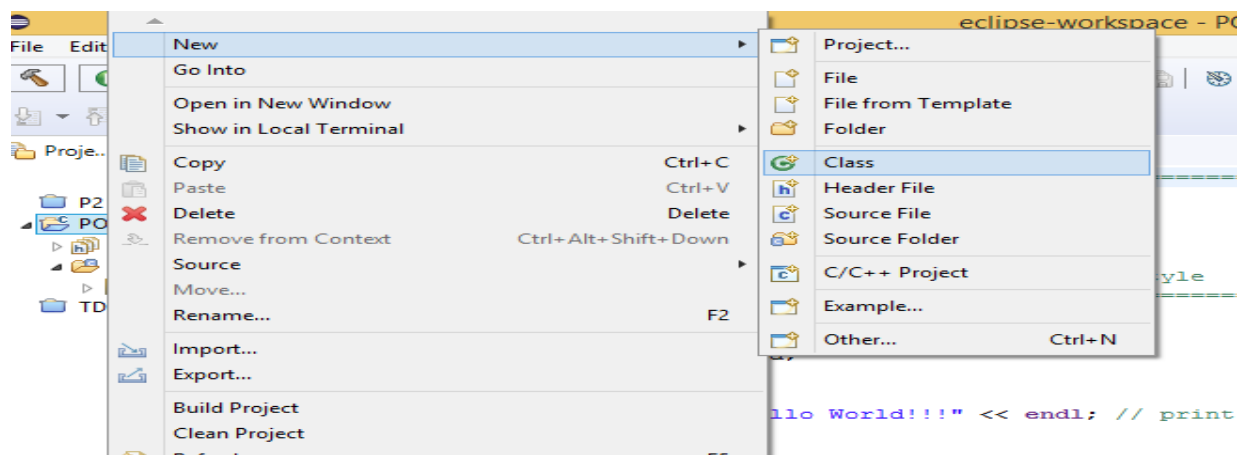


Figure 4 : ajouter une classe à un projet

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

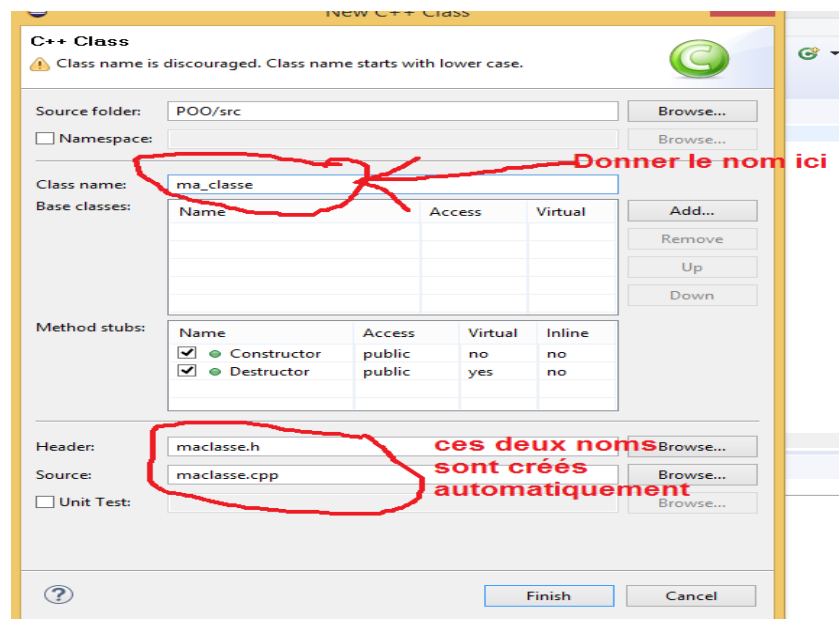


Figure 5 : donner le nom de la classe créée

The screenshot shows a C++ IDE with two files open: `maclasse.h` and `maclasse.cpp`. The `maclasse.h` file contains the class declaration, and the `maclasse.cpp` file contains the implementation. Red and green annotations highlight the relationship between the two files.

maclasse.h (Declaration):

```

1  /*
2  * maclasse.h
3  *
4  * Created on: 5 déc. 2018
5  * Author: Sony
6  */
7
8  #ifndef MACLASSE_H_
9  #define MACLASSE_H_
10
11 class ma_classe {
12 public:
13     ma_classe();
14     virtual ~ma_classe();
15 };
16
17 #endif /* MACLASSE_H_ */
18

```

maclasse.cpp (Implementation):

```

1  /*
2  * maclasse.cpp
3  *
4  * Created on: 5 déc. 2018
5  * Author: Sony
6  */
7
8  #include "maclasse.h"
9
10 ma_classe::ma_classe() {
11     // TODO Auto-generated constructor stub
12 }
13
14
15 ma_classe::~ma_classe() {
16     // TODO Auto-generated destructor stub
17 }
18

```

Annotations:

- Red:** A red box highlights the `maclasse.cpp` file in the project explorer. A red arrow points from this box to the `maclasse.cpp` file in the editor. Another red arrow points from the `maclasse.h` file in the editor to the `maclasse.cpp` file in the editor.
- Green:** A green box highlights the `maclasse.h` file in the project explorer. A green arrow points from this box to the `maclasse.h` file in the editor. Another green arrow points from the `maclasse.h` file in the editor to the `maclasse.cpp` file in the editor.

Text:

- maclasse.h:** The word `déclaration` is written in green next to the class declaration.
- maclasse.cpp:** The text **Programmation des methodes** is written in red next to the implementation.

Exercise 1 :

```
15 class A{
16     int x;
17 public:
18     int get_x();
19 };
20 int A::get_x() {
21     return x;
22 }
23 int main() {
24     A a;
25     cout<<"a.x="<<a.get_x()<<endl;
26     return 0;
27 }
```

=====

```
15 class A{
16     int x;
17 public:
18     int get_x();
19 };
20 int A::get_x() {
21     return x;
22 }
23 int main() {
24     A a;
25     cout<<"a.x="<<a.get_x()<<endl;
26     return 0;
27 }
```

3

```

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