# TUTORIAL 6: Basic of OOP

**OOP** : Object-oriented programming is about creating objects that contain both data and functions.

OOP is faster and easier to execute

OOP provides a clear structure for the programs

OOP helps to keep the C++ code DRY "Don't Repeat Yourself", and makes the code easier to maintain, modify and debug

OOP makes it possible to create full reusable applications with less code and shorter development time

***Class and Object:***

A class is a template for objects, and an object is an instance of a class.

When the individual objects are created, they inherit all the variables and functions from the class.

Everything in C++ is associated with *classes and objects, along with its attributes and methods*. For example: in real life, a **CLASS** car is an object. The car has attributes, such as weight and color, and methods, such as drive and brake.

Attributes and methods are basically variables and functions that belongs to the class. These are often referred to as "class members".

A class is a user-defined data type that we can use in our program, and it works as an object constructor, or a "blueprint" for creating objects.

**Create a Class (CLASS is forward concept of Structure /Union)**

To create a class, use the **class** keyword:

**Example**

Create a class called "MyClass":

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};
```

Example explained

The class keyword is used to create a class called MyClass.

The public keyword is an access specifier, which specifies that members (attributes and methods) of the class are accessible from outside the class. Inside the class, there is an integer variable myNum and a string variable myString. When variables are declared within a class, they are called attributes.

At last, end the class definition with a semicolon ;.

**Create an Object**

In C++, an object is created from a class. We have already created the class named MyClass, so now we can use this to create objects.

To create an object of MyClass, specify the class name, followed by the object name.

To access the class attributes (myNum and myString), use the dot syntax (.) on the object:

**Example**

Create an object called "myObj" and access the attributes:

```
class MyClass {       // The class
  public:             // Access specifier
    int myNum;        // Attribute (int variable)
    string myString;  // Attribute (string variable)
};

int main() {
  MyClass myObj;  // Create an object of MyClass

  // Access attributes and set values
  myObj.myNum = 15;
  myObj.myString = "Some text";

  // Print attribute values
  cout << myObj.myNum << "\n";
  cout << myObj.myString;
  return 0;
}
```

Same way you can create multiple objects of same class.

## Class Methods

Methods are functions that belongs to the class.

There are two ways to define functions that belongs to a class:

>> Inside class definition

>>  Outside class definition

In the following example, we define a function inside the class, and we name it "myMethod".

**Note**: *You access methods just like you access attributes; by creating an object of the class and using the dot syntax (.):*

**Inside Example**

```
class MyClass {        // The class
  public:              // Access specifier
    void myMethod() {  // Method/function defined inside the class
      cout << "Hello World!";
    }
};

int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

To define a function outside the class definition, you have to declare it inside the class and then define it outside of the class. This is done by specifying the name of the class, followed the **scope resolution :: operator**, followed by the name of the function:

**Outside Example**

```
class MyClass {        // The class
  public:              // Access specifier
    void myMethod();   // Method/function declaration
};

// Method/function definition outside the class
void MyClass::myMethod() {
  cout << "Hello World!";
}

int main() {
  MyClass myObj;     // Create an object of MyClass
  myObj.myMethod();  // Call the method
  return 0;
}
```

**NOTE:** You can use declaration of function outside the class using scope resolution only and only if class declaration is global.Also method declaration required even you use function call outside from class.

Passing Parameters in Method

**Example:**

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

class Car {
  public:
    int speed(int maxSpeed);
};

int Car::speed(int maxSpeed) {
  return maxSpeed;
}

int main() {
  Car myObj; // Create an object of Car
  cout << myObj.speed(200); // Call the method with an argument
  return 0;
}
```