Classes, objects and methods:

Attributes are variables and methods are functions. Attributes are also referred to as “class members”.
Class – It is a user-defined data type that we can use in our program , it works as an object constructor or a “blueprint” for creating objects.

```
Class MyClass{
public:
int myNum;
string myString;
};
```

```
Object: MyClass myObj;
myObj.myNum=5;
myObj.myString=”Hello”;
```

Methods: They are functions that belongs to the class, they define the behaviour/action taken by object.
There are 2 ways to define functions that belongs to the class:

- Declaration and Definition at one.
- First Declare and then define: void MyClass::myMethod(){}

```
Class MyClass{
public:
Void myMethod(){
Cout<<”Hello World!”;
    }
};
```

```
Example: class Student{
public:
string name;
```

```
int rollNo;
static int age;
Void display(){
   }
};
```

Access specifiers: They define how the members (attributes and methods) of a class can be accessed. In C++ there are 3 access specifiers:
Public – members are accessible from outside the class.
Private – members cannot be accessed(or viewed) from outside the class.
Protected – members cannot be accessed from outside the class, however, they can be accessed in inherited classes.
By default, all members of a class are private if you don't specify an access specifier, they will be private.(for C++, but in java default is public).

```
Class myClass{
Public: int x;
Private: int y;
};
Int main(){
myclass myObj;
myObj.x=5;
myObj.y=5;//Error: y is private
return 0;
}
```

Default copying: By default, objects can be copied by assignment operator. In particular, a class object can be initialized with a copy of an object of it's class.

Ex: Date d1 = my_birthday;//d1 and my_birthday are objects of Date.
Date d2{my_birthday} ; // Above statement can be written in this way too.
Copy of a class object is a copy of each member.

Static members of a C++ Class:
We can define class members static using static keyword.
When we declare a member of a class as static it means no matter how many objects of the class are created, there is **only one copy** of the static member.
A static member is shared by all objects of the class.
All static data is **initialized to zero** when the first object is created, if no other initialization is present.

We can't initialize it in the class definition but it can be initialized outside the class by using the scope resolution operator because there is no memory allocated to the class but the memory will be allocated to the object of the class so we can initialse it outside the class.

Eg:
```
Class Box{
Public:
Static int objectCount;
};
Int Box::objectCount = 0;
```

Static Function members: It is independent of any particular object of the class.

A static method cab only access static data member , other static member function any other functions from outside the class.
Static member functions have a class scope and they do not have access to the **this** pointer of the class.

Eg:

```
Class Box{
Public:
Static int objectCount;
Static int getCount(){
Return length;
}
Private:
Double length;
};
Box::getCount();
```

We can call the static methods directly using class.
Error in the above code is static method can only access only static members but it can’y access length variable.

Passing and returning objects in C++:
We can pass class objects as arguments and also return them from a function .
Syntax: fun(object_name);
Eg: void fun(A){}

```
Class Example{
Public: int a;
Void add(Example E){
 a = a+E.a;
  }
};
```

```
Example E1,E2;
E1.a=5;
E2.a=10;
E2.add(E1);
Cout<<E1.a<<E2.a<<”\n”;
```

Returning object as arguments:

```
Class Example{
Public: int a;
Example add(Example Ea, Example Eb){
Example Ec;
Ec.a = Ec.a + Ea.a + Eb.a;
Return Ec;
  }
};
```

```
Example E1,E2,E3;
E1.a=5;
E2.a=10;
E3 = E3.add(E1,E2);
Cout<<E3.a<<”\n”;
```