What Is Class

A class is a blueprint, or prototype which defines and describes the member attributes and member functions.

A class is a three-part containing the class name, data attributes (variables) and the member functions.

The Class name is an identifier, which is the user-defined data type name. The member attributes hold data just like variables which can be accessed by the member attributes only via particular class/objects. The member functions may have productive behaviors or may be designed to do some operations with member attributes just like normal functions, but the member functions can be accessed only via the corresponding class or its objects.

Class Advantage

1. A class is a user-defined and reusable entity.
2. A class reduces complicity of holding data.
3. A class just holds data and do operations with help of member variables and member functions.
4. A class maintains software quality and is well structured.
5. An Object is a realization of the particular class.

Objects

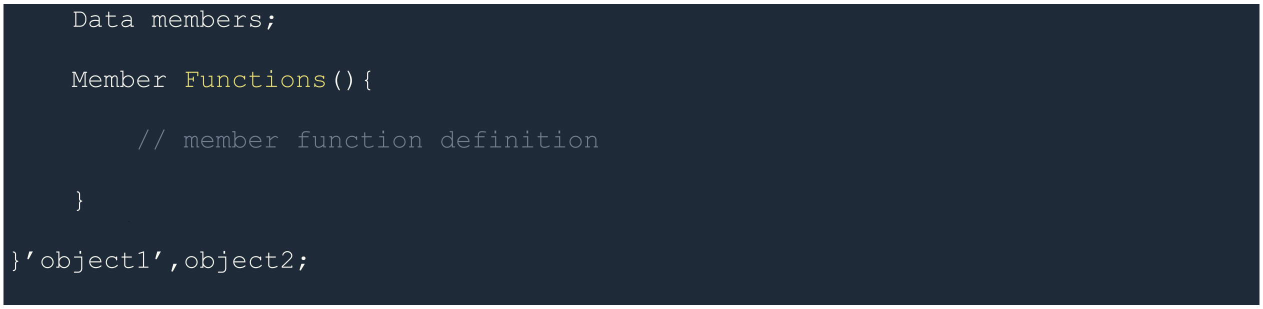
Objects are the instances of a class. No storage is assigned when we define a class.but if an object of a class is defined then it will need a space in memory to hold its data. Objects are actual representation of a class

Each object has different data variables. Objects are initialised using special class functions called **Constructors**. We will study about constructors later.

General syntax of the Classes

```
class ClassName{

    Access specifier:
```



```
    Data members;

    Member Functions(){

        // member function definition

    }

}'object1',object2;
```

where **'Classname'** is an identifier of the class, **'object1'** and **'object2'** are objects of this class (which is an optional declaration, we may be declared in the main function).

A class may contain member variables, member functions, and access specifiers. But all these are optional. ie., a class can be created with or without all three of these. For a class or a variable or a function without an access specifier, the compiler itself provides private access by default.

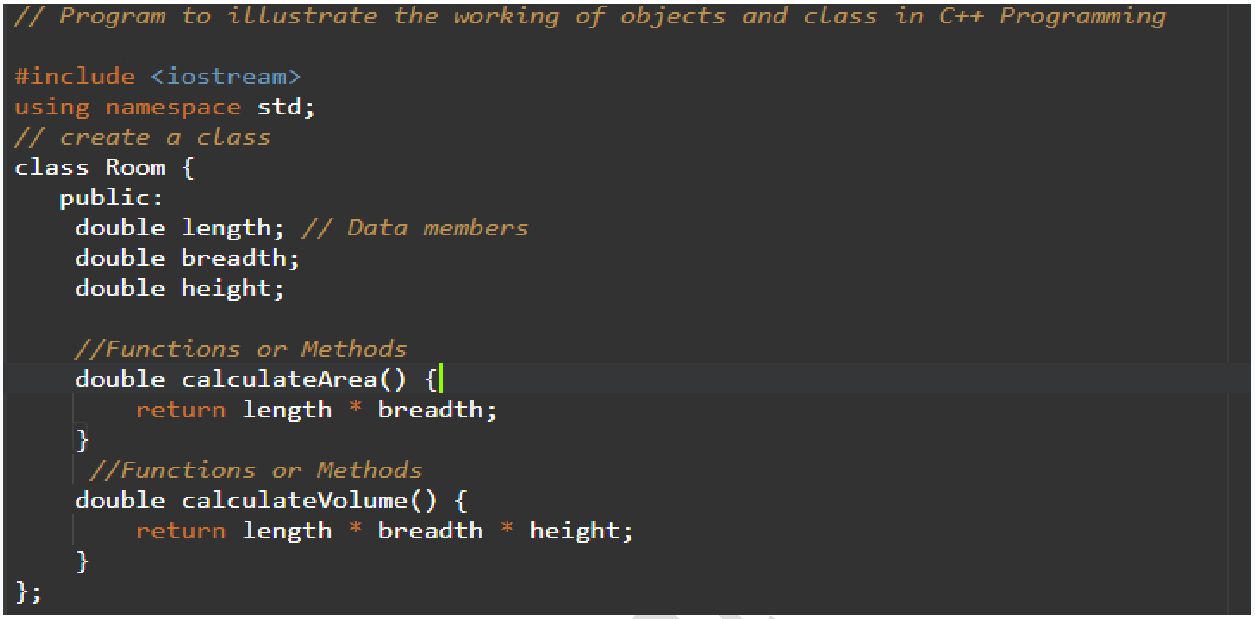
```
#include <iostream>
using namespace std;
class Course {
  public:
    int id;//data member (also instance variable)
    string name;//data member(also instance variable)
};
int main() {
   Course  s1; //creating an object of Student
   s1.id = 201;
   s1.name = "ccat preparation : c++";
   cout<<s1.id<<endl;
   cout<<s1.name<<endl;
   return 0;
}
```

C++ Access Data Members and Member Functions

We can access the data members and member functions of a class by using a . (dot) operator. For example,

```
Room1.calculateArea(); // Calling the Method or function
```

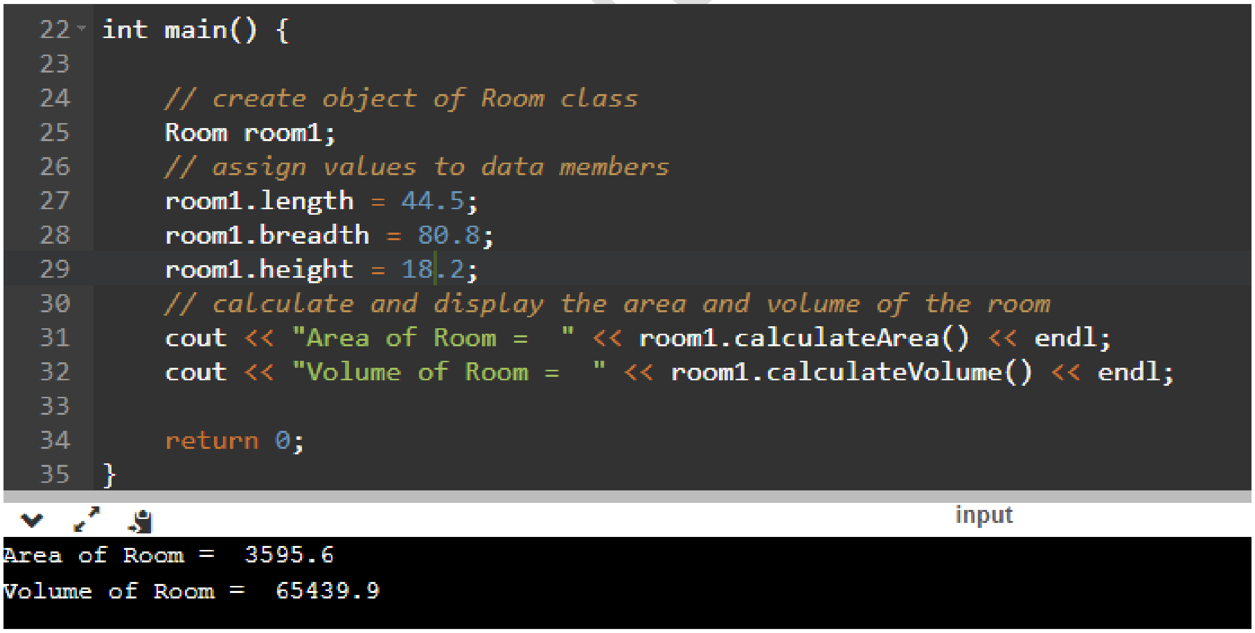
```
room1.length = 5.5; // Assigning the value to member variable of class
```



```
// Program to illustrate the working of objects and class in C++ Programming

#include <iostream>
using namespace std;
// create a class
class Room {
   public:
    double length; // Data members
    double breadth;
    double height;

    //Functions or Methods
    double calculateArea() {
        return length * breadth;
    }
     //Functions or Methods
    double calculateVolume() {
        return length * breadth * height;
    }
};
```



```
int main() {

    // create object of Room class
    Room room1;
    // assign values to data members
    room1.length = 44.5;
    room1.breadth = 80.8;
    room1.height = 18.2;
    // calculate and display the area and volume of the room
    cout << "Area of Room =  " << room1.calculateArea() << endl;
    cout << "Volume of Room =  " << room1.calculateVolume() << endl;

    return 0;
}
```

```
Area of Room =  3595.6
Volume of Room =  65439.9
```

In this program, we have used the `Room` class and its object `room1` to calculate the area and volume of a room.

In `main()`, we assigned the values of `length`, `breadth`, and `height` with the code:

```
  room1.length = 44.5;

   room1.breadth = 80.8;

   room1.height = 18.2;
```

Note the use of the keyword `public` in the program. This means the members are public and can be accessed anywhere from the program.

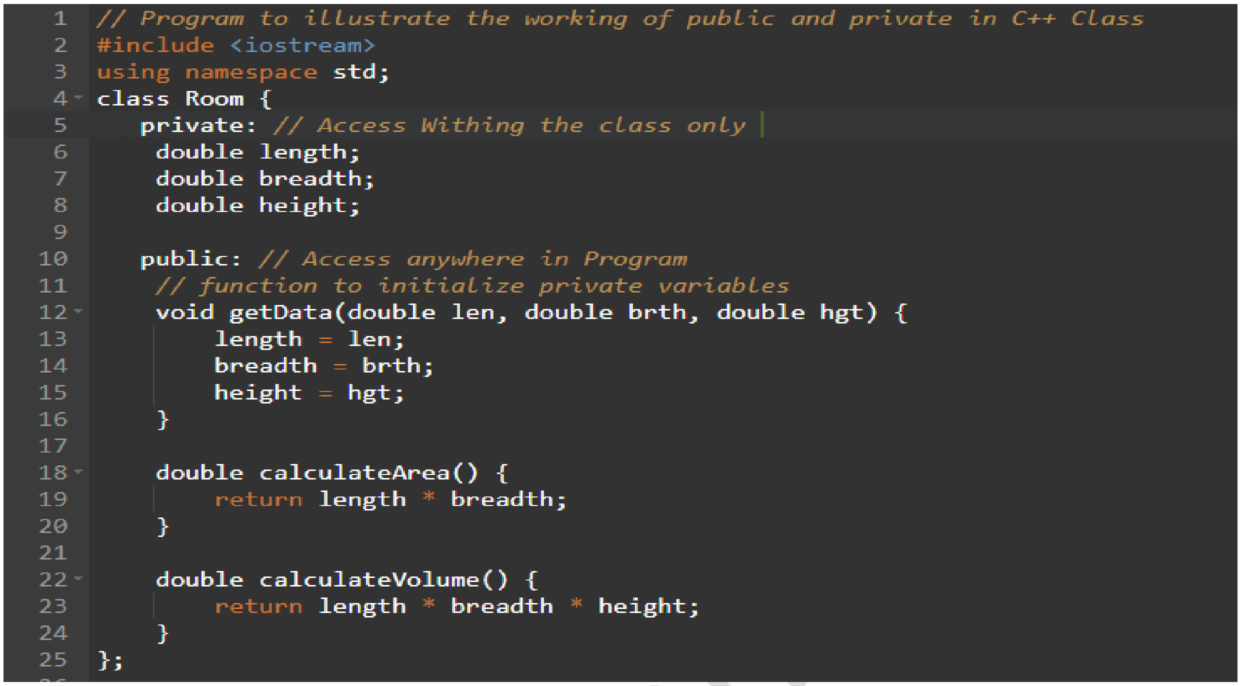
As per our needs, we can also create private members using the `private` keyword. The private members of a class can only be accessed from within the class. For example,

```
class Room {

private:
    int length;
    void function1() { }

public:
    int breadth;
    void function2() { }
}
Here, length and function1() are private and are. Thus they cannot be accessed
from outside the class.
```

Using public and private in C++ Class

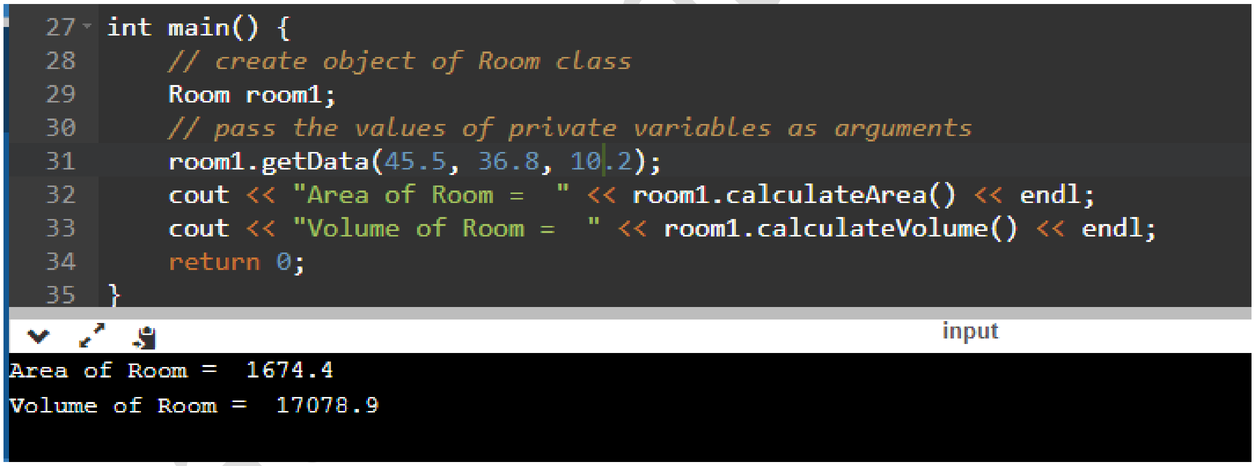


```
// Program to illustrate the working of public and private in C++ Class
#include <iostream>
using namespace std;
class Room {
   private: // Access Withing the class only
     double length;
     double breadth;
     double height;

   public: // Access anywhere in Program
     // function to initialize private variables
     void getData(double len, double brth, double hgt) {
         length = len;
         breadth = brth;
         height = hgt;
     }

     double calculateArea() {
         return length * breadth;
     }

     double calculateVolume() {
         return length * breadth * height;
     }
};
```



```
int main() {
    // create object of Room class
    Room room1;
    // pass the values of private variables as arguments
    room1.getData(45.5, 36.8, 10.2);
    cout << "Area of Room =  " << room1.calculateArea() << endl;
    cout << "Volume of Room =  " << room1.calculateVolume() << endl;
    return 0;
}
```

```
Area of Room =  1674.4
Volume of Room =  17078.9
```

The above example is nearly identical to the first example, except that the class variables are now private.

Since the variables are now private, we cannot access them directly from `main()`.

Hence, using the following code would be invalid:

```
// invalid code
obj.length = 42.5;
obj.breadth = 30.8;
obj.height = 19.2;
```

Instead, we use the public function `getData()` to initialize the private variables via the function parameters `double len`, `double brth`, and `double hgt`.