Classes and Objects in C++

Everything in C++ is associated with classes and objects, along with its attributes and methods. For example: a car is a class. 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.

We can think of a class as a sketch (prototype) of a house. It contains all the details about the floors, doors, windows, etc. Based on these descriptions we build the house. House is the object.

Create a Class

A class is defined in C++ using keyword class followed by the name of the class.

The body of the class is defined inside the curly brackets and terminated by a semicolon at the end.

Syntax:

```
class className {
  // some data
  // some functions
};
```

Syntax Example:

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

Example: Creating Class

```
class Room {
   public:
      double length;
      double breadth;
      double height;

      double calculateArea(){
         return length * breadth;
      }

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

We defined a class named Room.
The variables length, breadth, and height declared inside the class are known as data members. And, the methods calculateArea() and calculateVolume() are known as member functions of a class.

Create an object

When a class is defined, only the specification for the object is defined; no memory or storage is allocated.

To use the data and access functions defined in the class, we need to create objects.

Syntax:

```
className objectVariableName;
```

Syntax Example:

```
void sampleFunction() {
   Room room1, room2;
}

int main() {
   Room room3, room4;
}
```

Access Data Members and Member Functions

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

Syntax: Accessing data member

room1.length = 5.5;

In this case, it initializes the length variable of room1 to 5.5.

Syntax: Accessing data function

room2.calculateArea();

This will call the calculateArea() function from Room class for object room2.

Working of Object and Class

```
class Room {
   public:
      double length;
      double breadth;
      double height;
};

void sampleFunction() {
   Room room1;
   room1.length = 1;
   room1.breadth = 1;
   room1.height = 1;
   cout << "Room1 Area - " << room1.length * room1.breadth << endl;
   cout << "Room1 Volume - " << room1.length * room1.breadth *
room1.height << endl;
}
int main(void) {
   Room room2;
   room2.length = 2;
   room2.breadth = 2;
   room2.height = 2;
   cout << "Room2 Area - " << room2.length * room2.breadth << endl;
   cout << "Room2 Volume - " << room2.length * room2.breadth *
room2.height << endl;
   sampleFunction();
}
```

Working of Object and Class with methods()

Methods are functions that belongs to the class.

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

- Inside class definition
- Outside class definition

Example1: Inside class method()

```
class Room {
   public:
      void class_method(){
         double length = 2;
         double breadth = 2;
         double height = 2;
         cout << "Room1 Area - " << length * breadth << endl;
         cout << "Room1 Volume - " << length * breadth * height << endl;
      }
};
int main(void) {
   Room room1;
   room1.class_method();
}
```

Example2: outside class method()

```
class Room {
   public:
      void class_method();
};
void Room :: class_method(){
   double length = 3;
   double breadth = 3;
   double height = 3;

   cout << "Room2 Area - " << length * breadth << endl;
   cout << "Room2 Volume - " << length * breadth * height << endl;
}
int main(void) {
   Room room2;
   room2.class_method();
}
```

methods() with parameters and return value

Example:

```
class Room {
   public:
      int length;
      int breadth;
      int height;

      void Data(int len, int brth, int hgt){
         length = len;
         breadth = brth;
         height = hgt;
      }
      int Area();
      int Volume();
};

int Room :: Area(){
   return length * breadth;
}

int Room :: Volume(){
   return length * breadth * height;
}

int main(void) {
   Room room1;
   room1.Data(2,4,6);
   cout << "Room1 Area - " << room1.Area() << endl;
   cout << "Room1 Volume - " << room1.Volume();
}
```

methods() with parameters and this pointer

```
class Room {
   public:
      int length;
      int breadth;
      int height;

      void Data(int length, int breadth, int height){
         this -> length = length;
         this -> breadth = breadth;
         this -> height = height;
      }

      int Area(){
         return length * breadth;
      }

      int Volume(){
         return length * breadth * height;
      }
};

int main(void) {
   Room room1;
   room1.Data(2,4,6);
   cout << "Room1 Area - " << room1.Area() << endl;
   cout << "Room1 Volume - " << room1.Volume();
}
```