FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture02

Lecture 2: Introduction to Classes and Objects in C++

So welcome to lecture 2, Introduction to Object-Oriented Programming. So in the last lecture, we talked about the basic introduction about the objects and the classes. And here, right, I will start with what you mean by objects in, let us say, C++ or any object-oriented programming. So, an object is nothing but the instance of a class, right? As I told you in the last class, when we are having, let us say, class A, the small a is an object.

So let us assume. right? So small a is an object, and capital A is a class, right? So with this object, assume that in the main, I can write like this a dot; let us say there is a factorial function, right? The factorial member function.

So through this object, this object can invoke whatever the member functions it has, right? So that means this object can access the right, all the members right inside the class. So, for example, let us assume there is a function called the member function called factorial, right? So, this factorial function can be accessed.

So, object A can invoke this function, right, and then it is going to store in some integer value, M, and assume that it is right. So, we will do it in the main. So that means this object A, right? So, which can invoke any members? All the members inside the class, right?

So for example, time. Time is the class, and capital T1 is the object, right? So time T1, if I write, T1 is an object, and time is the class. So here you have when you are defining, let us assume, The object, so the object can be tangible or intangible. right.

So, in real-world applications, let us assume you have a chair, bike, marker, pen, table, or car. So, these are all considered objects. So now, we have two different

objects: tangible and intangible. So, any physical objects like a chair or bike are all called tangible objects. So what do you mean by intangible objects?

A banking system. So assume that I have a bank class, right? Under which I have some objects, B1, B2, B3, etc. Correct? So now I call this bank as the intangible object.

Similarly, institutes are universities. Alright? So, at university, I have U1, U2, and U3. So U1, U2, U3 are all the objects, right? And university is the class.

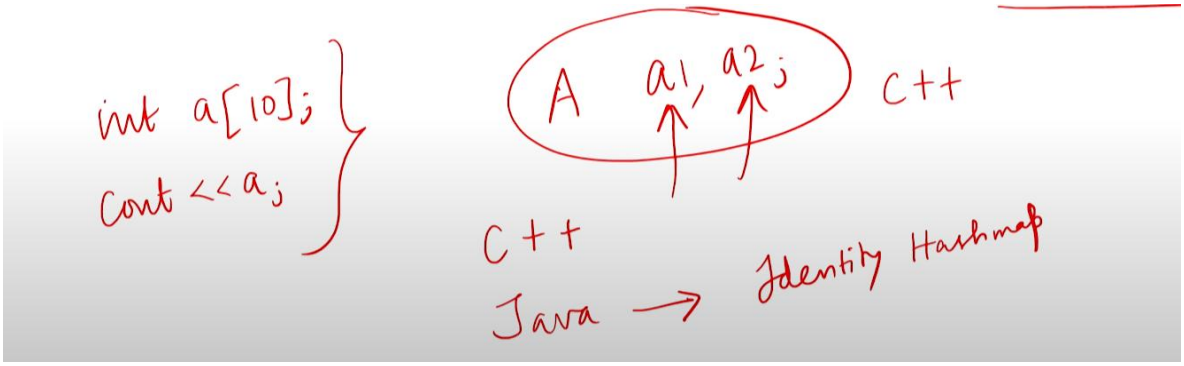
So this is intangible. So, these are all considered intangible objects. So, this is the difference between tangible objects and intangible objects. So as we have seen in the last class, right? So when I am defining, so let us say a state.

So state is nothing but the member data, right? So the data which is hold, suppose assume that I define int a. Right? Assume that I define int A and A equal to let us say 10. So it is taking the value 10.

Right? So that means this is representing a value of an object. A state is nothing, but it represents the value of an object. And similar behaviour. Right?

So which is representing the functionality of an object. So, suppose we define, let us say, a bank as an intangible object. So, in the bank, what are all the functionalities that happen, right? So the customer can deposit the money or withdraw the money, right? So these are all considered as the behavior.

Or when I am talking about the physical, right, objects, so what are all the functionalities, right? So if I take a bike, so the bike has to run smoothly, right? Right, so these are all the functionalities or in the program if I take, so what are all the functions you are writing? What are all the member functions that you are writing? Right, so these are all nothing but the behavior.



And when I am defining, suppose assume that I define a class A. Right, so under which I have let us say a1, a2, two objects. So now these two are all having identity. So if I take C++, right? So when I instantiate an object a1, so this a1 will be stored in some memory space, right? So it will have a unique identity.

And A2 will be stored in the separate memory space. So when I want to access, let us say you put C out. Let us say A1. Right? You want to find out where exactly it is being stored.

You might have done in the case of arrays. Right? So, assume that you are declaring an array int A of 10 and you try to print this. C out A. Right? So, can you think what will be the answer?

Yeah. So, this will be a base address. Right? So, when you define. So, in fact, array, this you consider as an object.

Consider as an object. Array of 10 numbers. And then you write C out A. What is the identity? Where it is being stored? So in a similar way, when I am talking about the objects and classes, A1, A2 are objects here and capital A is the class and if I want to know, so A1 will be stored in some place and A2 will be stored in another place and we can have the unique ID.

Every object has the unique ID. In a Java programming, so this is identity hashmap. So this is identity hashmap. So A1 is identity hashmap and A2 will also have the identity hashmap.

So this is how we have three characteristic features of an object which is a state, behavior and identity. So the tangible object. So pen. Assume that pen is an object. So it will have a name.

Parker. Parker. And color is some golden or blue. Right. So this is known as its state.

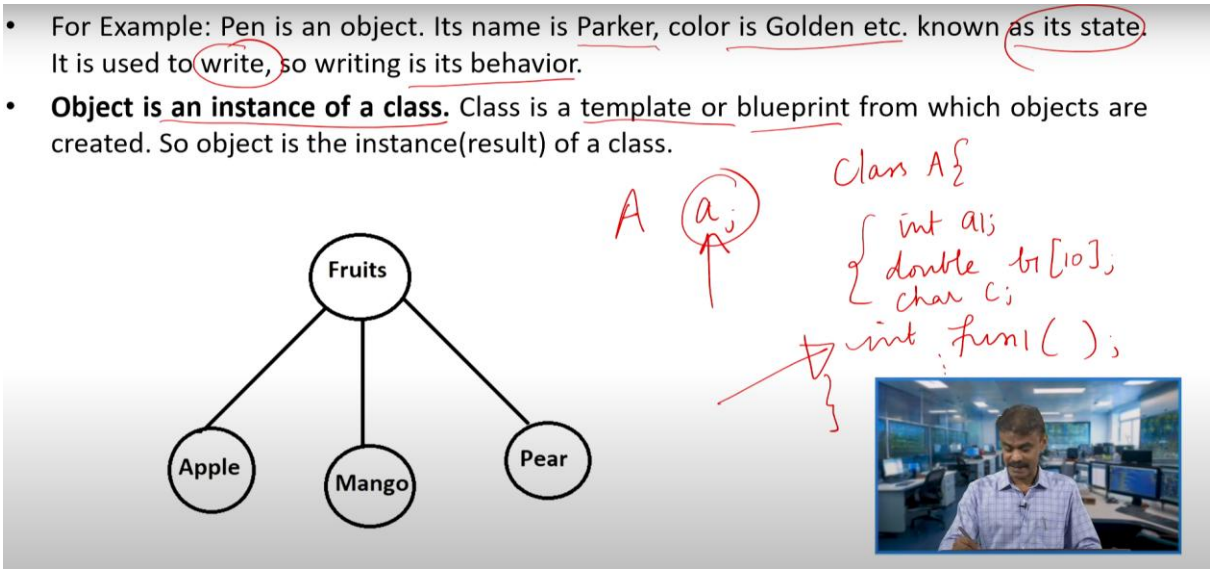
Right. So pen is an object. So it has. So either the mate. Right.

Parker. Or you have a golden color. Right. So these are all nothing but state. State.

State of states. It is like int A comma B. A is a Parker. B is a golden color. Right. And what do you want the functionality, right?

So that is the behavior. So pen means it has to write, right? The writing is the behavior, okay? So writing is the behavior. So as I said, when we have an object A and, sorry, class capital A and an object small a, right?

So the object is nothing but the instance of a class, right? So class is nothing but A template or a blueprint, right? So it may have several objects. The class may contain several objects.



So suppose I am defining, let us say, class A. So these are several objects int A, right? Int a1. So you have double, let us say, b1 of 10, right? Array of 10 elements or character C. So you can have several objects. Similarly, this can have several function name.

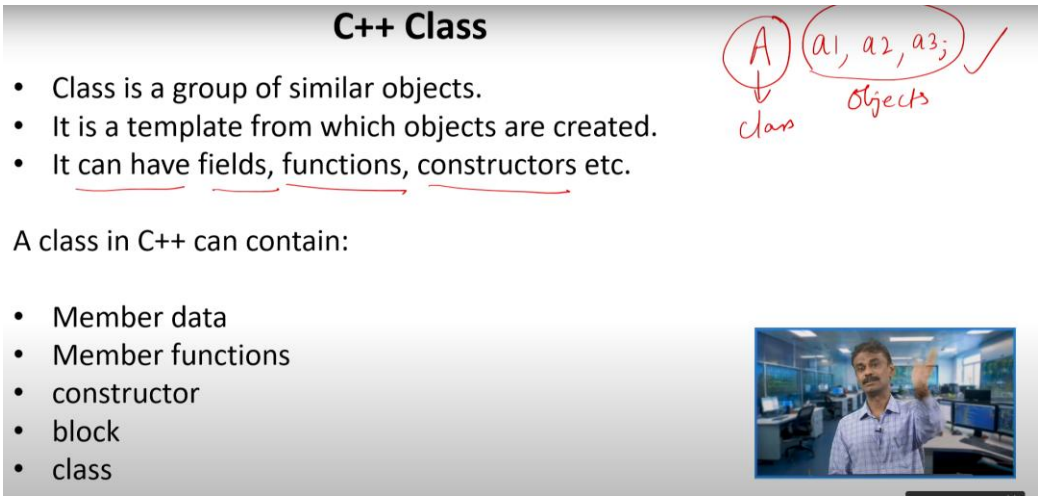
Int fun1, fun2 and so on. Right? So, you will have a blueprint. What exactly you are going to do in this program? So, I can use various objects and various functions.

So, one class. You may use several classes. Right? So, it is basically blueprint. What are the objects that I am going to use?

What are all the member data that I am going to use? What are all the member data that I am going to use and what are all the member functions that I am going to use. So which is nothing but the template or blueprint, the complete A, the class A. Which has several member data and several member functions. So I put dot dot dot, that means several member functions. You can have function 2, function 3, let us say factorial.

So the complete template or complete blueprint which contains objects. Right. So, which contains member data and member functions. So, now from here, let us say class A, I have object A. Right. I have created an object.

So, now this A, small a can access, let us say a1. It can access b1 of 10. It can access C. Right. It can access the function. In fact, it will invoke those.



I can put A.a1. Assume that a1 is taking the value. A.a1. Right. Or I can write in the main a dot a1 equal to 10, which I can do that.

Right. So I can instantiate an object. So this is what exactly you are doing. So object is the instance of a class. You have instantiate an object.

So if I consider fruits as a class. Right. If I consider fruits as a class, I have objects. So apple, mango, pear, they are all considered as objects. For the layman point of view, if I consider fruits as a class, so I have apple, mango, pears, they are all called the objects.

I hope it is clear. Now you know how to distinguish what is meant by class and what is meant by an object or together class and objects. So now when I consider C++ class, so class is a group of similar objects. Right. So when I write A, I will use the same.

Right. a1, a2, a3. These are all the objects. Right. So they are all the group of similar objects.

Three objects are there. Right. So they are all similar. For example, in the previous case, we have seen fruits. Right.

So these are all objects. And here you have a class. Right. So class. When I consider a class, it is a template from which I can create the objects.

Right? Class A is a template. So it contains so many. Right? The member data and member functions.

Correct? So when I have the class, so it can have the fields. Right? That means the member data. And then I can have a member function.

We are going to study the concept called constructors, distractors. So several concepts now we are going to see that. So we have seen only so far I have in fact introduced only the member data and member functions. So we will have several concepts like a constructor. Inside the class you can have another class.

Destructors. So these are all the concepts that we are going to study under the class. So in general when I am defining a C++ class So it will contain, let us say, member data, member functions. So this extensively I told.

And then we will study the concept of constructor. And then inside the class you have the block, right? So inside the class you have blocks. You can have blocks, right? And then inside the class you have another class or several classes.

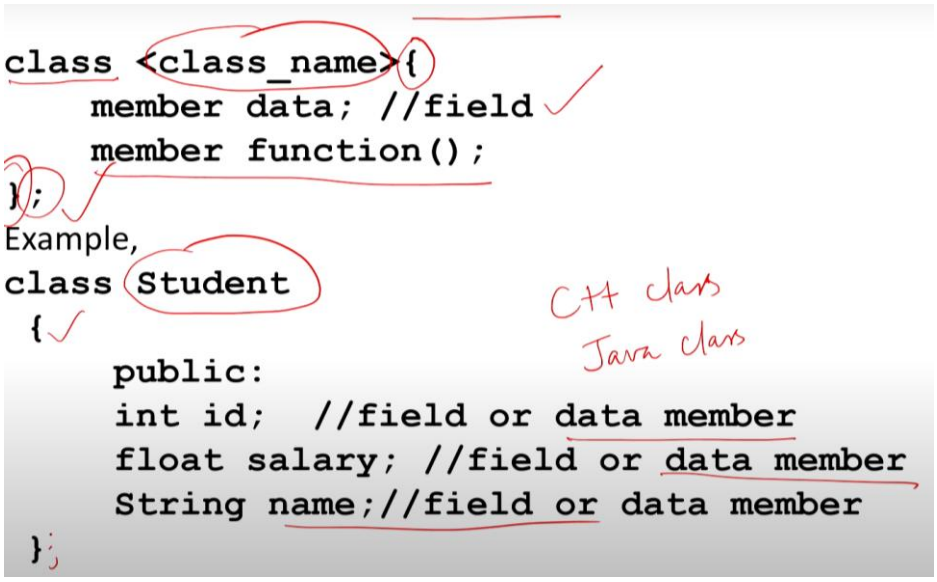
So, in general, so these are all some of them I told because you have Destructor also. So, a class in C++ can contain member data, member functions, constructor, block, class, Destructor, etc. So, now the programming point of view, syntax point of view. So, how do you define a class? So, class.

So, here you have a keyword called class. You have to use a keyword class. And then it has name of the class. So here you have the name of the class, right? So here we have member data, right?

Just you call it as a field. First time we are introducing field. Last slide we had seen and member data is nothing but the field. And you can have the member function, right? So you can have, in fact, we are going to study several functions.

Constructors, destructors and all those. So here you have member data, member function and you have the braces. So starting with the braces and you are ending with the brace followed by a semicolon. Whereas the same syntax we are going to see in the case of Java, you will not find the semicolon. So there is a difference between C++ class and Java class.

Almost same. You will have a member data, member functions or whatever I told. So it will be there in C++ as well as Java. But the syntax will slightly differ. In the case of Java, you cannot find the semicolon.



So we can take a typical example. Whereas in the case of C++, we should have this semicolon. So in the case of C++, we are having this semicolon. So now we will take a simple example. Class student.

That means student is the name of the class. And then you have a braces which is beginning. Right? So you have a public. Int ID.

Right? So int ID. This is the data member. Right? So this is the data member.

Float salary. Another data member or you call it as a field. String. Right? Name.

Name. Right. So in the case of C++, you put the semicolon here. I purposefully did not put this because I want to distinguish the C++ class and Java class. So if I put semicolon, so you call this as C++ class.

Right. So suppose I do not put, let us say the semicolon. Right. So if I do not put the semicolon, you consider this as a Java class. You consider this as Java class, right?

So this is how your syntax goes. So let us consider C++ initially. We will write a program in Java as well. So let us consider the C++ class, right? So with semicolon and I put the semicolon, right?

So you have a class, name of the class and inside the name of the class you have data member and member functions, right? So how the blueprint looks like. Right. So for example. Right.

So we are planning to create a class. Let us say class home. Right. So when you consider class home. So you will have.

Right. So like this. So you may have a hall. Right. You may have a bedroom.

You may have a kitchen. So you are planning. Right. So let us say. 40 meter.

So let us say this is 32 meter. Something like that. So we will plan. So that means I am planning the class home. Right.

So now you will have certain functionalities in home. Correct. So let us say pooja room. Some name I am giving. So now you have the functionalities.

What are those functionalities? You have design hall. Design hall is a functionality. Right. So write it.

Or you may have, before that you have certain data. Right? Member data. Length. Breadth of the complete house.

Right? So length of the hall. Breadth of the hall. Length of the puja room. Breadth of the puja room.

Right? So these are all considered as the field objects. And then I am going to define the design hall. So what is the total area of Right?

So I will write some code over here. Let us say I am going to find out the area of the hall or area of the kitchen. Right? So design kitchen, design pooja room. So this we consider as a class.

Right? So starting with classroom. So I am ending with the semicolon. Right? So this I considered as the example of a class.

So it means you have the blueprint. Right? So this blueprint has several objects, several member data and the functionalities. So this is how we are building the home. So exactly you can think when you are building the home, physically when you are building the home, what are all the plans that you are going to make.

So in a similar way, when I am writing a class, what are all the member data I require? What are all the functionalities it requires? So these things you have to think and then make a class so now we will start working on some programs right based on the classes and objects so we have studied the extended right the definition of classes and objects so now you know so we will start right so as i am keep on saying so your prerequisite is required you should have the knowledge on the basic C programming before object oriented, right?

So that is what the idea of this course. So with that, we will start with the first example, right? So, you know, as include IO stream exactly like a procedural oriented you are doing. And whenever you want to write this C out C in statement, you are using, using namespace, the standard function STD, right? So now I define a class student, right?

So class students. So these two we had seen, right? Student is the class. Student is the name of the class. And then here first we are using the access modifier.

So right now no need to worry about this. Alright. So this is called the access modifier. This syntax we will worry slightly later once I introduce the access modifiers. But before that we are using the data members.

So the first data member is int id. Alright. So id is the data member. And string, you have another member data, data member, whatever way you can call it. A string name is a data member, right?

String name. So you are using only data members here, right? So you are using only data members here. So now your class is ending over here. This is where your class is ending, right?

So now you go to the main program. So here, line number 10, you have the main program. Right. So in this main program, you are student S1. Right.

So that means S1 is an object. You are creating an object. Creating or you call it as instantiating an object. Under the class, student. So now you have.

Right. This object S1. Invoking the data member ID. Right. Right?

And ID is assigned the value 18. Right? The ID is assigned value 18. You might have written in your procedural orienter int a is equal to 10. Simply you write.

So, in a similar way here, the object has to instantiate. Right? The object has to be invoked. Right? The object has to be invoked.

The object has been instantiated over here. S1 is an object. So this object S1 dot. So the dot operator. Right.

So, the dot operator bridges between the object and the member data or object and the member functions. Right. So here, S1 is the object name. So the name of the object. Then, we use the dot operator.

So this is your dot operator. Right. On the right-hand side, you have member data or member function. So, for example, here is member data. Right now, we are not using any member function in this code.

So the member data is id. So, I call it a member data or data member. Whatever the way you call it. So then, I am assigning some value. Right.

So this syntax exactly we are using this syntax. Exactly we are using this. So dot operator. And then you have ID. So that means the relation between the object name and data member.

So, the dot is operating like a bridge. And then, it takes the value 18. I hope it is clear to everyone. Now let us consider line number 13. Alright.

So S1 dot name. Object name. Dot operator 18. Another data member called name. Right.

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

class Student {
   public:
      int id;//data member (also instance variable)
      string name;//data member(also instance variable)
};

int main() {
    Student s1; //creating an object of Student
    s1.id = 18;
    s1.name = "Virat Kohli";
    cout<<s1.id<<endl;
    cout<<s1.name<<endl;
    return 0;
}
```

The name is a string. So I can use a string, Virat Kohli. Right. I can use a string, Virat Kohli. So now, in the next two steps, we are going to write an output.

All right. So the output is cout s1 dot id. So suppose if you write here cout a. Right. In the procedural oriented, you might have studied. When you write int a equal to 10 and cout you know the value will be a. The value will be 10.

For A, right, so that you know, the output is 10 for this code. So, in a similar way, you are printing out s1.id, and another one is s1.name, right? So, when you run this code, we will get the output 18 and Virat Kohli, right? So, when you run this code, when you compile and run this code, you will get the output 18. So, s1.id is 18.

Right. s1.name is Virat Kohli. Right. So this you are writing using the class and object. So the first program that we are using is the basic program.

Right. So the name of the class is student. And then you are instantiate an object s1 under the class student. And then how to give the value for this. So we are just using the member data or data member.

Right. So two data members we have used here, ID and name. And then you are printing. So it is like reading and writing. The initial program.

Kind of allowable. Right. So reading the value. And writing the output. Right.

Assigning the value. And giving the output. Writing the output. So we will see one more example. Right.

So let us consider the class box. Right. Let us consider the class box. So here you have the member data length, breadth, and height. Length, breadth, height, and the name of the class end over here with the semicolon. So now I create two objects right in the last program, we created one object, and here, let us assume we create two objects, box1 and box2 these are all two objects, so now it's a usual one double volume equal to 0.0, all right so you are using Variable call volume whose data type is double.

```
17      // box 1 specification
18      Box1.height = 5.0;
19      Box1.length = 6.0;
20      Box1.breadth = 7.0;
21
22      // box 2 specification
23      Box2.height = 10.0;
24      Box2.length = 12.0;
25      Box2.breadth = 13.0;
26
27      // volume of box 1
28      volume = Box1.height * Box1.length * Box1.breadth;
29      cout << "Volume of Box1 : " << volume <<endl;
30
31      // volume of box 2
32      volume = Box2.height * Box2.length * Box2.breadth;
33      cout << "Volume of Box2 : " << volume <<endl;
34      return 0;
35  }
36
```

This is the usual one you might have used in procedural oriented programming. The basic C or basic C++. So now you have box1.height = 5.0. Box1.length = 6.0.

Box 1.breadth = 7.0. This is the specification of box1. The second one is a specification of box2. 10.0, 12.0 and 13.0. Right.

So now volume. We are calculating the volume. Volume is box1.height. Into box1.length. Into box1 dot breadth.

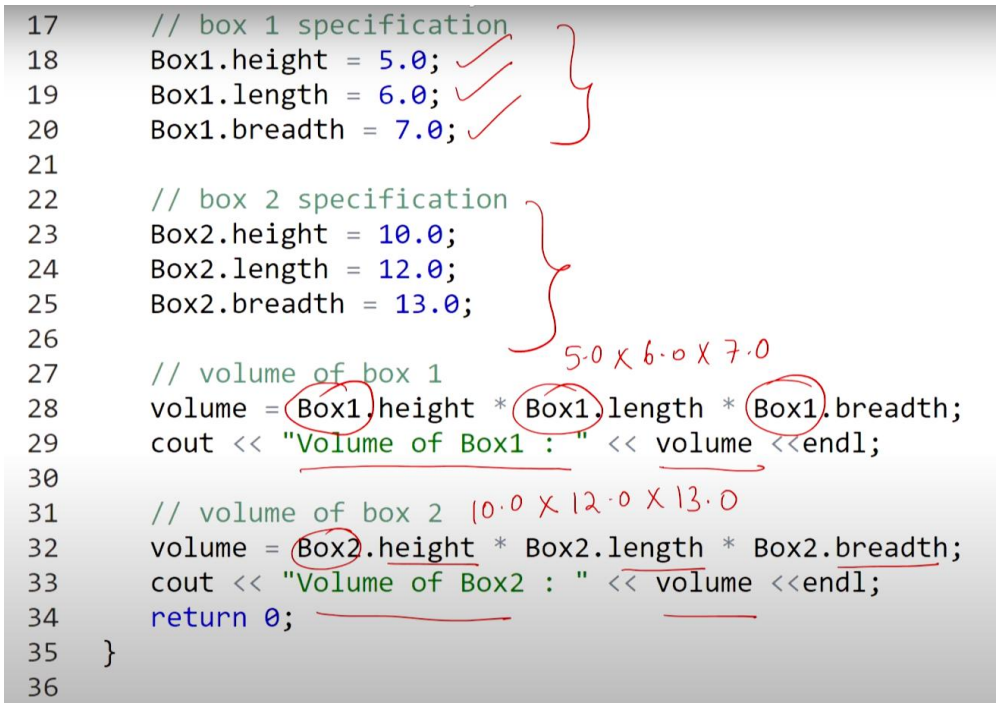
Right. We are calculating the volume. Correct. So if you look at the previous slide. So we define volume in mean.

Right. And how we can access this length, breadth and height. Now you can see the object box1, right, which is invoking height. Again the object box1, which is invoking length. Box1, which is invoking breadth, right.

So we know that, box1.height is 5.0. So 5.0 will be multiplied by 6.0 will be multiplied by 7.0, right. So this will give you the volume. Similarly, you can see box 2 which is invoking height, length and breadth. So that means here in this case 10.0 will be multiplied with 12.0 and that will be multiplied with 13.0, right?

So you will get the output. So the output you are writing volume of box1, right? What is the volume, right? And volume of box2, what is the volume? So, you can run the code now.

So, this we have written defining a class name of the class is box and you have objects box1 and box2 correct. So, you are using one variable volume right whose data type is double and then right. So, box1 is invoking and the corresponding height is 5.0 we are adding a simple code. So later stage how elegantly we can write this. That we will see.



So box1.height. Box1.length. Box1.breadth. It is taking the value. Assigning the value.

Similarly box2. Height, length and breadth. So now when I multiply these three. Respective box1 and box2. We will get the volumes of box1 and box2.

So when I run the code. we will get volume of box1 and volume of box2. So in the next class what we can do so we will take one compiler and run the code. So right now in the first introductory classes we had seen right so how we can define right the classes and objects. So in fact the basic overview we had seen in the last class and today's class

We are more focusing on the objects and classes. And now you know very well what do you mean by objects and classes. So under these objects and classes we have written two simple code. So I have written the code and then we have got the output. So in the next lecture what we can do.

So we will take a compiler. You can use any compiler. So since All of you have done your procedural oriented programming right or basic C up to let us say

structures and unions. So you can start running whatever the compiler that you are using.

But in the next class what we will do. So the two programs that we had seen. So we will run with a compiler with the help of a compiler. So in this lecture we have more focused on the objects and classes. And we had seen some of the examples, right?

How the objects and classes are being defined. So we have taken the fruit is considered as a class and the apple, pears, oranges, they are considered as the objects, right? So in a similar way in this program, so the box is considered as a class, right? And the box1 and box2 are the objects. So the two examples, so various examples we had seen without program.

But two examples we have seen with program right. So the next class we will begin with running these two examples. So with this I am concluding this lecture. Thank you very much.