

FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture01

Lecture 1: Introduction to Object-Oriented Programming

Welcome to the course fundamentals of object-oriented programming. I am Professor R. Bala Subramanian working as a professor in the Department of Computer Science and Engineering at IIT Roorkee. So the first lecture or the first week of the lecture is based on the introduction to object-oriented programming. So here in the introduction to OOP, right, so it is assumed that you all should know procedural-oriented programming language. So either C or Pascal language.

So starting from the basic programming, let us say arrays, pointers, unions and structures. So, it is assumed that these are all the prerequisites. So, in today's lecture, we are going to see the introduction to object-oriented programming. What are all the benefits of object-oriented programming? We will see the historical overview of OOP.

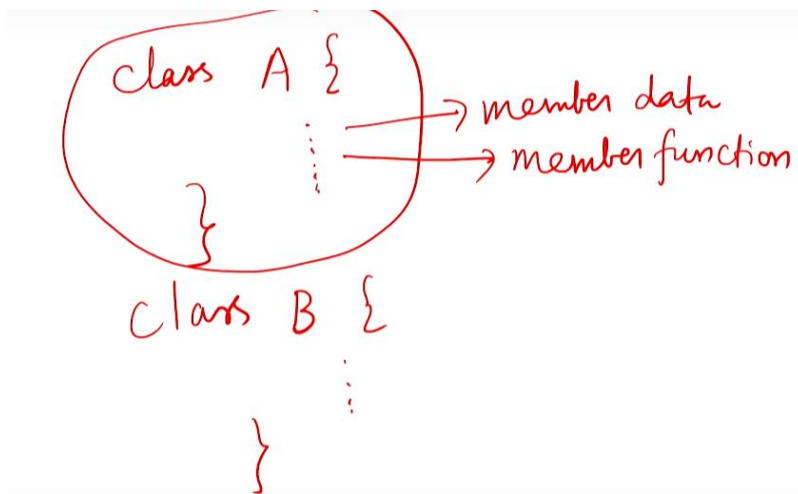
and we'll talk about objects and classes. So you might have studied procedural-oriented programming, so the procedural-oriented programming depends on the functions, whereas here, in the case of object-oriented programming, it is entirely based on the objects. So, for example, if I say animals, which I consider as a class right here in OOP, The programming is structured based on the objects and classes. Right. So, let us consider animals as a class. So under animal, you have, let us say, cat, dog or even tiger.

Right. Elephants. These are all the objects. So this is how the classes and objects are being. We are going to in fact technically we are going to define.

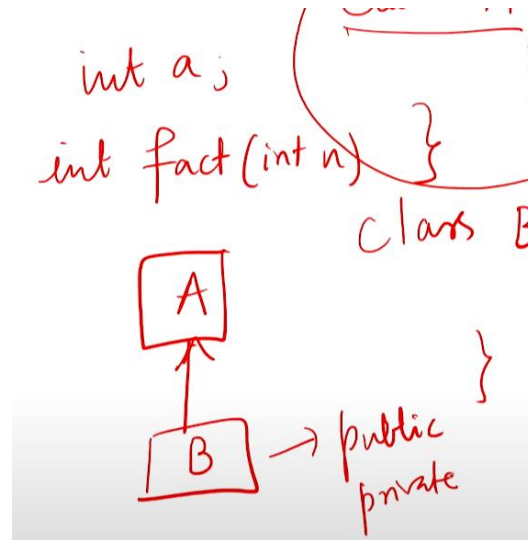
So the objects and classes. Right. So, when I am talking about objects and classes in object-oriented programming, they are like the grammar of the language. So you definitely require grammar in the language so that the communication will be easy. So similarly, the object and classes enable

communication with other devices to achieve specific tasks or goals in the programming paradigm.

So now, what are all the advantages of object-oriented programming? The first one, modularity through classes. So when you are writing the entire program using classes, all right, so elegantly you can handle. So you are dividing there by parts, right? And then finally you will have a main and when you look at the main, so whenever you are writing, suppose we are going to instantiate an object using the class, right?



And then we will be calling either the member data or member functions so you can find elegantly you can use and the program looks much simpler when you are using different classes so next application is code reusability suppose one user let us assume that he is using class A and then with the help of member data and member function, assume that he or she has written the code. So now the task is given to you. So you have to use another class.



Class B, right? So when you are writing a class B, so the question is, is it possible to use, right? The member data and member functions, or let us say the attributes of class A, right? So which is possible? That is what it is called reusability.

I mean, practically it happens. So some user might have done with class A and class, let us say, he or she has used many member data and member functions. So can this class B, can I incorporate this? Yes, you can do that. So that is called the code reusability.

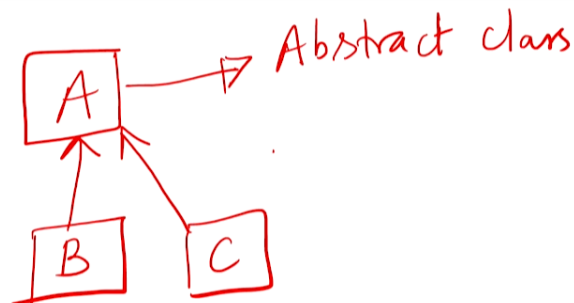
That is the advantage of OOP. You can have the code reusability. The third point is encapsulation, right. So the encapsulation in the sense, so whenever I have a class like this, the one you are seeing over here, so this class contains the member data and the member functions, right. So this contains member data and you have member functions.

So you might have studied, right, for declaring a variable. Right. What is the meaning of member data? So if I declare `int a`. Right. This is a member data.

Or if you are writing a function `int`. Let us say factorial of `int m`. Right. This is a function. So one is called member data and member function. So it is like a capsule.

So class A contains in fact many member data and many member functions. Right. So this is called encapsulation. The next concept is the inheritance. So the meaning of inheritance, as I said, in the case of code reusability, the class B can inherit the property of class A. So we are going to see the access modifiers like public, private.

So some of the data may be public to all the classes. We are going to call it as a derived classes. So class A. is a base class and class B is a derived class. So you call it as inheritance.



I can have several. Class C can inherit class B. Class C can inherit class A. So you have a base class and then classes like class B, class C are all going to be derived classes. So assume that we have only two level class. You have class A and class B is going to derive class A. So that means here, this class B can use the complete properties of class A. So assume that we are going to have access modifiers like public, private, etc.

So these are all, anyway we are going to study and this is called as the inheritance. So the next concept is the abstraction. Alright? So we talked about the inheritance.

Class A, right, which is the Assume that it is inheriting class B, right? So class B is inheriting class A, right? Let us assume. So in this case, right, so I can make the base class as a complete abstract, right?

So you know, some of you might have seen, right, how to write a research paper, right? So in the research paper, you can see title, right? So followed by abstract, the word abstract. So that means in the word abstract you can see the content of the. Right.

Paper in short the researchers used to write. Right. So that means in a similar way in the case of object oriented programming. So if I want to say. Abstract class.

Let us assume that A is abstract class. Right. Let us assume A is abstract class. So here what is happening. I will write.

Whatever the functionality that I am going to do in my program. Right. So you can define a member data. And then you are defining a member function. Right.

You are declaring a member function. Whereas the definition of those functions will go in the derived class. For example I can do it in B. Or let us see one more class is deriving A. So I can define all these functions. functionalities that means the member functions that i have declared in a right so you can only declare that is what abstract so in the research paper so they write i have done this particular algorithm so we have done this particular algorithm so they don't give the details in the abstract if you look at they don't give the details in the abstract all right in a similar way when i am talking about the abstract class so we are going to define let us say Member data and member function without definition.

So those definition will go in the derived classes. So you call this as a abstract class. So this abstraction is possible in object oriented programming. So the next concept is polymorphism. So the meaning poly.

- Modularity through Classes
- Code Reusability
- Encapsulation
- Inheritance
- Abstraction
- Polymorphism
- Data Maintenance and Security
- Design Advantages

```
graph BT; B[B] --> A[A];
```

Handwritten code for class A:

```
int max(int a, int b) {  
    ...  
    return ...;  
}
```

Handwritten code for class B:

```
double max(double c, double d) {  
    ...  
}
```

Handwritten text: "Run time polymorphism", "Late P"

Suppose I want to write let us say maximum of two numbers. right so you go back to your procedural oriented programming so what you might have done so you might have written `int maximum of int a comma int b` right and then assume that you are writing an algorithm right you are writing an algorithm for maximum finding out the maximum of two numbers So, can I use the same max function? So, can I use the same max function for two double? So, let us say the return type is float.

Assume that the return type is float or double. So, let us consider double. So, double the same max function. So, the same max function. So, here you are `double C comma double D`. You are writing another function.

Right only thing is you are using same max function. So let us assume I believe you know how to write the maximum of two numbers. Anyway during when we are writing a programming we will take care. Right now we will talk only about the concept. Right so now if you look you have same function.

Right so when I am calling this. Right so when I am calling this assume that I am calling this in a main. Right. Assume that I am calling this in a main program. So when I am calling this in a main program, I used to call like this, let us say maximum of 7 comma 8.

Right. So putting in some, let us say `int m1`. Right. Another statement, let us say `double m2`. So which is maximum of, let us say 77.2 comma 99.3.

Right? So assume that you are calling these two functions inside the main. Right? So when you are calling this, we know that when you are calling this max, you have passed 7, 8 as an integer. Or you take `int a` is equal to 7 and `int b` is equal to 8.

Pass maximum of a, b. You can do like that also. Right. So then what happens in the compiler. Right. During the compile time.

So during the compile time. So this max function will call this function. Right. So this max call will call this particular function. And then you are writing the maximum of two numbers between the integers.

And right. That will be return. And that will be return. Assume that you are returning something. Here you will have the return function.

So this will return something and this return value will be stored in M1. And the next line, let us assume that you want to find maximum of 77.2, 99.2, correct? So same function name. So now your parameters, right, are the values that are double, right? Double or float, not a problem, right?

So that means this will call this particular function. So the compiler will decide during the compile time. Both are same function name but during compilation time this will be decided. So therefore this is called the compile time polymorphism. This is called the compile time polymorphism or you also call this as a early binding.

So this is the name as early binding. Anyway in detail we are going to study all this. This is just the introduction class. So you can see. This is called early binding or you call this as a compile time polymorphism, right?

I hope it is clear. Both are same name, right? Similarly, we have runtime polymorphism. What do you mean by runtime polymorphism? So let us assume that class B is a derived class and class A is the base class.

Right? So let us assume your class A contains this maximum function. The class A contains this maximum function. So you have assumed that you have defined all your int A, B, etc. You have member data and this function which is available in class A. So let us consider the second function which is in class B.

Now in the main we are calling. So assume that we are calling. So we are going to instantiate an object and that object is invoking the function. And when it is invoking the function, which one will be called now? So similar to what we have studied in compile time polymer.

So you have base class and derived class. As I said, in the case of inheritance, the B can use all the property of A. If it is, let us assume that publicly inherited. Correct. So in that case, when it is publicly assumed that it is publicly inheriting, and then when you are defining an object or you are instantiating an object in the main program, and that object is invoking, let us say, the function max. Now the question is, which one will be called?

So this will be decided during runtime. Right. So this will be decided during runtime. Whereas previously we saw polymorphism, it is early binding. So that

will be decided during compile time whereas here in this case this will be decided during run time.

So therefore this is called the run time polymorphism right we call it as a run time polymorphism or it is late binding. Right. So the meaning is I can use the same function name. You may ask one more question. So can I use the same function name like this `int maximum of let us say int l comma int m comma int n`. That means maximum of three numbers.

Yes, you can do that. Right. The same function name. Now this function will find out the maximum of three numbers. You can use it in the same program, whether it is a compile time polymorphism or it is a runtime polymorphism.

You can do that. Okay. So this is about the introduction of the polymorphism. I hope you understood. So now the hoop has another property, data maintenance and security.

So I said when you use, let us say, class A, class B, class C. Right. So there is a chance that you may use, let us say, same member data, same name. Right. Same possible. Right.

Similarly, we already talked, same function name. So in that case, how it is being handled, right? So that maintenance property is very well defined when you are using classes, right? So this will not be a problem. Unlike the procedural oriented programming, suppose in the program, if I define, let us say `int a` two times, right?

Procedural, you can check it, right? You have already studied procedural oriented programming. So let us, you can try `int a` two times. So you know that it will give an error, right? So whereas in this case, I want to use it in class a, not a problem.

Class b, the same `int a`, right? So which will be maintained, right? So which will be taken care. And similarly, when I am talking about the security, so security in the sense, so this variable `int A`, you can allow only class B can use, but class C cannot use, right? So we are going to study about the access modifier, the public, private, so that, right?

So any function you want to share with, let us say B can inherit, share means B can inherit A, right? So that means B can use, but C cannot use, or you cannot use this data outside the class. Right? You cannot use the member data outside the class or member function outside the class. Right?

That contains member data and member functions. So you are allowing B can use member data and member function. Whereas outside this or you can have only A can use. Yes it is in fact by default. Suppose you are not using any inheritance concept.

Assume that whenever you are having the member data and member function. So this cannot be. I mean you can have the access modifier name. Let us say private. So we are going to use the access modifier name private.

Anyway this will be explained in detail when we are going to talk about this inheritance. So when I put private int a, so this can be accessed only inside the class, let us say a, right? So that is called the security. That is called the data security. So the data maintenance and data security is very well defined or available only in object-oriented programming.

So this is one of the beauty of the object-oriented programming. So the last concept, you have design advantage. So in the procedural-oriented programming, most of you have studied flowchart, right? So the flowchart, diagrammatically you are representing an algorithm. Correct?

So in a similar way, so here we have, right, object-oriented analysis and design. OOAD stands for object-oriented analysis and design. So when you want to explain your complete code, so where you are going to use a member data, member function, or when you want to explain about the inheritance, abstraction, or polymorphism, you are drawing the diagram. So there are various diagrams in object-oriented analysis and design. Right?

So the design advantage, All right. This is overcoming whatever you studied the flowchart concept in procedural oriented programming. So analysis and design in the sense you can analyze. So you can show it as a diagram.

Right. User A can show to user B as a diagram. There are various diagrams. This is a separate course called object oriented analysis and design. Right.

So here. So you can find various diagrams. So those diagrams, so one can analyze and you make a design, you call it as object oriented analysis and design and you can explain, A can use or A can explain the complete design and in fact analysis of the algorithm to B or the program to user B. So these are all the main advantages of object oriented programming. I hope it is clear for everyone. So now we have seen the

introduction of object-oriented. So, how did this object-oriented programming evolve, right? So initially, in the 1930s, right, scientists or programmers, so they have The mathematically oriented paradigm. So, the programming paradigm is considered as mathematical oriented.

So what have they done? The lambda calculus. Principles of programming. In the course called principles of programming. I mean, one can learn what lambda calculus is.

It is complete mathematics. And the Turing machine. So, your programming is based on the 1930s. It is based on the lambda calculus and Turing machine. So in 1966 when Alan Kay worked for his PhD.

So, at that time, he only coined the word called object-oriented programming. Right, 1966. So there are certain applications like Sketchpad. So this was developed during 1961 and 1962. So this has got the inspiration.

to go for object-oriented programming, right? So that is the first application. So, the Sketchpad application is useful for scientists or programmers to think about the object-oriented programming paradigm, right? So then you can see, so 1965, so they had a language. So before 1966, Alan Kay, so they had a language called Simula, right?

So which is considered as the object-oriented language. So, they have used classes, subclasses, inheritance and the virtual method. So, whatever be the object-oriented programming concepts like classes, subclasses, all these inheritance. So, these concepts they have used in Simula. But in 1966, Alan K. only, so during his Ph.D.

right when he was in a graduate school so he coined the word or he coined the term object oriented programming so now as i said when i am talking about the object oriented programming so let us have right the languages like c++ and java

right we will more focus on how to write the code So, assume that you know the basics of C, right? So, that is what the main objective or prerequisite of this particular course. So, assume that you are well familiar with the basic C, right? So, without object oriented concept.

So now the idea is, with the help of C++ and Java, in this course, we are planning to give the flavour of this object-oriented programming. So that means the programs are designed based on the objects and classes, right? So when I define an object, so for example, here you can see cylinder objects, right? So these are the cylinder objects. So that means I can say the cylinder as a class, right?

The cylinder as a class. And you can find various cylinders. So these are all the objects. So this is object 1, 2, 3, 4 and so on. So these are all various objects.

So, a cylinder is a class, and these are all the various objects. So now I can define what do you mean by object. So, an object is nothing but an entity that has, in fact, encapsulated. Both the state and behavior, right? So, state nothing, but it refers to the data, right?

State is nothing, but it refers to the data. And behavior is nothing but the functionality. So whenever I said, right, in the previous slides, whenever I defined class A, I talked about member data and member function. So state is nothing but member data, right? And behavior is nothing but the member functions, right?

So whenever I am having, let us say, I have member data and member function. I can hereafter simply write data and function. It is understood, right? Now I am writing, but in the future when I write data and function, it is understood. So various data and functions, right?

So now assume that I define a class, right? So state and behavior. So this is the state. Member data is a state, and a member function is a functionality, right? The behavior, correct?

So now, Let us assume I'm coming to the main program. So, for the main program, I can define a, let us say small a, correct? So I instantiate this object, small a is the object. What about capital A?

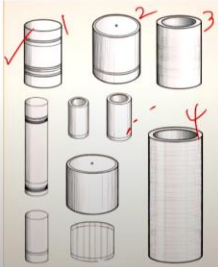
Capital A is a class. So this object can be instantiated, right, during runtime. So when I am writing like this, so we are going to see some more syntax, right? So right now, we do not need to worry about that. So when I instantiate like this, so we are writing, you know that we are writing int l, right?

So, in a similar way, this is not the exact class; it is a wrapper class. So now I have capital A as a class and small a as the object. So, like this, we are going to define, and you call this as, when you do like this, you can say that you have instantiated an object small a, right? Under the class capital A. Right. So this can be done during run time.

Classes and Objects in C++


- In the object-oriented programming approach, programs are designed using objects and classes.
 - An object is an entity that encapsulates both state and behavior, where the state refers to data and the behavior refers to functionality.
 - Objects are instantiated at runtime.

<https://www.imagine.art/dashboard/tool/text-to-image>



Class: **Cylinder**

Class A {
✓ member data;
✓ member function;
}
A a;



Right. So when you look at the cylinder. Cylinder is a class. So, under the cylinder, you have several objects. So whatever objects that you are seeing.

Under cylinder. Right. So, they are all considered objects. And the class. The main class is.

cylinder, right? So, this is how the classes and objects are defined in C++ or Java, all right. In particular, object-oriented programming is completely based on classes and objects. So, in this lecture, we saw the basic introduction to object-oriented programming and very little introduction to classes and objects. So, in the next lecture, we will talk about how you are going to instantiate an object, and we will also look at the details of the objects.

Thank you.