FUNDAMENTALS OF OBJECT ORIENTED PROGRAMMING

Lecture20

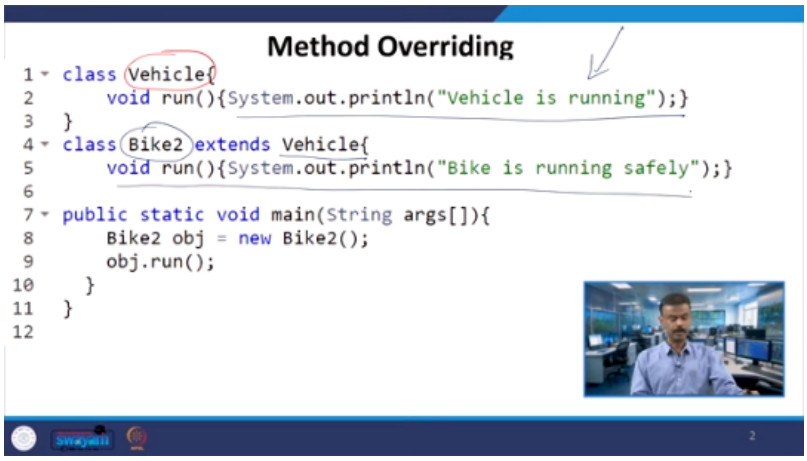
Lecture 20: Method Overriding

Thank you. Welcome to lecture number 20, Polymorphism. So, in the last class, we talked about method overloading. So, on the continuation in Java, we are going to study method overriding. So, here you can see an example.

So, you have the superclass. So, in this, in C++, we call base class, right? So, here, let us consider a class vehicle, all right? So, under class vehicle, we have one method, void run. All right.

So, run you have system dot out dot printer and it is printing vehicle is running. Now, you have subclass let us say bike 2. Right. So, your superclass vehicle bike 2 which is extending vehicle. All right.

Which is inheriting vehicle. All right. And then this also has void run. Right. So, this also has void run.



So, now when you are creating an object OBJ under bike 2, you are dynamically allocating a memory with constructor bike 2. the default constrictor right so now when the obj that you are creating an object under bike2 class when obj is invoking the function run invoking the method run right so this one will be

overridden this one will be overridden and this one will be printed bike is running safely will be printed and you can see when you are running the code you will get the output bike is running safely Right. So this is called method overriding. Right.

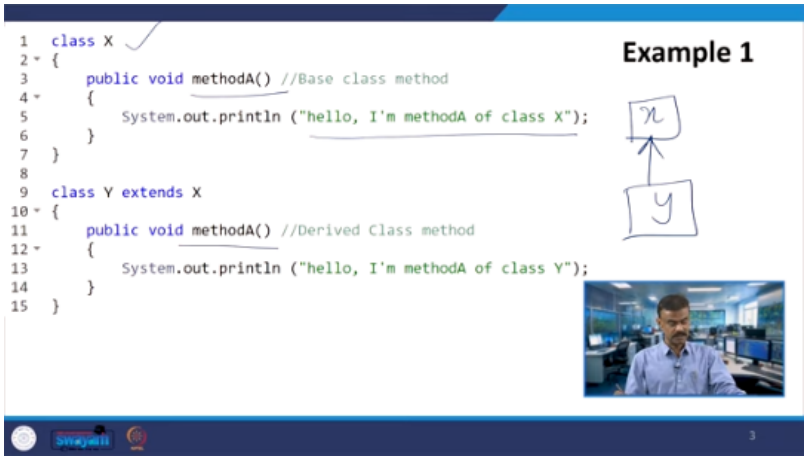
So now we are going to see various examples. So you may ask the question. So when I ride bike to OBJ equal to new. So can I do vehicle or the other way around vehicle under vehicle? I have OBJ.

Right. So all these cases we'll discuss. And when you have same method name run in both superclass as well as subclass. So you can see that which one will be invoked. So, this will be decided during runtime all right.

So, whereas method overloading or function overloading that we had seen. So, which function should be called will be decided during compile time. So, they are called compile time polymorphism whereas this is an example of the runtime polymorphism right this is an example of the runtime polymorphism. So, now we will take one more example right. So I have a base class X and under base class I have one base class method A. Right.

So here I am printing system dot dot dot printl and hello I am method A of class X. It is a very right trivial print statement. Right. Now class Y is extending X. So X is the base class. I mean to say it is a superclass where Y is the subclass and which is extending X. So, now you have the same method name method A right in the derived class.

So, now you have system dot out dot printl and allow I am method A of class Y. So, it is printing. So, you have same method name method A and in the derived class also you have method A. So, now you have an object right. So, assume that you have an object 1. Object 1 is under X class and then you have constructor X right. So, the reference and object you call it as a reference and object X first time you are seeing right reference and object X. So, can we have a difference reference and different object?



Yes, it is possible. If you look at line number 20 OBJ 2 is an object right object 2 whose class is capital X right. So, dynamically allocating a memory with a constructor is Y right. So that means you call this as X is a reference but Y is the object. X is a reference and Y is the object.

Now OBJ1 when it is calling the method A. OBJ1 which is under X. Constructor is also X. So therefore this function will be called. So you have an object OBJ1 in class X. And then this will call this particular function. So although I am method of Right I am method A of class X will be printed first. So this will be your first output.

And then you see obj2 under X. Again it is also X. But now it is calling method A. Right again method A. So now your reference is X but object is Y. So therefore the subclass method A will be called. So this will be called and then you will get this as a second output. So now if I run the code. I will get hello, I am method A of class X will be one output and hello, I am method A of class Y is an another output, right. So, this is how you can see that when you are creating an object, right, when you put the same reference and object that is a base class, all right.

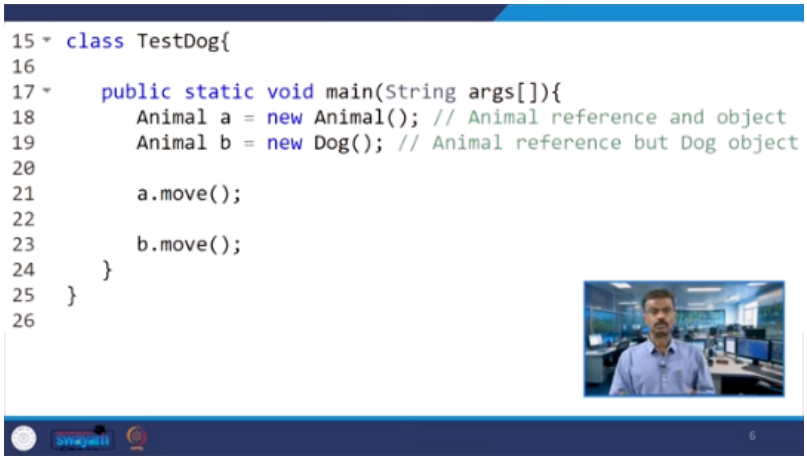
So, in fact, class Z which is nothing but the driver class. Why I am saying driver class? You have main program. So, you have an object under class X and then you can see the constructor is also X. And the second one you have X reference right object 2 under X you call it as X reference, but the constructor is Y you call it as Y object. So, therefore, when OBJ1 is calling the method A, it is calling the base class or you can call it as a superclass.

And when OBJ2 is calling the same method A, it is calling the subclass. So, when it is calling the respective classes, first output you are getting, hello, I am method A of class X. And second output, hello, I am method A of class Y, right. So, this is what you are getting as a output. And the next example, okay, almost similar example, only thing we have to focus on the line number 19, all right.

So, when you are creating an object in one class and one is a reference and another one is object, that means your class is animal here and the constructor is dog. So, then in that case, right, how it is being called and what it is called. right. So, here your class animal, you have one member function called move, one method called move. So, system.out.printl and animals can move, right.

So, this we are writing, right, this particular function and next one class dog which is extending animal, right, the class dog which is extending animal. So, now this is also having a move method, Right. So which is printing dogs can walk and run. Right.

Which is printing dogs can walk and run. So now you have class test dog. Right. So this is a driver class and you can see A is an object. Right.



A is an object whose class is animal. Right. So it is dynamically creating a memory with animal as the constructor. Right. So that means animal reference and object.

So both are reference and object. Whereas in line number 19 you have an object B right whose class is animal right but when you are talking about the object it is a dog object right when you are talking about the object it is a dog object. So in that case right you have a object B whose class is animal that means the

reference is animal but the object is dog right. So now the object a is invoking move right object a is invoking move so you call over here when the object a is invoking move so this particular function will be called right so animals can move will be printed and b.move it is under doc so docs can walk and run should be printed so when I run the code right so I will get the output like this so animals can move

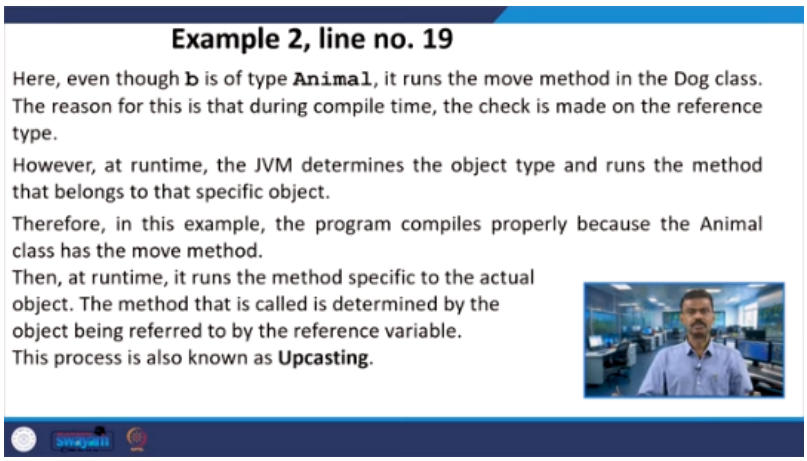
And docs can walk and run will be printed, right? So this is what exactly happening in this case. So now if you look at line number 19, right? So you have an object B. So that object B is under animal, right? So object B is under animal, right?

So when I am calling the move method, it is going to the doc class, right? So the reason is, so during a compile time, right? So the check is made on the reference type. Right. So what is your reference?

Reference is animal. So during compilation time, B is an animal. Right. So during compilation time, B is an animal. But during the runtime.

Right. So the Java virtual machine during runtime, JVM stands for Java virtual machine. So during the runtime. So this particular object type. Right.

So in fact, the Java virtual machine, which will determine the object type. And it will be running the method that belongs to that specific object, right? So, in this particular case, what is happening? So, during compile time, it is animal, right? So, but during runtime, so it runs on the method specific to the actual object, right?



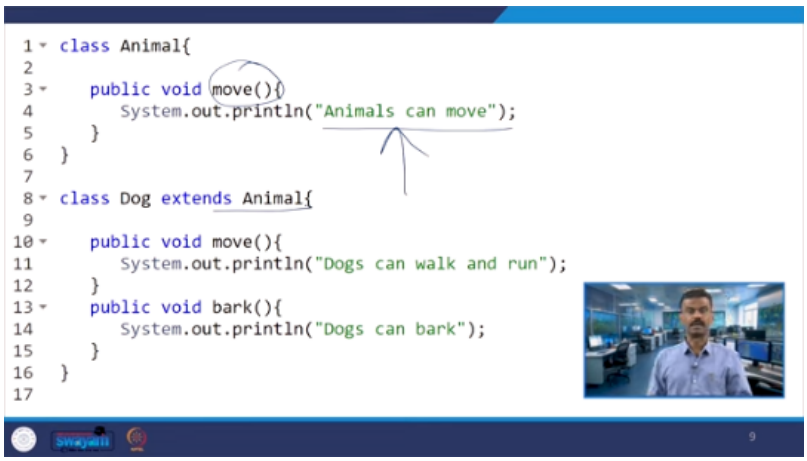
So, what is the actual object? Actual object is nothing but dog. So, during compile time, it is an animal and during runtime, I mean the reference object is, I mean the object is doc, right. So, it will go under doc.

So, under doc, what is the move function? So, that will be printed. So, therefore, during a runtime, the Java virtual machine determines the object type And it is running the method that belongs to that particular object or specific object, right? So, since during the runtime it is happening, during compile time animal, but during runtime it is becoming dog.

So, this concept is called the upcasting, right? So, the method that is called is determined by the object being referred to by the reference variable and the process we can call it as upcasting, right? Okay, so during compile time, the object B was an animal and during runtime, so the object is a dog. So, that will be called. So, this concept is called upcasting and here in this case, this is upcasting.

So, during compile time, it is an animal, but during runtime, it is dog. So, now we will see one more example. So, assume that we have class animal and under this class, you have animals can move. Right. Under this class, you have function.

You have a method move. So under this move, you have animals can move. Right. So now class dog is extending animal. Class dog is extending animal.



So you have void move. Dogs can walk and run. And you have a bark. Dogs can bark. So there are two methods.

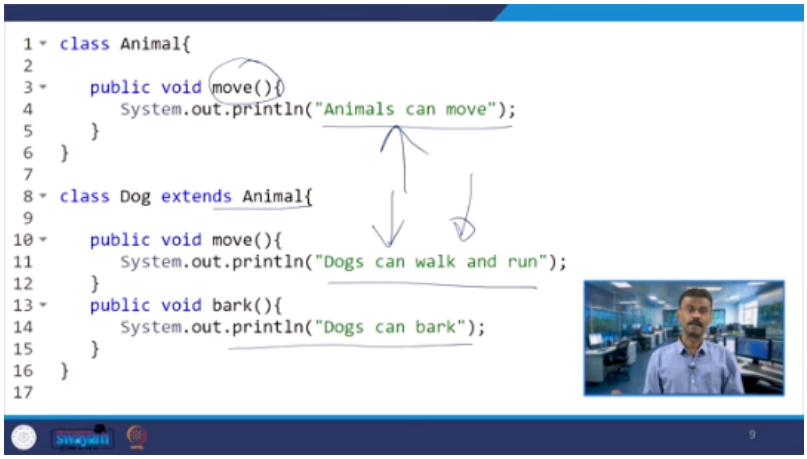
Move method and bark method. So, one is being overridden right one can be overridden right. So, let us see how this is working particularly the casting right one move dogs can walk and run and another one is bark dog can bark. So, now I have object A under class animal right. So, whose constructor is animal and I have another object called B whose class is animal and I have dog object.

So that means the upcasting is being done over here. So now I call the function a.moo. So when you have a.moo, it the moo, when a is invoking moo, so animal moo will be invoked. Animal a will be invoked. Animal moo will be invoked.

And when b.moo, when b is invoking moo, so you can say that dogs can walk and run will be printed. But When I call b dot bark, can you guess what will happen? Yes. So when b dot bark, when b is invoking bark, right?

So in that case, you have function, right? You have function or you call it as a method. You have a method at the subclass, but there is no method in the same name, right? So same name is not available in your superclass. Right.

So you have to define when you are doing the casting. Right. You have to be bit careful. So you have you have to have this bark function, the bark method in both the classes. Otherwise, your casting will not work.



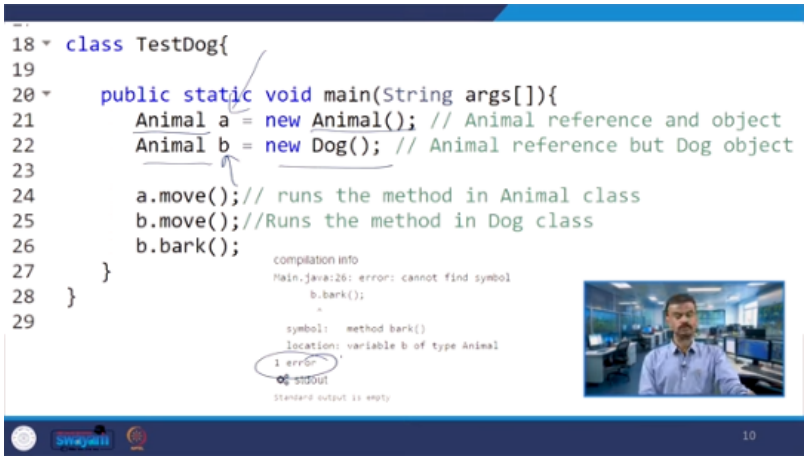
Right. So in this case, you will get an error. You should get an error. Right. So this is an error.

So B dot bark. So what is happening it is immediately going to this superclass right and then bark is not there so then and moreover right if I put it then I can overridden all right. So when you want to do the overridden so both line number

24 and 25 work properly right because move method is available both superclass as well as subclass right. So whereas in the next method We had some problem.

Right. The reason is we do not have this method in the superclass. So, therefore, we are getting this particular error. All right. So, you are getting the error.

So, how do you rectify the error? So, you have to define the bark method in the class animal. And then it will be overridden. Right. And then it will be overridden.



So, these are the examples of the overriding methods. And now we will see one case study. Right. So the case study is based on overriding and we are going to use it for bank account management system. Right.

So we will build a simple bank management bank account management system. Right. Let us consider a simple bank account management system. So with the help of method overriding in Java. All right.

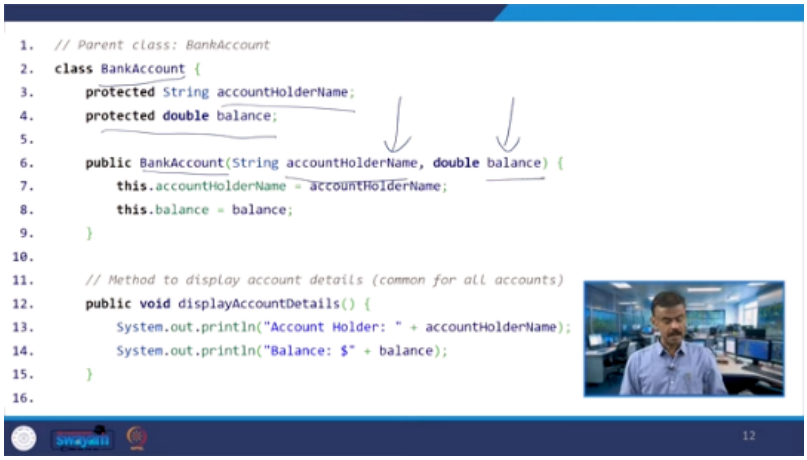
So let us assume the system has different types of accounts. So let us assume you have the savings account and the current account, right? So now each account type may have a different way of calculating interest or applying service charges, right? And we will overwrite these two functions or these two methods should be overridden. Calculate interest, apply service charges, right?

So, these methods we are going to overwrite. So, we will see how we are going to overwrite this. So, now assume that class bank account assume that you have class bank account. So, under class bank account you have the protected string account older name. We already have seen protected access modifier right or access specifier.

So, you have One more member data protected double balance. Right. Protected. Your variable name is balance.

Whose data type is double. So here you have one member function. So in fact. Yeah. So you have in fact the constructor bank account.

Right. The constructor bank account. So, it is a parameterized constructor. The parameterized constructor has account holder name and balance. And this dot account holder name is account holder name, the right side.



So, whatever you are passing, so that will be, right, the overshadow will be taken care. So, account holder name you are copying and balance you are copying, right, using this pointer. So, now you have method to display the account details, all right. So, public void display account details. So here you go, system.out.printl and you have account holder.

So account holder, name of the account, account holder name, right? So that will be printed. And balance. So balance I put in terms of dollars, right? So what is balance?

So this is a member function or you call it as a method which will be under bank account. And the next method is a calculate interest, right? So this should be overridden. To be overwritten. Right.

Public void calculate interest. So system dot out dot println. So we are writing. So interest calculation for a generic bank account. Right.

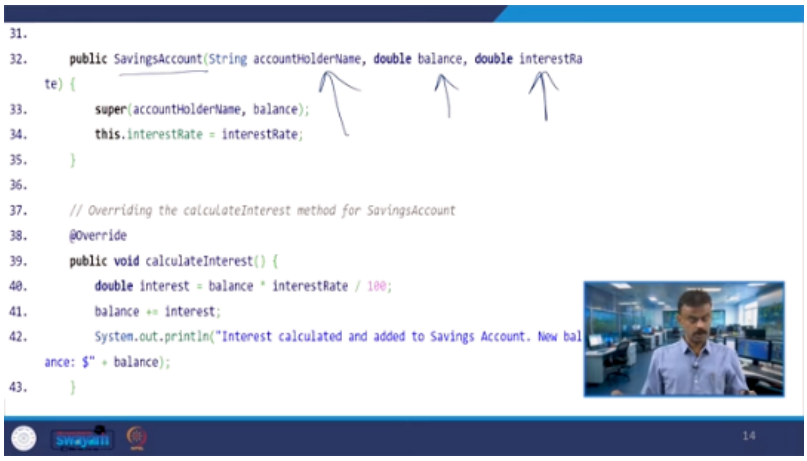
And then you have next one is public void apply service charges. All right. So you have one print statement. So now the class savings account. Right.

You have bank account is a base class. And here you have class savings account. Right. So you have one private account. data, member data, interest rate, right.

So, this is a subclass, savings account is a subclass which is extending bank account. So, you have one private member data, right. And then here you can see savings account. So, which is nothing but the constructor, right. So, the constructor, this is a three argument constructor.

So, you are passing three parameters, account holder name, balance and interest rate. Right. So you have account one older name and balance in bank account. Right. So the bank account is inherited by savings account.

You are passing one more interest rate. Right. So this constructor you have account older name balance and interest rate. So now you have the super. Right.



Account older name comma balance. So that means this super is will invoke this constructor, right? So this constructor will be invoked. Bank account constructor will be invoked.

So account holder name and balance will be copied, right? Next one is interest rate, right? So this dot interest rate is equal to interest rate, correct? So now we are going to override the calculate interest, correct? So it will be overridden.

We have already seen how you are going to have the object and what will be the constructor. So based on that, All right. Upcasting and all. So the method which is available in the superclass will be overridden.

So now calculate interest. So here you have. Right. So calculate interest. So there we put interest calculation for a generic bank account.

So now what is happening? Yeah. Here you go. Calculate interest. Right.

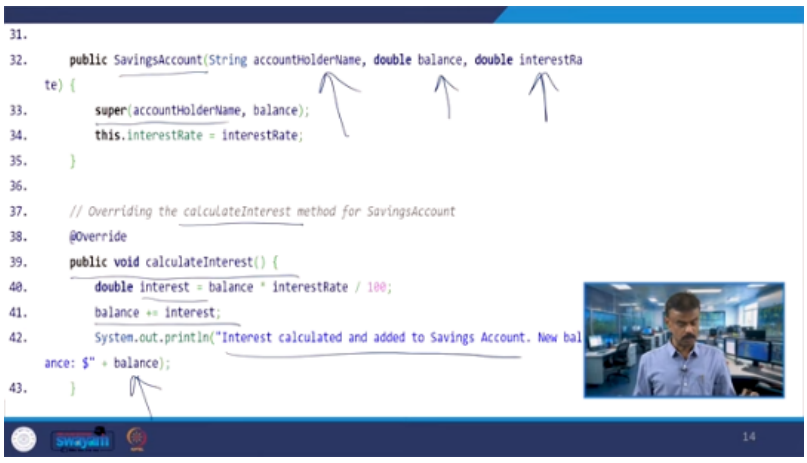
So this will be overridden. Right. The previous calculate interest will be overridden by this one. I mean to say. So, balance into interest rate divided by 100.

So, which you are assigning to interest which is a double data type. And balance is equal to balance plus interest. So, you are adding the interest. Right. After calculating the interest, you are printing interest calculated and added to savings account.

Right. So, you are printing as a new balance. So, which is nothing but the balance. Right. So, balance you are calculating.

Balance equal to balance plus interest. And another one. right, overwritten function, right, overriding function. So, you can call this as a apply service charges, right. So, here you go, apply service charges.

So, there the base class we have written, service charges applied for a generic bank account, right. So, apply service charges. So, here you go, right. So, it is here in this case, it is printing no service charges for savings account, correct. So,



you have which one is the base class here you have bank account is the base class right bank account is the base class savings account is the derived class savings account is a derived class so which is extending bank account right so now current account is extending bank account current account one more right current account so there is one more class which is extends bank account so again you have one service charge data member whose Data type is double and here you have current account which is a constructor. So, here you have account order name balance service charges exactly same. The super means it will call the base class and this dot service charge will be service charge. So, in this class right the current account class which is extending bank account.

So, here you have calculate interest right again it can be overridden I mean you have overwrite. Right. Line number 62. All right. So the base class of this same name can be overwritten.

All right. So here it is printing no interest for current account. All right. And when you look at apply service charges, balance equal to balance minus service charge. Balance equal to balance minus service charge.

So now you are printing system.out.println service charges applied for current account. and a new balance in terms of dollar you are printing balance okay so this is you are having the scenario like this right so bank account is the super class and savings account and current account is inheriting bank account right so when you have this kind of scenario let us go with the main program so in the main program you have object savings right you have a object savings And whose class is bank account, right? So, you have dynamically allocating a memory and then here you have the savings account, right? You have savings account, which is a constructor.

You are passing Pradeep 1005, that means three member constructor. So, savings account. Right. So that is your in fact it is upcasting. So bank account under bank savings is under bank account.

Then it is casting to savings account. So savings account if you look the class savings account. So here you go. So you are passing the account holder name balance and interest. And then here it will be called the super of this.

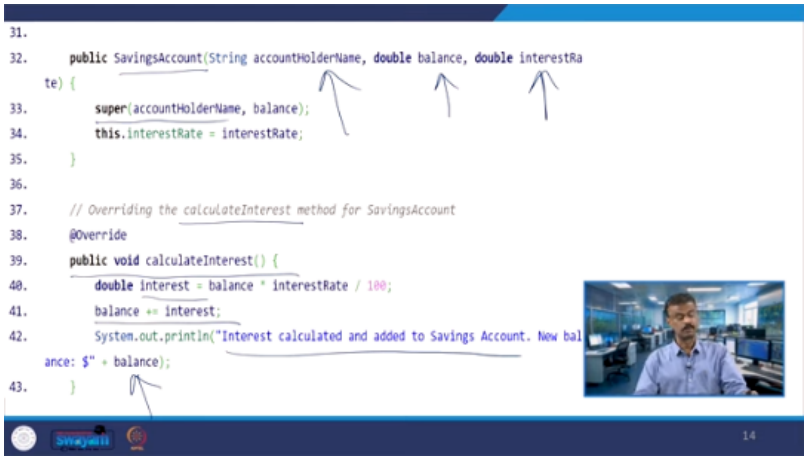
So that means the base class constructor will be called. In fact this we have already studied. This concept. So the base class constructor will be called and then whatever you have passed, I pass the account holder name and balance. So that means this dot account holder name will be equal to account holder name.

That means it is properly assigned. Otherwise the garbage value we printed. So this dot balance is equal to balance. So I am passing the information Pradeep 1005. So now your savings is invoking display account details.

right. So, here it is upcasting. So, here it is upcasting, correct. So, savings account, it is calling the function display account details, calculate interest and apply service charges, right. So, one by one we will see.

Line number 89, when it is invoking display account details, right. So, we know that what is the name of the constructor? Savings account. So, it goes to savings account, right. So, here you go, savings account.

So, savings account, it is calling The super keyword is calling the super class constructor. Passing account holder name and balance. Alright. And then this dot interest rate equal to interest rate.



So we have seen in the case of a super class. So this is the constructor. So this dot account holder name is equal to account holder name. And this dot balance will be balance. Equal to balance.

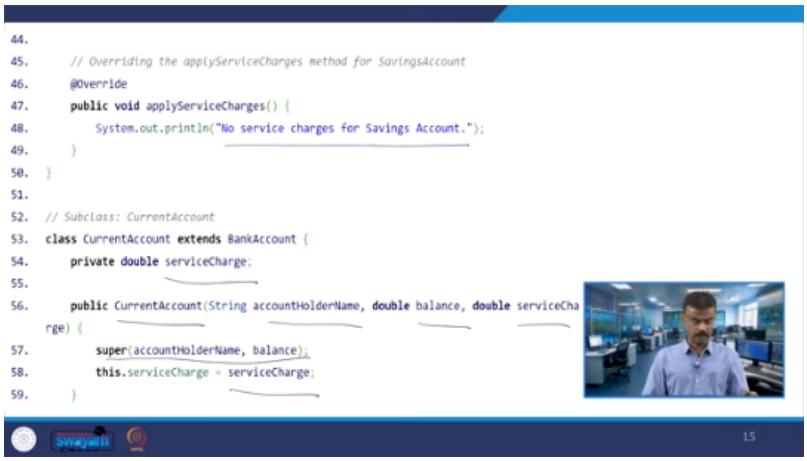
Alright. So in that case. When it is calling. All the information. These three details.

Pradeep. Thousand and five. Will be displayed. If you are. If you are displaying.

These things will be displayed. Alright. So the next one. Calculate interest. Alright.

So when it is calling. This function. Under savings account. This method. Under savings account.

Right. So this method. Under savings account. Here you go. So the method is.



Calculate interest. So you have all the. Values. So, balance whatever be the balance that will be multiplied by interest rate because you are passing the

interest rate also and this will be divided by 100. So, that will be your interest then balance will be incremented by the interest balance plus interest.

So then you are printing the balance. So it will print interest calculated and added to savings account. So whatever be the balance will be printed. So this is what happening in that particular savings account method. And the last one is apply service charges.

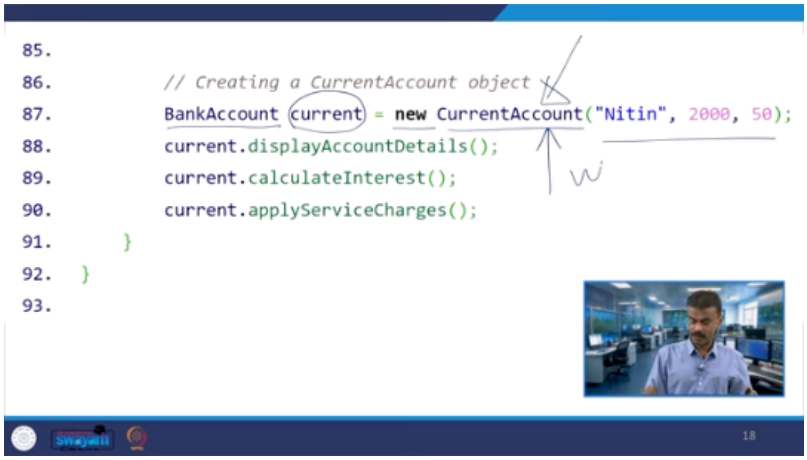
So this function, this method will be called Under savings account, right, under savings account, right, apply service charges method over here. So, this will be calculated and no service charges for saving account will be displayed. So, this will be displayed. Whatever balance, fine.

So, you are passing this and you will get the output, right. So, we will see the first output. Yes, you are getting name of the account holder, balance and then interest when you calculate based on the formulation, this will be displayed. And also we had seen no service charges for savings account. So this will be displayed.

It was in fact we had seen. And in the second case, right, we are doing one print statement, one new line print statement. So then the second one. So you have the current object, right? You have the current object whose class is bank account, right?

You have the current object whose class is bank account. So here you have dynamically you are allocating a memory with current account as the constructor, right? You are passing three arguments nithin, 2000 and 50, right? So first, that means you are accessing current account, right? So your object which is created during compile time is bank account.

But during runtime, it is current account upcasting, right? So, here you have upcasting. So, this current object which is invoking display account details, right? So, current account, what is display account detail? So, this is your current account.



So, display current account detail. So, display is in, right? Current is invoking display account details. So, you have display account details. Here you go, account holder.

and balance everything will be printed right so it will invoke the current account right and then you are all having the values and again similarly this will be invoking the superclass constructor and then you are also passing the service charge so all will be copied correct so this is what exactly happening in the case of display account details and the current is invoking calculate interest so here you can have right method will be overridden in the superclass and here In this particular case. Right. This is the first one. Savings account.

Under current account. Right. So here you go. You are having calculate interest. So in the case of current account.

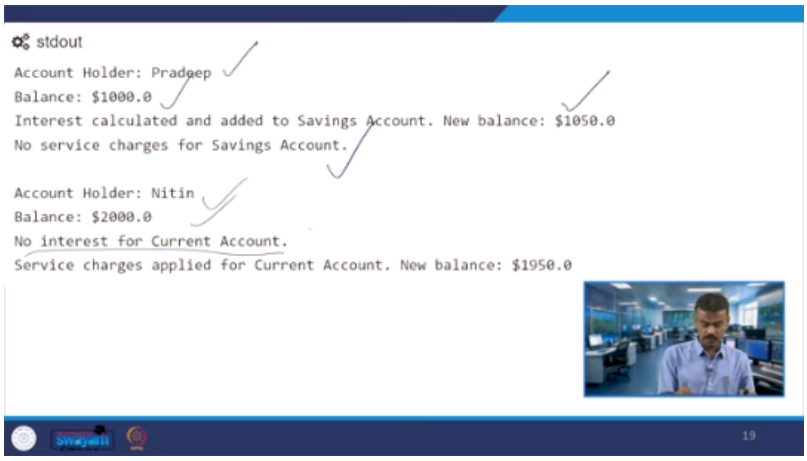
So there is no interest. That will be displayed. So this is being displayed. No interest for current account will be displayed. Right.

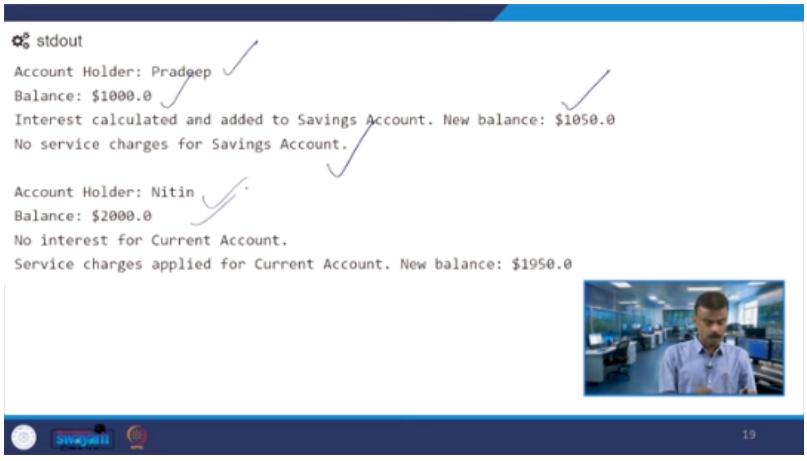
And then finally you have apply service charges. So apply service charges. In this case. Service charges applied for. Current account, right, new balance, whatever be the balance will be calculated.

So, in fact, balance will be subtracted, service charges, right, balance minus balance equal to balance minus service charges, correct. So, that will be detected, but yeah, so service charges we are passing, correct. And then whatever be the value that will be subtracted with balance and the remaining balance, right. Balance minus service charges, right? So, when you are passing this, your name will be Nitin, right?

So, name will be account order name. This is super class, right? And you are having a balance, account order name balance. And then in the respective account, savings account, you have the remaining interest rate, right? So, here you have the interest rate, correct?

So, when it is calling apply service charges, so that particular method will be called. And when I run the code, I will get the output Nitin 2000, right. No interest for current account. This also we have seen, right.





And then interest you have passed, right. How much you have passed? You have passed 50. The interest rate is 50. So, based on that, this current account function you have, right.

So, you have the balance. The balance will be calculated, right. The balance will be calculated. So, 2000 which will be subtracted with 50, right. right.

So, 2000 will be subtracted with 50. So, therefore, you have to get 1950. So, that is what we are getting as a output, right. So, this is how your method overriding works, ok. So, we had seen several examples on method overriding and in one of the examples you are getting the error because the move method is available in both base class and derived class or super class and sub class whereas the other one the bark

which is not available right in the super class therefore, it is giving an error. So, this example you keep it in mind and accordingly you have to write the code. So, with this I am concluding of today's lecture. Thank you.