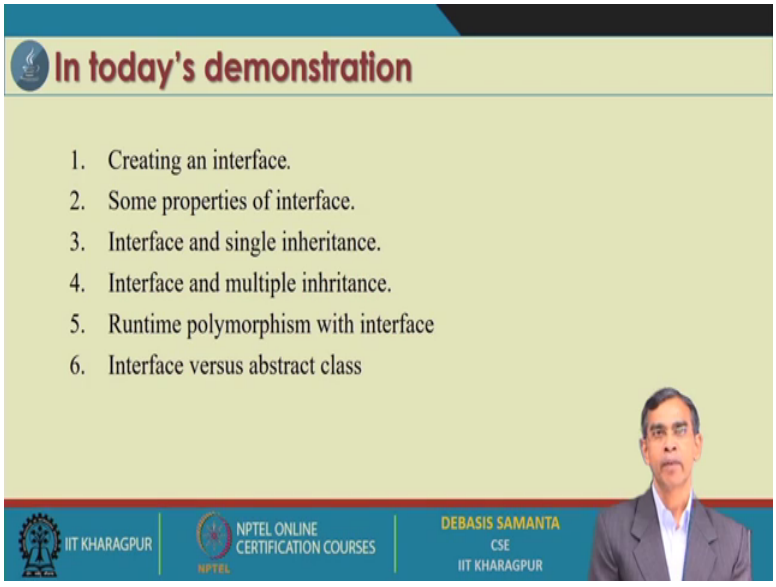
Programming in Java
Prof. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture - 22
Demonstration – IX

So, we have learned about interface in last two modules and this is a plan for this module plans for a I mean quick a demonstration of the different concepts that we have learnt in our lecture session.

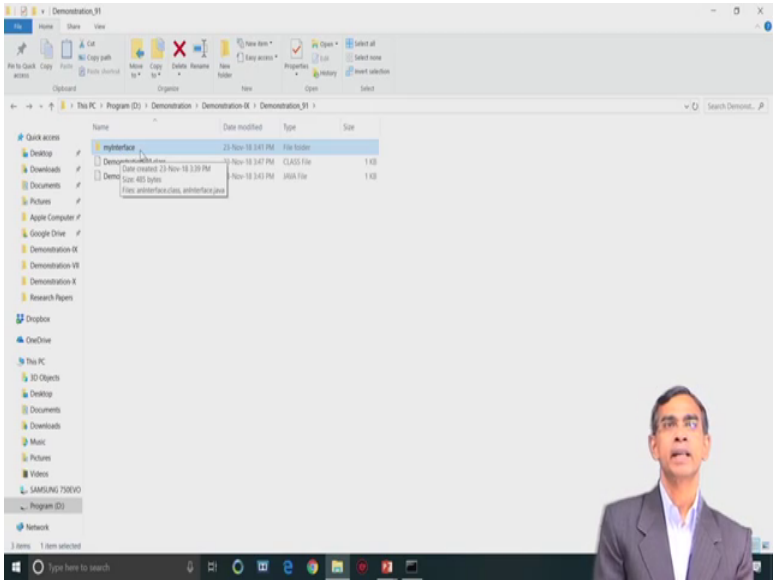
(Refer Slide Time: 00:36)



So, in today's Demonstrations mainly we have planned that how user can create an interface and then maintain it in their package. An interface as we have discussed that it follows certain specific properties so, if all those properties are not satisfied then whether the interface creation will be successful or not that we will learn it. And then interface is mainly used for inheritance purpose more specifically it is a multiple inheritance. So, single inheritance as a multiple inheritance using interface will be discussed. And another great application of the interface is to provide a dynamic binding in the form of runtime polymorphism so, that will be discussed.

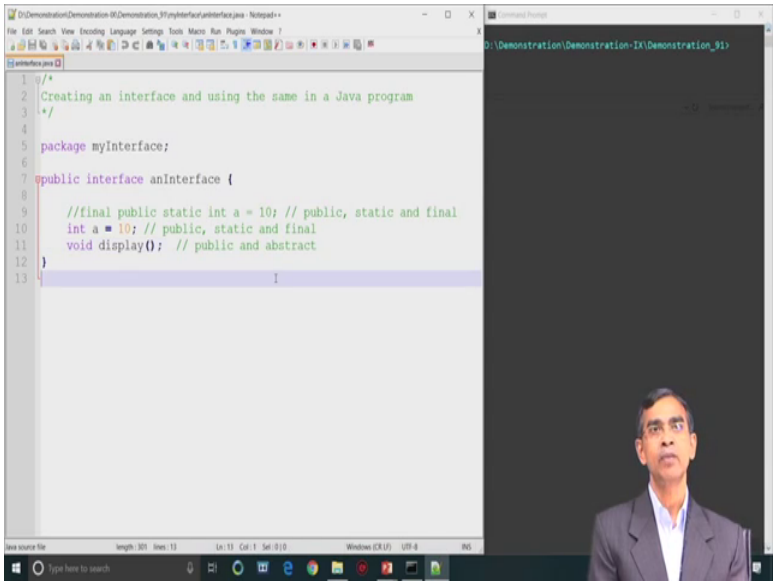
And then we have also mention that abstract class and the interface has many similarities between the two. So, what is the difference between the two so, in our last demo that we will try to clarify further.

(Refer Slide Time: 02:04)



So, let us have the demonstration fast this is regarding how we can create an interface. So, these are the first program that you can think about how an interface can be created. So, we have to first create a package where we have to store the interface. let us this is the name of the package, my interface. We have created a subdirectory name the my interface where as the interface will be stored there. So, let us go to the subdirectory my interface and as we see there is one program then you can just open the program.

(Refer Slide Time: 02:22)

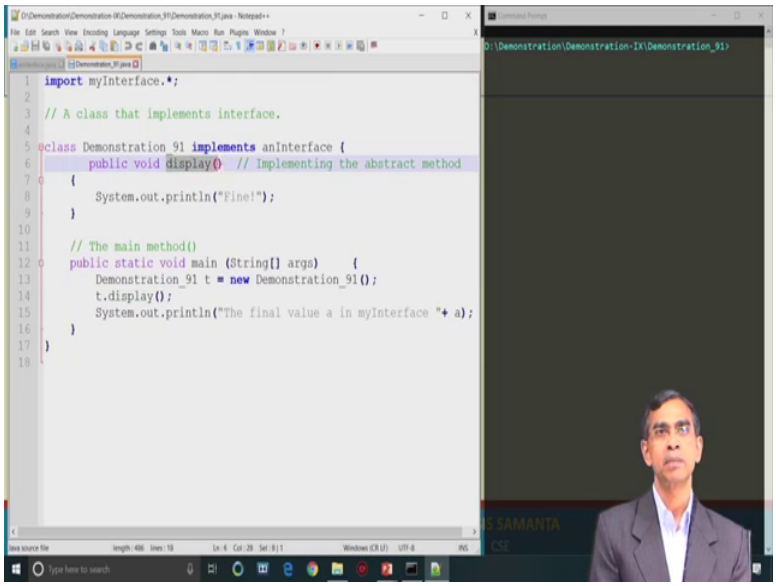


So, this a program, yes. So, this is the one program that we have developed just this program as you see the package statement includes that this my interface is the in the package named my interface and then the name of the interface as we have mentioned here with the key word interface the name of the interface is an interface. And in this interface we include two elements a as a member type integer and then the method display or we can type void.

And as we see the member elements that we have discussed here we do not have mentioned explicitly, but as you know in case of interface the member element should be public, static, as well as final anyway. So, these are the default explicitly you do not have to mention implicitly also if you do not mention anything in an interface, if it is declared they will be created as a public, final, static and the method as we have mentioned here the method should be public and abstract. So, we have we do not have to mention it explicitly again. If you do it very it is good if you do not do also the java compile time compiler will take care about it.

So, this is the interface that we have created and we stored this interface in the package my interface. Now, let us see the application of this interface in our program.

(Refer Slide Time: 03:56)



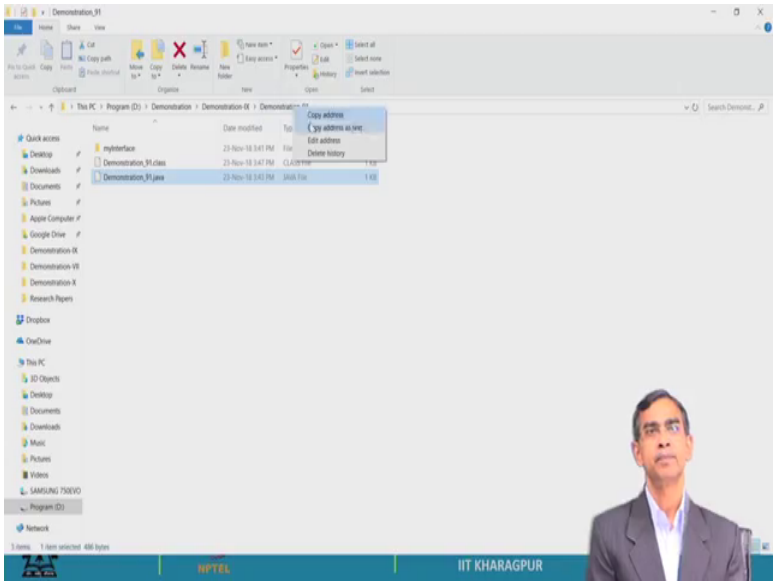
So, for this we can write one program here as we know an in interface can be used if you want to use this interface then a class should implement it is. So, this is a first the class declaration the name of the class is Demonstration underscore 91 implements the an

interface which is in the package my interface we have to input this of course. So, this is the input statement as we see in the code at the top.

Now, so this method the method in this class demonstration as the implementation of display because in interface the method display is there. So, we just implement the code the display here. So, the code includes a simple print statement, fine and as you do not have to do anything the member element so no issue it is here and in this method the main method as we see we create an object of the class Demonstration underscore 91 and then we call the method display is basically interface method it is here.

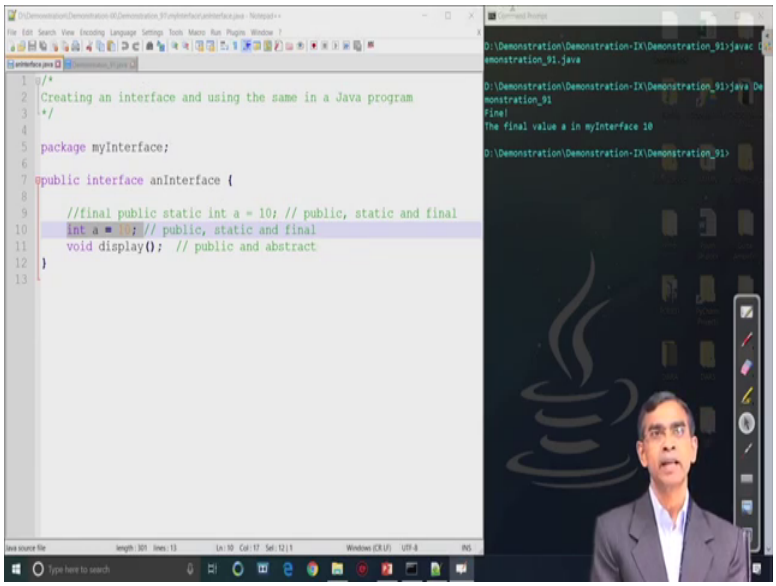
And also in the last statement you can note it is basically the print state statement in the main method, but it can access the member elements which is already there in the interface method a. So, this is the this way this program compares with use and interface and let us run this program.

(Refer Slide Time: 05:20)



Here in this case main program is Demonstration underscore 91. So, if you run it, so, if you run it we will see exactly how it works ok.

(Refer Slide Time: 05:42)

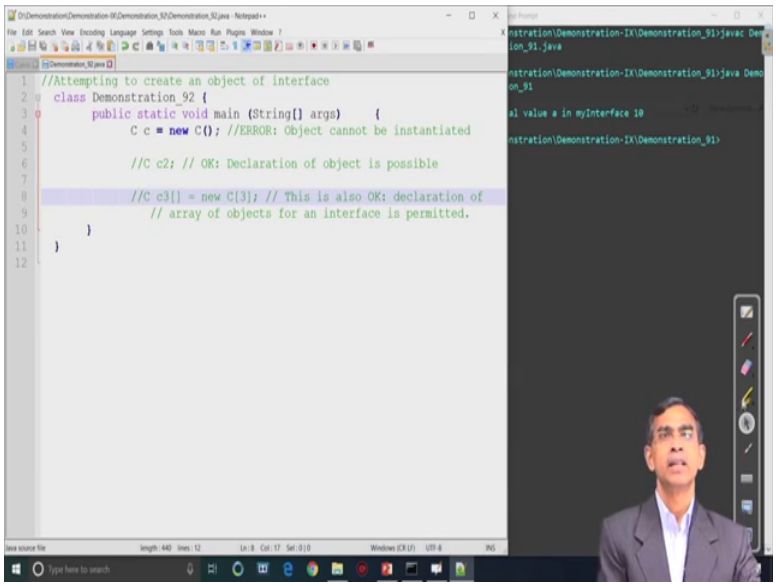


So, the program is compiled successfully. So, this program runs it, right. One thing you can see whenever you create an interface a class file also should be stored there as you see in the my interface both the interface in the form of dot Java file as well as the dot class file is also stored there. So, it is the customary as we know the process of package creation the both dot Java file as well as the class file should be there.

Now, so in the interface as you see that thus keywords public then final static void a method is not necessary. So, if you put it that will work fine no problem, but if you do not put it also it will work of it is own no problem. Now, our next demo let us have the next demonstration after knowing how an interface can be created and then how the same interface can be used in the program.

So, our next the illustration that the whether we can create any object of the interface type? Now, so, in this illustration will give you an idea about. Let us consider the an interface.

(Refer Slide Time: 07:03)

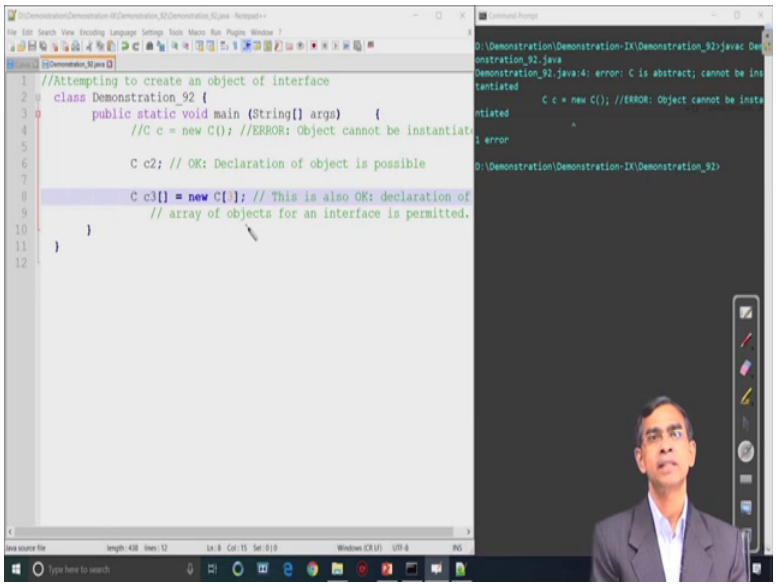


Let us consider an interface C here. So, these is the interface ok, built it here. It can be stored in any sub directory, in the same directory so, this is the interface. Now, our objective is to if we attempt to create an object of this interface so, what will happen? Now, let us see the program that we can write it a class program may be and then we want to create an object of this.

So, let us have the class program here. So, this is a class program the class file name of the file is Demonstration underscore 92 and here you see the main method. The main method is basically in the main method we try to create objects of the type interface there are many ways we can create. So, in the first statement as we see the C is the interface type and small c is an object name new C. Here we want to create an object of the interface C. As we told you as interfaces is an abstract so, no object can be created. So, definitely this will leads to an error.

And, then so, the last two statement you can make it comment ok. So, we will discuss about the last one. Now, let us see if we do it whether we can do it or not.

(Refer Slide Time: 08:42)

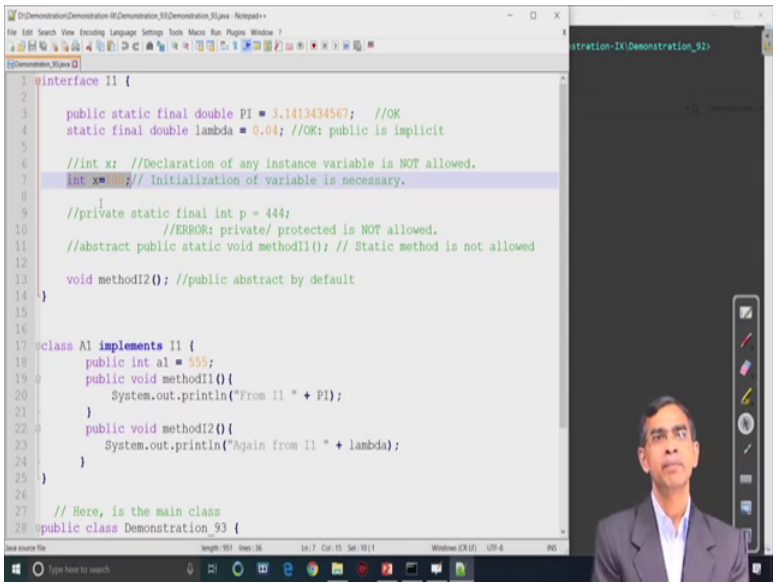


Now, as you see here is a compilation error in the statement where we have attempted to create an object of interface C so, it is not possible. So, now, let us again revise the program so that we can comment this statement and then now other two statement let us see. Now, others two uncomment it so, other two statement uncomment, now here again. So, here C is a interface type and it create an object c 2, this is quite ok.

Here basically we so, there are two difference between the first statement and the this second statement. Second statement is basically create an object and instantiation by calling the constructor, but here we just simply declare an object c 2 of C type. So, this is quite possible we are declaring only, but not creating a new object, no instantiation is involved. In our second case also we can declare an array of types C through array of interface object, but not any instantiation. So, what we can say is that instantiation of an interface type is not possible however, declaration of interface object is possible.

So, in that sense the last two statement is correct. Now, let us see if you run it. So, this program run successfully this means that we can create it. Now, we will see the usage of this kind of declaration in our next example we will discuss it, not now ok in due time.

(Refer Slide Time: 10:30)

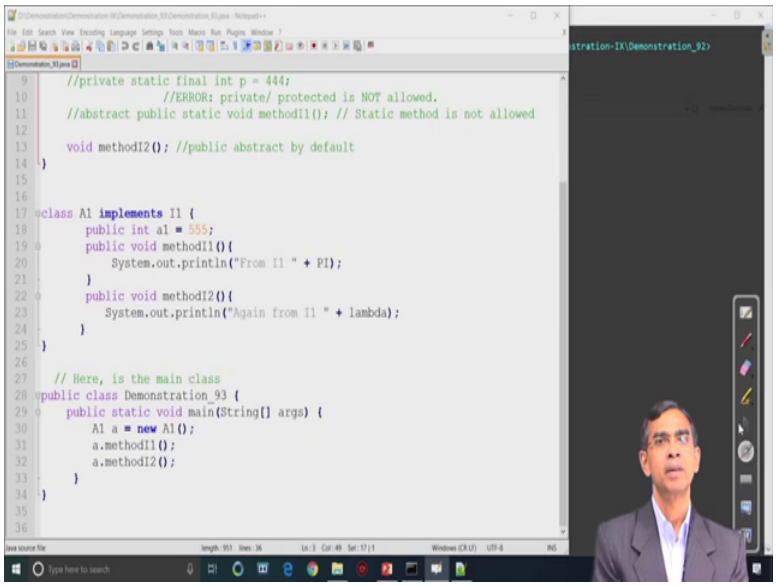


So, now we have discussed about interface; how a interface can be created and then how the same interface can be used in our class program. Now, as you have told that an interface should be implemented and in the process of implementation all the methods which are there in a interface should be implemented successfully. If you do not implement any method or if we implement one method as a private or something other type then it should give an error.

So, now, let us have the complete program here. Here in this code we see the first few declaration about Interface I1. In this interface we declare one field called PI and declared public static final that is ok. Also we declare another member elements lambda the floating type. Now, so, this is fine and then also in this method in this interface we declared on public abstract method by default namely the void method 2; method I2 here ok.

So, all these things are perfectly it will work because is the as per the specification of the interface declaration. Now, as we know in case of interface only the member that will be declared as a static no instance variable is not possible. Now, let us see if you want to declare a variable say x of type integer and it is declared as 100 and so, these basically considered as a instance variable. So, this leads to an error if we run this program. Fine, we run it later on, let us have few more discussion about it here.

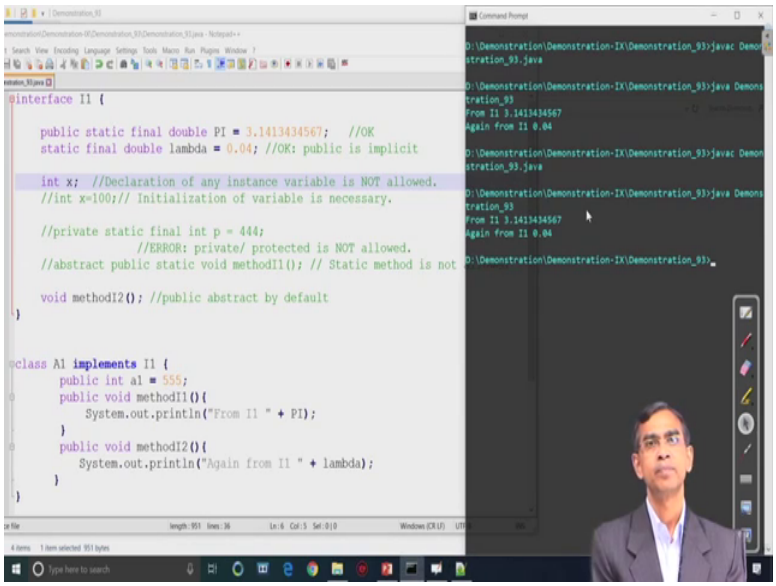
(Refer Slide Time: 12:12)



Now, here class A1; class A1 implements I1 and this is a simple implementation where we have the class A itself its own member A1 and then method 1 its own and then method 2 is an implementation of the interface method. And as you see as the class A implements I1 all the elements those are there in interface is readily accessible to this class A. For example, PI this is accessible via the method of its own method I1. And now the main class as we see in the main class here we just create an object of class A1 and then all the methods which basically implements the a A1 including the class A1 own method is invoked here.

So, this will run successfully, but before running this program that definitely we should comment the int x. comment int x should be commented here no next one, now fine. So, this program compilable, as well as executable so, let us quickly compile it and then we will come back to this program.

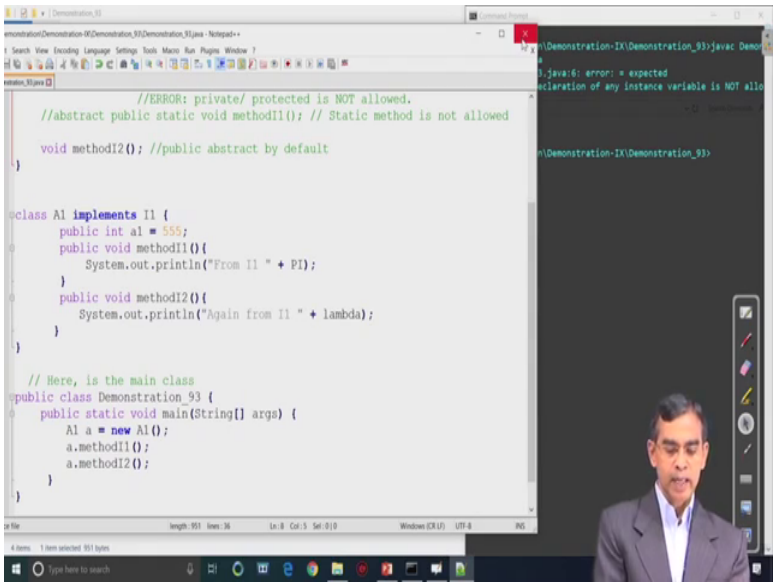
(Refer Slide Time: 13:25)



Again making the x as an instance variable and then see whether it work or not. So, compilation is successful, no error has been reported and then this is the execution of the program. So, it basically call method I1 and method I2 from the program itself. As you see method I1 can access the pi which is defined in the interface now let us come back to the program again and here we are in the process of creating the instance variable here.

So, in int right now. So, here int x equals to 100 as we know that this is basically if you do not specify the key word that is basically public what is called the public static will be default there. What about the int x declaration so, this comment keep it here no problem. this one. Now, let us see that is what this basically creates an instance variable basically run compile.

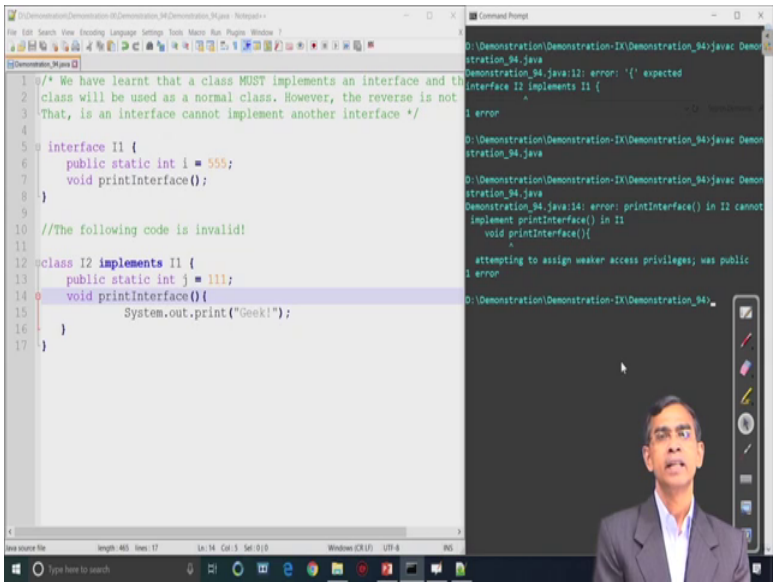
(Refer Slide Time: 14:46)



Now, here we can see int x without any initialization it indicates that it is basically as an it works as an instance variable. So, if we declare without any public static final keyword it will automatically specify this one but this is only applicable after the initialization. You should if it is a static variable you should initialize this one. As there is no initialization it is as an instance variable and no instance variable declaration is allowed in an interface method. So, this example explain that we cannot declare any instance variable this one.

Now, here so, we have learned about that a class can implement an interesting and interface. Now, it is interesting to note whether an interface can implement another interface or not.

(Refer Slide Time: 15:38)

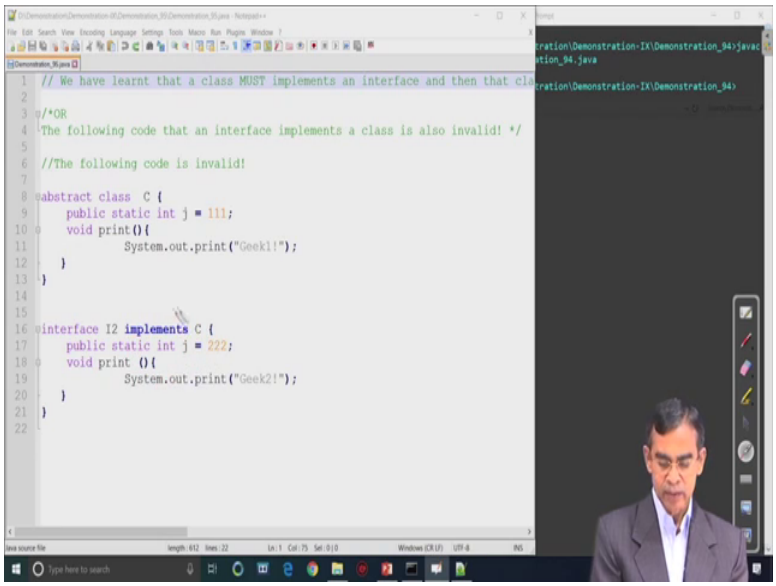


So, here is an example as we see here I1 is an interface declared with its own member I as 555 and its own public abstract method namely print interface and here is a second is there I2 interface basically attempts to implements I1. In fact, all the entire course is invalid code because I interface I2 cannot implements other interface. Now, if we run this program then it will report a compilation error.

As you see in the statement interface I2 implements I1 it basically is an error indicated here. So, instead of interface I2 if we write class I2 implements I1 it will work possibly and another thing is that in the last statement print interface if we made it we have to make all the method that needs to be implemented as a public. Suppose if we declare as a default the print interface method in class I2 yes, just write public term remove ok, fine. So, here it is default.

Now, let us see no default method implementation is allowed or method implementation with any other access specifier is not allowed. As we see here in the void interface it gives an error so, we have to make it public there then only it will be there. Now, a class can implements another interface if a class implements any methods which is there in a interface that methods should be declared as a public that we have learnt it here ok, right. So, we can run it.

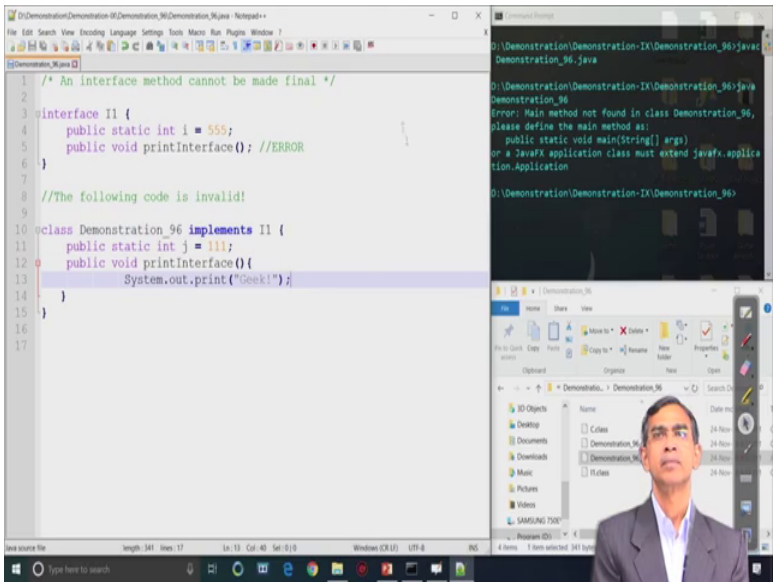
(Refer Slide Time: 17:52)



Our next example, whether an interface implements an abstract class? As we know an abstract class in many ways similar to instan interface that abstract class also can have the abstract method that mean method without any code and then it also can includes static variable. So, here you can see in this example a class C is an abstract class as we have declared and here the static variable is declared as a and it has one method print, but in case of abstract class we know it can includes both abstract method as well as non abstract method; in this case it is non abstract method.

Now, if I if we attempt to implement interface this class C that abstract class by using implements it is basically invalid. Now, if we run this then you will see this program will not compilable actually. It will give the compilation error as we see interface I2 implement C is basically it is not allowable to implement this one ok. So, no abstract class also can be implementable using any interface, but a class can be plan to implement this one.

(Refer Slide Time: 19:19)



Now, one thing is that no method or any variable can be declared final in case of interface it will leads to an error. Here is an example. Here we can see interface I1 which has one static variable public final it is as i equals to 55 by default it is final and now, the method here we can declare you can see we have declared the method as a final and static. Note, these two key words are not applicable to any method declaration in an interface only public and abstract is applicable. So, by default it is abstract so, final static is not possible. So, this code will work only if we remove final and static from this one.

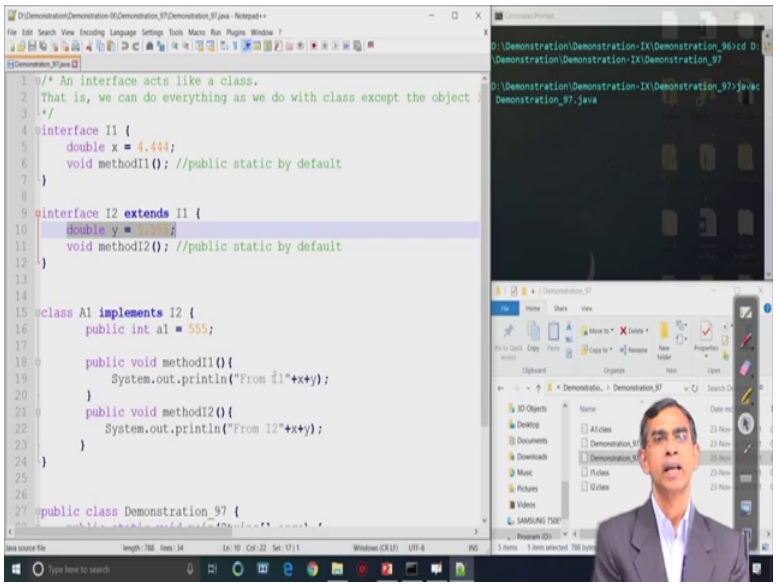
Now, let us have the final static and learn it and one by one if you execute it will see it in the next code basically class C implements I1 it basically declare its own method public and then implement the print interface here. As you see there is an error so, the error is a the modifier final not allowed here. Now, let us remove this modifier final and run it again keep it static ok. Now, run it again and let us see whether static key word is not there. Here we can see it also reports an error missing method body or declare abstract. As you have declare the abstract it is not taking this one. So, if we remove this static again it will now final, public it is basically, abstract we can write it or not write now it will work. So, void printed you have to declare public so, that is fine.

So, declare the public in the class implementation as we have not declared the public all the implementation of the interface method should be declared as a public here ok. So, fine now this program run successfully. So, there is a mistake regarding the name of the

class file. So, that is why it gives an error that it did not find the class file. Now, we have changed it, yes ok. We are compiling; yes, compilation is success full in this case running this program any way.

So, our next illustration anyway. So, the last illustration is not meant for any what is called a class there, only for the explanation that it will require the few properties to be satisfied.

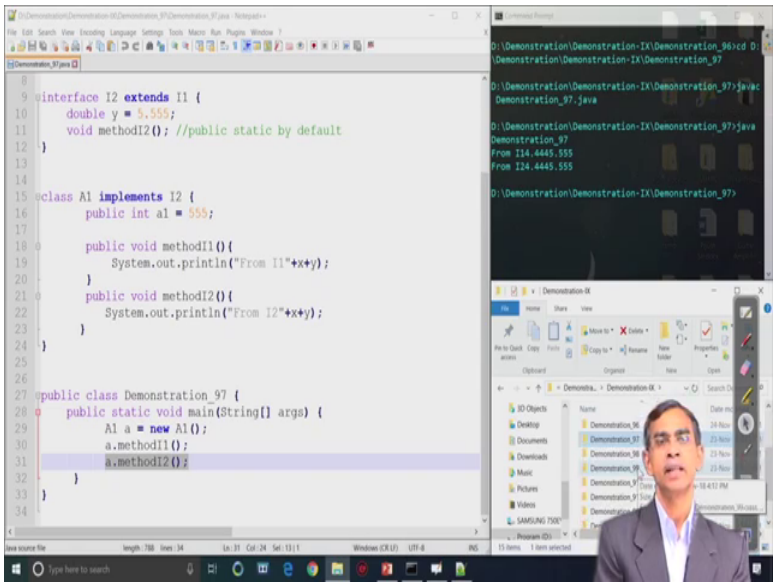
(Refer Slide Time: 22:43)



Anyway, so, our next example is basically explain the single inheritance. Now, here let us have the loop of the code interface I1 with it is public static final variable x and then it is method 1 and this interface I2 extends I1. So, an interface can be extended from I1, but cannot implements. Now so, extend means I2 inherits I1 and so, as a process of inheritance so, it has its own members y and its own method which is also abstract is declared because it is an interface method. Now, class A1 implements I2 so, whenever I2 come to this picture. So, by virtual of inheritance all the way method that is there I1 method 1 also now inherited to I2. So, here again see in this class A1 implements I2 method as we see we declare one variable in its own. This is the class A1 variable and it implements method one which is basically inherited from I1 via I2 and then also method 1 and method 2.

And, as you see here x and y are the two members who is belongs to the interface I1 and I2 and by the process of implementation class A1 has the readily accessible to these values x and y here.

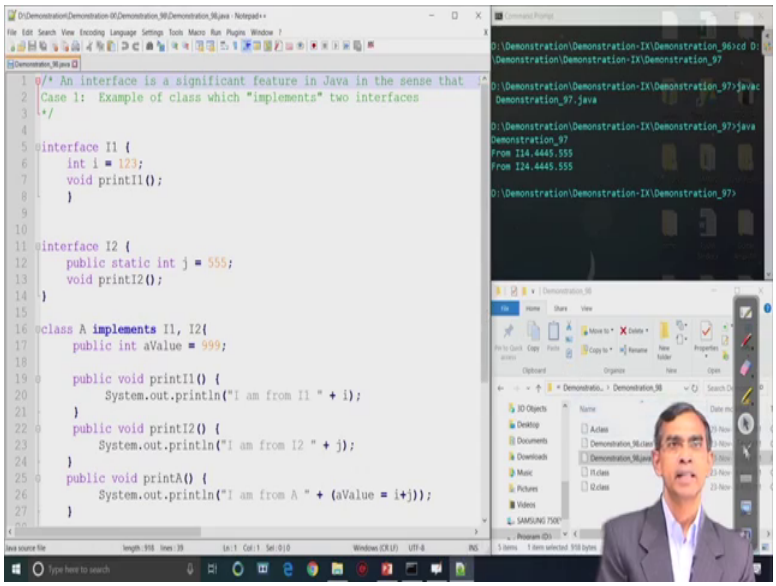
(Refer Slide Time: 24:18)



So, the main program looks like here we create an object of class A 1 and then we call the method, method 1 and method 2. So, if we call so, all the methods and then they are variables those are there in the interface will be readily accessible to this one. So, let us have a quick execution of this program followed by the compilation.

So, we can see yes, we can see that this program is now successfully executed and compilation there is no error there. Now, so, this is the idea about single inheritance; now, let us have the demonstration on multiple inheritance.

(Refer Slide Time: 24:56)

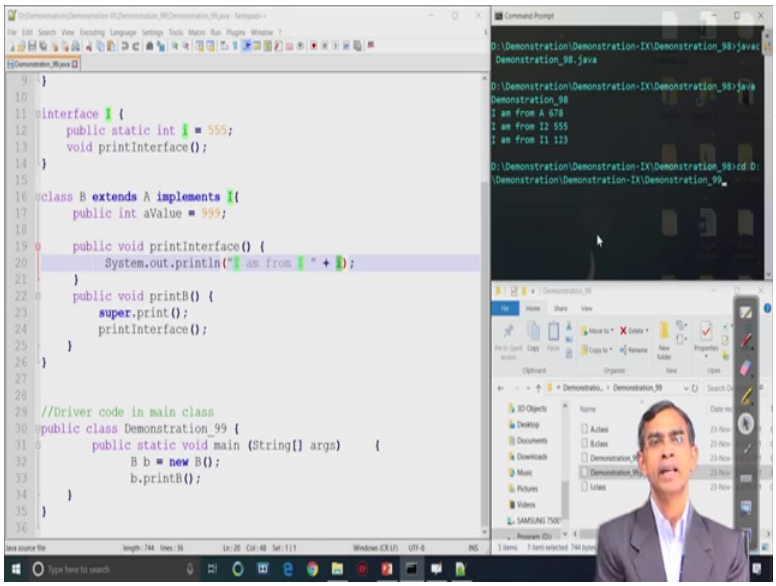


Here again I1 is an interface as we declare one of the static variable I here and print run method in interface I2 interface I2 is a again with there is again j as a static variable and print I2 methods are there. Now, multiple inheritance here as we see class A implements I1 and I2. So, this is basically the idea about how multiply a class can inherits both from I1 and I2. In the last example we see that I2 inherits I1 then class implements I2. Here basically class A can implements I1 and I2 here I2 is not necessarily inherit from I1 it basically.

So, here as the number of inheritance can be implemented by a single class; for example, class A implements I1 I2 and I 3 and so, on we can write this one. Now, in this class implementation as we see A value is the value of the class variable A and then print 1 is an implementation of class interface I1 print I2 is an implementation of interface method I2 and finally, the print A is a class A methods who is basically have the full access of the static variable those are there in interfaces i and j namely.

So, this is the program as we can see the multiple inheritance by the process of multiple inheritance we can access all methods those are there in interface as well as all variables those are there in the interfaces ok. As we see this program run successfully, now there is an alternative way of doing multiple inheritance our next demonstration is to explain this one.

(Refer Slide Time: 26:47)

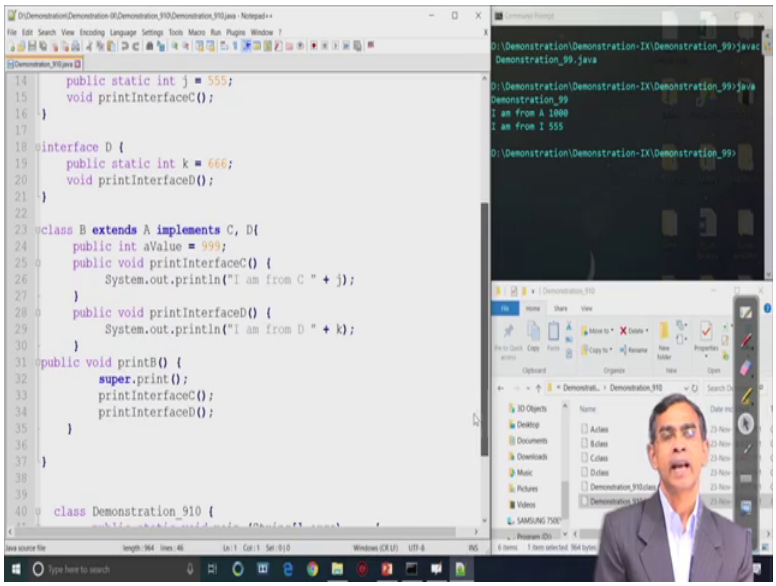


In the last example as we see class A 1 implements I1 and I2. Now, both extends and implements can be added there; this is also one way of inheritance multiple inheritance. In this example as we see class A is a class and interface I is an interface and will see is the class B extends class A; that means, it inherit the super class A as well as implements I it is also inherits basically the some variable as well as method.

However, this implementation as it is class B should implements I the method print should be defined here in this class as we have see the public void where is a print interface method. So, as we see interface I the print interface method is being implemented here that print and then call it here and class B basically ok. So, it implements the print interface method in interface and then print B is basically its own constructer who is basically call the class A constructor; that means, super class constructor using the super key word and then also call print interface methods because, this constructor has the access of because of the implementation. And the main class is very simple here. The demonstration underscore 99 is the implementation of this where we can create an object of class B and then we can call the print B the method of the class B.

So, this is the one example as we see the multiple inheritance by means of extending I mean inheriting a super class as well as implementing an interface. Now, there is another cases where it can extends on class A implements two or more interface at the same time.

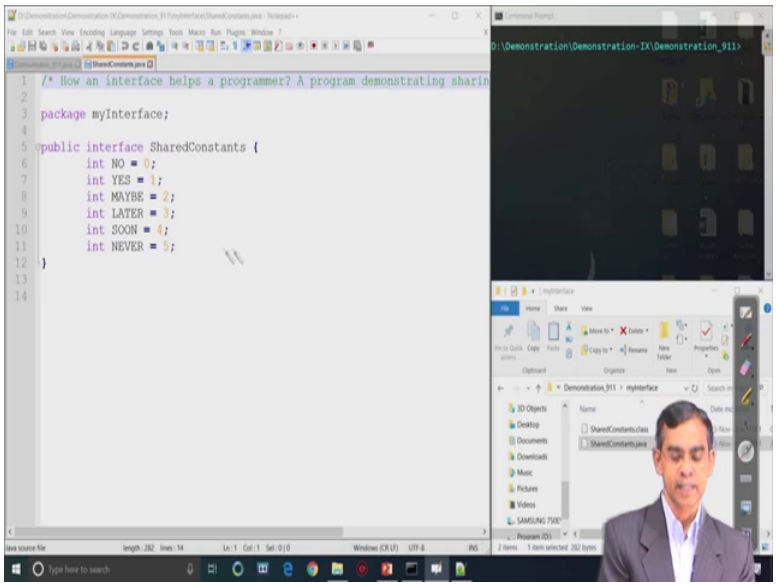
(Refer Slide Time: 28:48)



So, this example is here class A is a class interface C and D are the two interfaces and you see is a multiple inheritance form where class B extends A, that mean A is the super class, class B derived from A as well as is implement C and D that mean class B also inherits from C and D. That means, that all the static variable those are there in C and D namely j and k is also accessible as well as the variable which is protected in class is also accessible. So, we can see this method that is there or this class or main method we can say can access all the elements by means of this multiple inheritance. So, this is a one simple main method as we see here we create an object B which is a basically derived based on the multiple inheritance and then call the method print B ok.

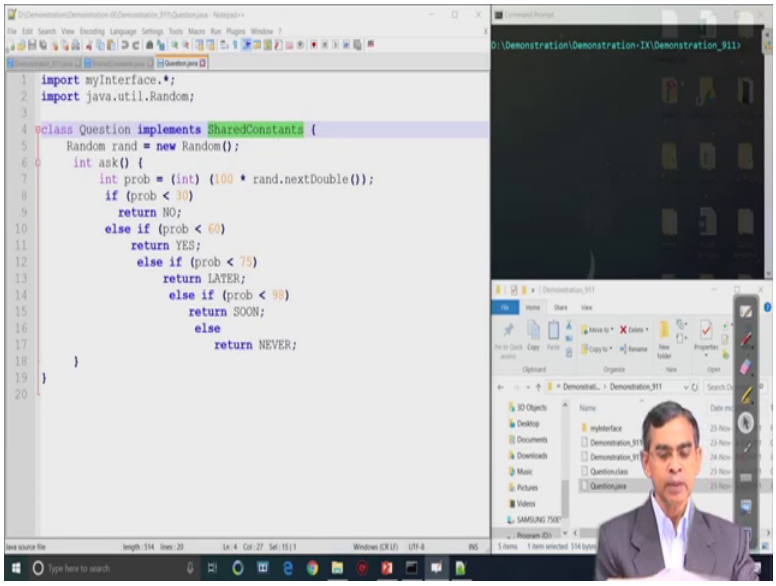
So, this is the different way that we can see that how multiple inheritance is possible with the help of interface as well as class A. Now, what is the usage of this interface we can do the multiple inheritance, this is the one application. Our next example basically explain our next example explain that the usage of the interface; in fact, interface can helps to support the shared variable to use across the many pack many packages or many classes in different packages.

(Refer Slide Time: 30:38)



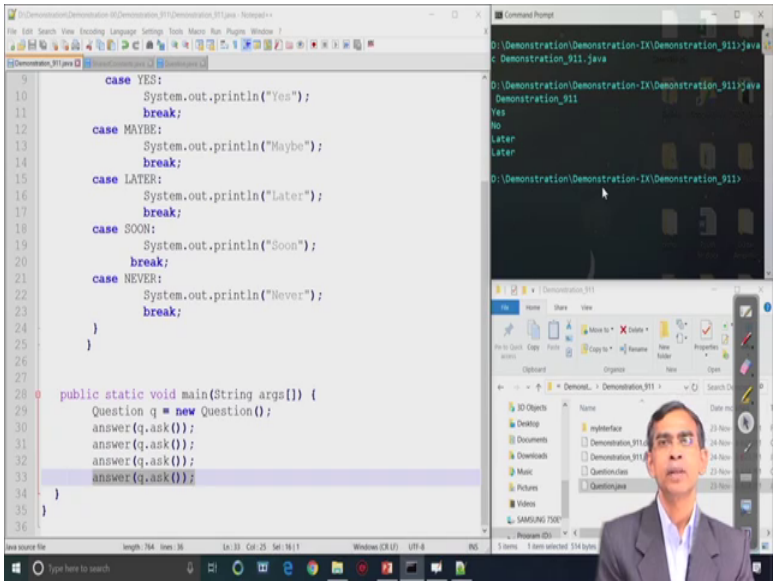
So, here is an example as you see the interface includes declarations of some variable go to the interface in package my interface go to the my interface. So, we have declared one interface here. This interface includes some static variable which is declared here; NO, YES, MAYBE this one. Now, so once this interface is declared and then stored in the package, then any other class can use them as a global variable look like so; that means, it is called as shared variable in that sense.

(Refer Slide Time: 31:10)



Now, here is an example of one class which use basically interface; that means the class question implements all this shared constant to the basic global variable we can say. And in this case is basically you see without any declaration it basically used return YES, return LATER, return SOON all these things and here is a simple code we can use the random function which is defined in java dot util it will basically generate a random number and based on the random number generation it will print all these values depending on the random probabilities.

(Refer Slide Time: 31:34)

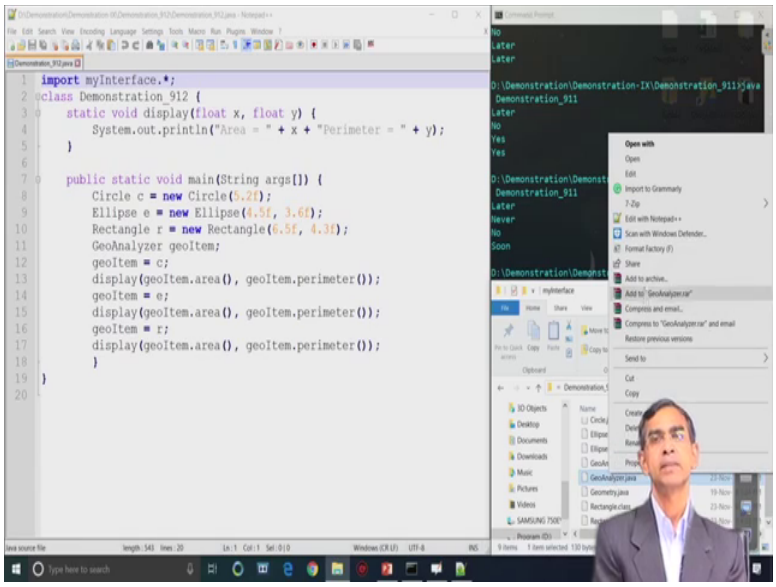


Now, here this is the another class which basically use the same things here and we define another method called answer and it basically take the results first do it and based on the result it will execute this codes are there. Now, let us have the quick method. In this class is as we see go to the next, so this method is basically create an object of question class which is declared which use the interface or implements the interface.

And then for this class we use the ask method. Ask method generate a random number based on the random number it will print either YES, No, SOON, NEVER, maybe all like this one. If we run this program let us see what is the output it will give. It will give the output as it is a probabilistic one. So, different execution will keep you not necessarily give the same output in this case ok. In this case we can see this is the output. If we run the same program again we may see different output like this one if we run again it can give another output and so on.

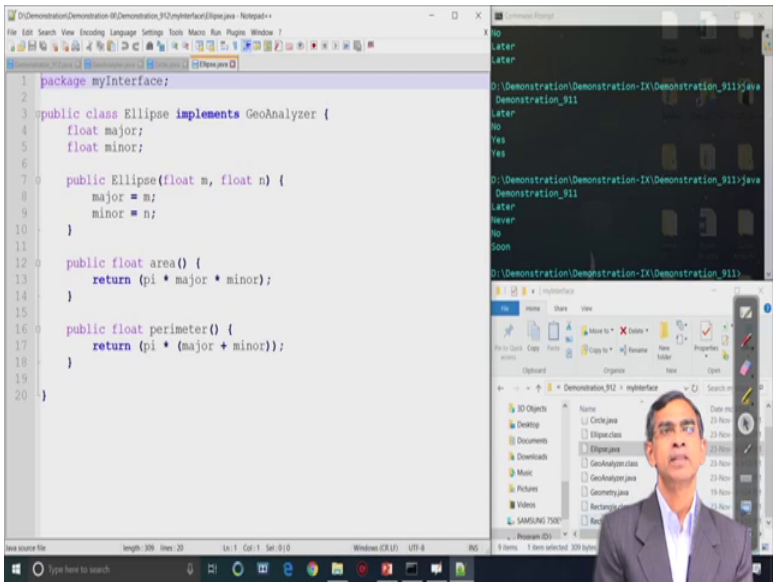
So, it is basically probabilistic program by means of generating random number, but it will discussed about how the shared variable can be used across the different classes. Now, we have to discuss in the ok, what is the difference between interface object and then abstract class interface type and then abstract class declaration.

(Refer Slide Time: 33:09)



Now, we have let us have the one program here and it is basically idea about the inheritance and by means of inheritance we can create many objects there and let us go to the main class ok. First discuss about the interface class here. Interface class go to the interface class interface go to the interface, fine no not this is the interface right ok.

(Refer Slide Time: 33:44)



So, now, this interface is defined in my package in the package myInterface. The name of the interface is GeoAnalyzer as you see this geointerface geo analyzer interface has one static variable namely pi and then two public abstract method area and perimeter. So, this method is basically the type and now we want to create few classes which implements GeoAnalyzer namely circle, ellipse and rectangle. So, this is the implementation of circle class which implements GeoAnalyzer; this is the implementation of ellipse class which is an implementation of GeoAnalyzer and GeoAnalyzer this is the implementation of rectangle class which is the implementation of the GeoAnalyzer it has its own this one and this one ok.

So, now, we see the different class implements the GeoAnalyzer interface by the different methods actually in their own way; circle has its own implementation, ellipse has its own implementation, rectangle has its own implementation. Once this implementation is done now we can come to the main program which basically create the objects of this.

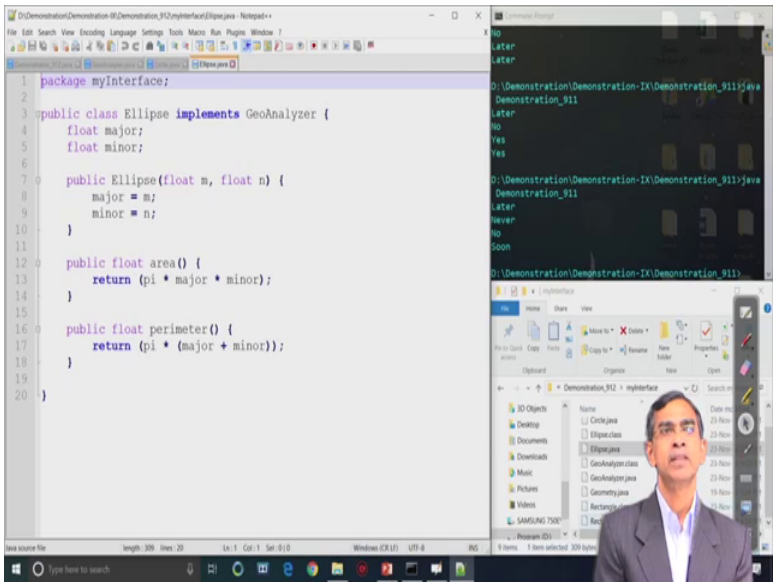
Now, here we can see the demonstration 912 is the main class and there is basically the method includes a display which basically take x and y, namely the area and perimeter of the geo objects. And here we can see we have created three objects; one object c of type class c, another object e of class Ellipse and another object r of class Rectangle.

Now, here Geoanalyzer geoItem the interface and geoItem is an object of interface we can just declare. So, we have declared and then once it is declared we see it basically holds the class c, basically the upcasting c is equal to geo item. Now, if we do this then we can call the display method which is defined in the main method by calling this geoItem and dot area and geoItem dot perimeter. Similarly, if the geoItem holds the reference to ellipse object, then also same method can print it. But, in this case this geo the display geoItem area for the referencing of ellipse object call the method area which is declared in ellipse class.

And, then again if it reference to rectangle and then display area of this one, it also reference to the methods those are there in interface rectangle in the class rectangle. Now, here we can see the display method is basically binded polymorphism. The different the display method can display area, perimeter for the different objects as per the references there. So, it is the great example of polymorphism by means of interface.

Now, this program if we run it will work for us, now again repeat the same thing, but using the abstract class. So, in the previous case we have discussed about Geoanalyzer as an interface we just want to do the same thing by means of abstract.

(Refer Slide Time: 37:08)



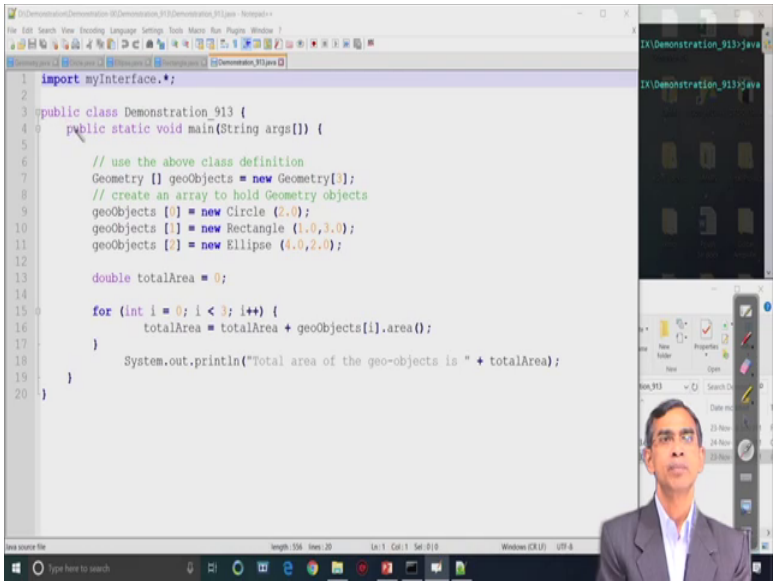
Here is basically this abstract class is declared in a package let us have the package it is there ok. So, these abstract class is declared here go to the abstract class there geo geometry. So, here we can see public abstract class geometry is an abstract class here and

here we define two methods and abstract method abstract as you know the abstract method means no code is there just look like a very similar to interface.

Now, this basically example to illustrate the similarity and dissimilarity between interface and abstract it is there. Now, let us have the second same way of class implementation it is basically extends because circle should extends geometry here we can see circle. Now, the difference we can note in the previous case circle implements Geoanalyzer, but here geometry being an abstract class we cannot implement, we just simply circle extends geometry. Likewise, Ellipse extends geometry and then Rectangle extends geometry as a process of extension is basically single inheritance as you know.

So, it basically inheritance or you can say override the method those are there in abstract method incase of geometry. So, these are the three implementation of three classes extending the geometry object and finally, the main class here demonstration underscore 9.13 so, this is the main class.

(Refer Slide Time: 38:35)



As you see the main class here it is more or less similar the previous example using interface. Geometry we create we basically declare an array of what is called the objects of geoObjects of type here three array size is three here array of abstract class method we can say just the interface objects we have created in case of interface in case of abstract class also the object can be created as we see here. And, then geoObject 0 we create an instance of the class circle and then it is basically assign to the location 0 and similarly a

rectangle instance and ellipse instance are created and we call the we pass the parameter to I mean instance share them properly.

So, these are the instantiation and instantiation, but after the instantiation they are basically referred in the abstract class object like geoObject in this case and here again runtime polymorphism as we see in the next for loop. So, here total area geoObjects i dot area. Now, for i equals to 0 so, this will this dot area refer to circle objects on the other hand for i equals to 1, this area refer to the Rectangle objects and then for i equals to 2 this area refers to Ellipse object. So, again it is an example of runtime polymorphism. Because it is polymorphycally resolved that different method depending on the objects it is there although it looks like same for all calls actually and if you run it the similar to the interface it will run and then give the execution ok.

So, this is the right. So, this is the successful execution of the program. So, what we have learnt about here that more or less interface and abstract behaves in the same manner, then why java developer maintains both the thing? The difference only here is that abstract class if you declare it cannot be multiple inheritance or whenever an inheritance interface is there it can be multiple inheritance. So, this is the only difference between the abstract class and the interface otherwise both the concepts are more or less same.

So, we have learned about the interface and if you have further any doubt you are most welcome to post your queries and all the programs those we have used in their demonstrations you are free to use it and then test it so that you can practice it much more.

So, thank you very much.