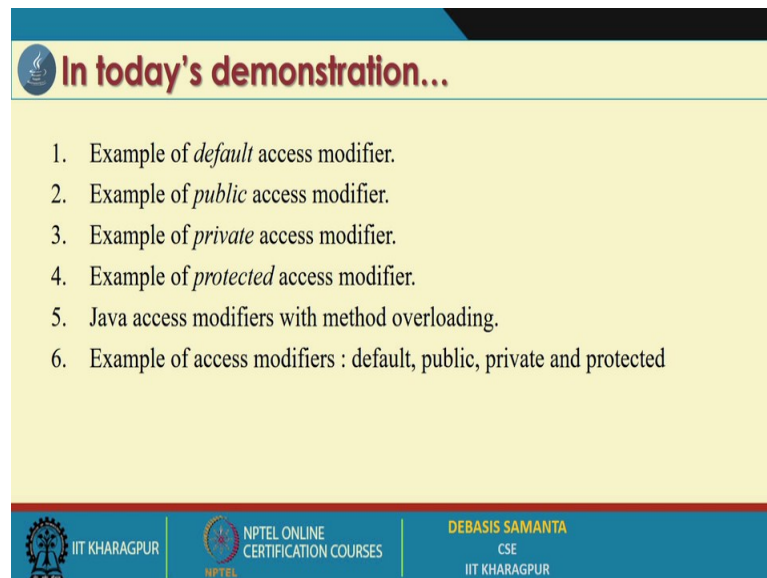


**Programming in Java**  
**Prof. Debasis Samanta**  
**Department of Computer Science Engineering**  
**Indian Institute of Technology, Kharagpur**

**Lecture – 16**  
**Demonstration - VII**

So, information hiding is a very important and object-oriented paradigm and information hiding is also very nicely has been featured in Java program. So, we have discussed the information hiding in our last module. Now, it is our time to have a quick demo of information hiding concepts in Java.

(Refer to Slide Time: 00:43)



**In today's demonstration...**

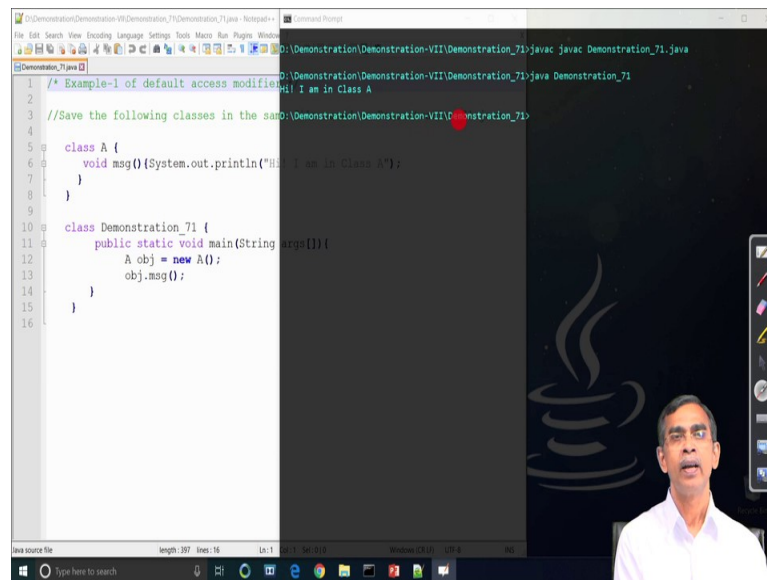
1. Example of *default* access modifier.
2. Example of *public* access modifier.
3. Example of *private* access modifier.
4. Example of *protected* access modifier.
5. Java access modifiers with method overloading.
6. Example of access modifiers : default, public, private and protected

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA  
CSE  
IIT KHARAGPUR

As we have already discussed at that information hiding in Java is by means of four access specifier; default, public, private and protected. In this demonstration, we will demonstrate how the different access modifier will have different access protection for the different classes in Java program.

And then with this modifier the method overriding also method overloading as well as overriding how the two things can be there and finally, we will have a quick demo about having all the modifiers in one program like. So, this is our plan for the demonstration today. Now, let us have the first demonstration in this series where we were going to discuss usage of default access modifier.

(Refer Slide Time: 01:33)

A screenshot of a Java IDE window titled 'Demonstration\_71.java'. The code defines two classes in the same file. The first class is 'A' with a 'msg()' method that prints 'I am in Class A'. The second class is 'Demonstration\_71' with a 'main()' method that creates an instance of 'A' and calls its 'msg()' method. The IDE interface includes a menu bar, toolbar, and a status bar at the bottom showing file length and line count.

```
1  /* Example-1 of default access modifier */
2
3  //Save the following classes in the same file
4
5  class A {
6      void msg(){System.out.println("Hi I am in Class A");}
7  }
8
9
10 class Demonstration_71 {
11     public static void main(String args[]){
12         A obj = new A();
13         obj.msg();
14     }
15 }
16
```

Let us have the program here so, class A is declared as a default class because, if we do not specify any modifier before a class name then it will be treated as a default. So, class A, in this case, is the default and then you same thing the message void is basically is also a default method because no access specifier is mentioned.

So, as a whole, the class A is a default class and with a default method the message. Now next is that main class which also defaults class here and it is basically in one file because both the class A as well as demonstration\_71 is stored in the same file. So, there is no issue of running it now. So, if we run this program so, it will basically run successfully, but it will run with default access specifier for this program it will run let us run this.

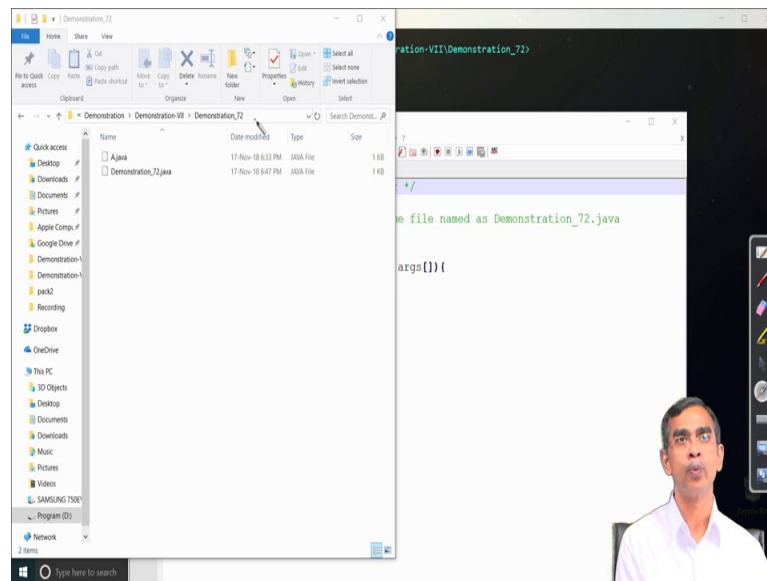
So, as it is compiled successfully definitely execution will be there and it will give the execution. So, it is successfully executed now. So, the program is not here actually, but the program will be seen somewhere else now let us see. So, here basically in one file the in this case we use the name of the program file as demonstration\_71.java. In one file we store two classes, but for good practice that a Java programmer should maintain a separate file for the different classes.

So, here is an idea about that class A is store not in the same file as in the main class, but store in another separate file and name of the file is same as the class file. So, class A stored in the file call A.java and demonstration\_72 is a new demonstration showing that

two files they are in two different two classes are in two different files if it is there whether they are if it is here in the class a file if you go to that you can see that this class A is declared here with default access specification.

Now, let us switch to the demonstration\_72 program file it basically includes the main file here it is also the default. Now, here we can see these two files are stored where we assume that these two files are stored in the same directory here\_demonstration 72.

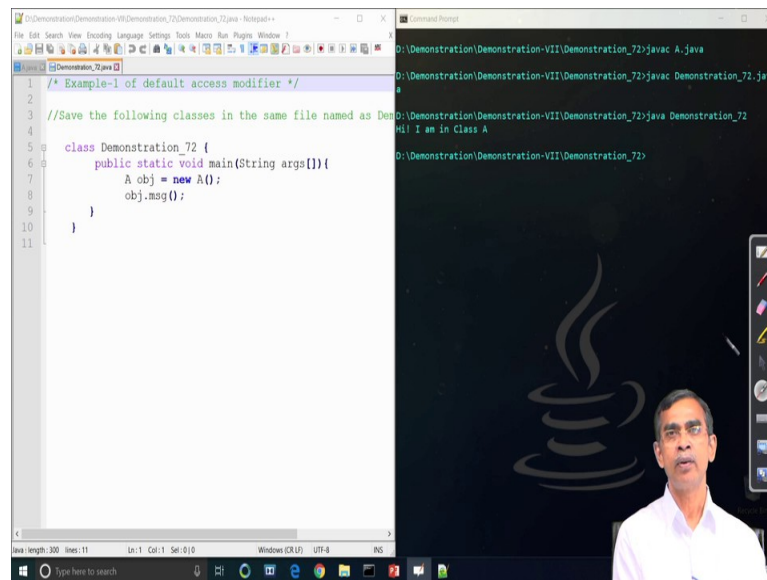
(Refer to Slide Time: 04:07)



So, if those two files are stored in the same directory. So, the resolution is not a problem and if the files are with access specifier so, member or accessing method is also not a problem. So, here from the main class, we are trying to create an object A as A is the default class. So, object creation is not an issue and then we are accessing the method m s g the default method in the class A also not an issue.

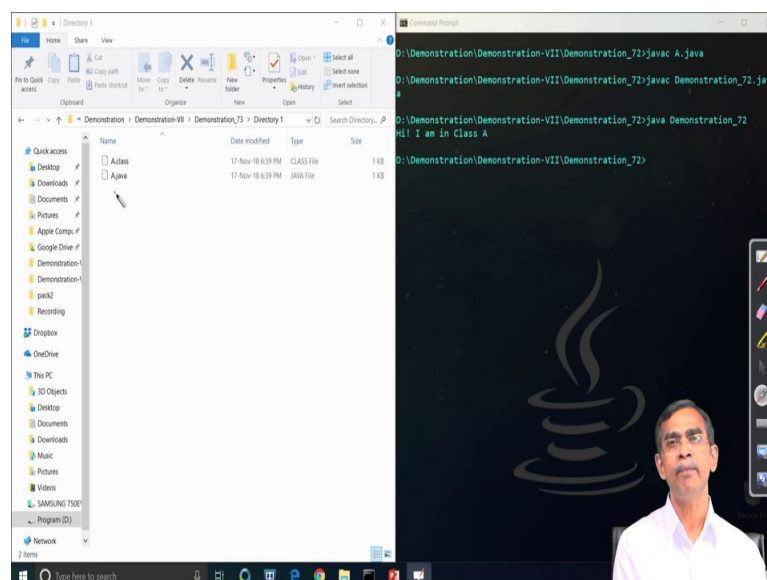
So, in this case, if we run the program it will run successfully this means that access specification is access specification allowed to access within the same file or in the same directory. So, no issue if they are in the same file or same directory access modification will work for you. Now, yeah so, this is working.

(Refer Slide Time: 05:01)



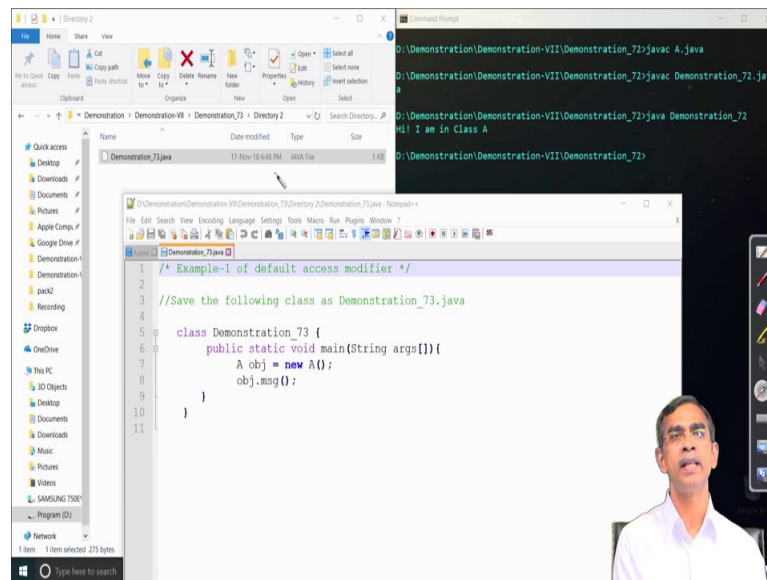
Now, in our next demonstration we will see if the class file A moved to some on the directory and then if we can run this program if which is from another directory, but with access, specifier is a default. Now, here if you see we created one directory `demonstration_73` under we can create a directory 1 where we have stored the file `A.class` and `A.java`.

(Refer Slide Time: 05:33)



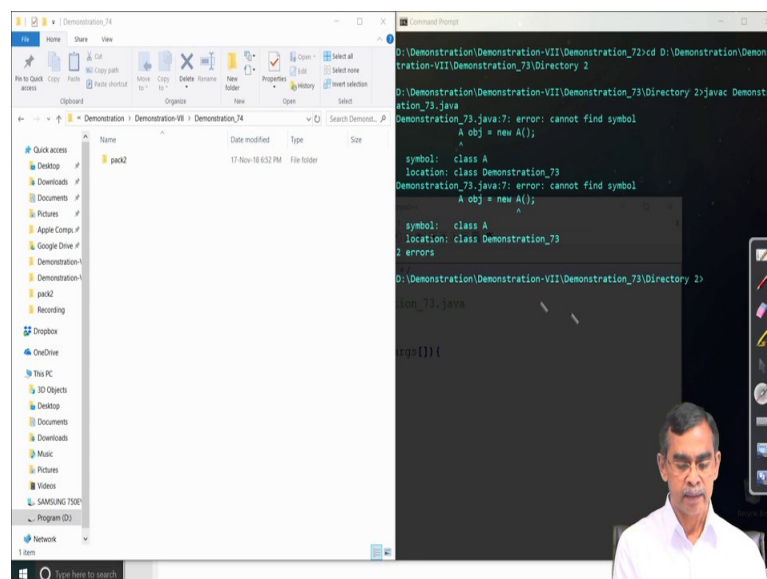
The file is there `A.java` let us see the `A.java` file here.

(Refer Slide Time: 05:41)



So, here the same file, but main stored in the directory the different directory. Now, let us go to the main class which is again stored in the different directory here, go to the main method yeah. In this case, the main method is stored in the directory demonstration\_73. Now you see the class A.java file is stored in a different directory whereas the main method is stored in a different directory. Now we are trying to execute run this program before going let us compile this now see whether it can compile successfully or not.

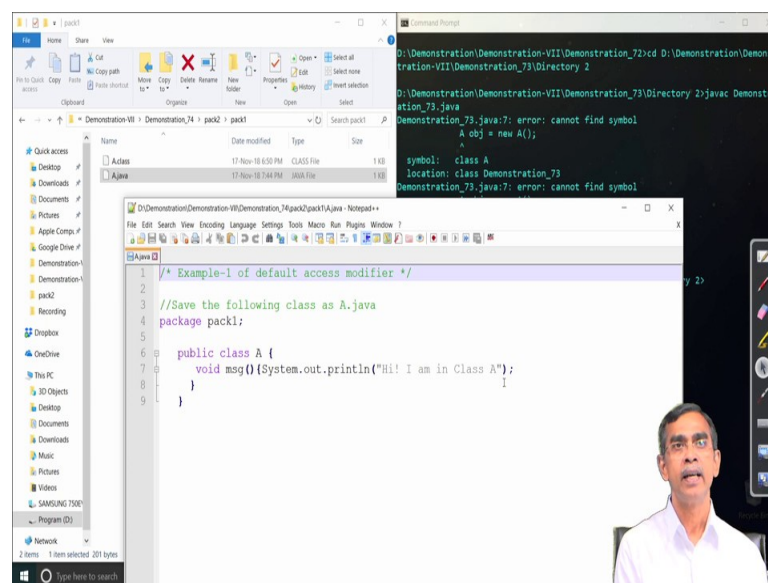
(Refer Slide Time: 06:19)



So, if it is an access specifier is the default and the two files are in different subdirectories then it will not successfully be compiled. Here you can see the compilation error is that that is basically error cannot find symbol A; that means, A is not known to this file. So, access specification if it is default then that is not accessible to any other file outside of this directory ok. So, this is why this error is giving there.

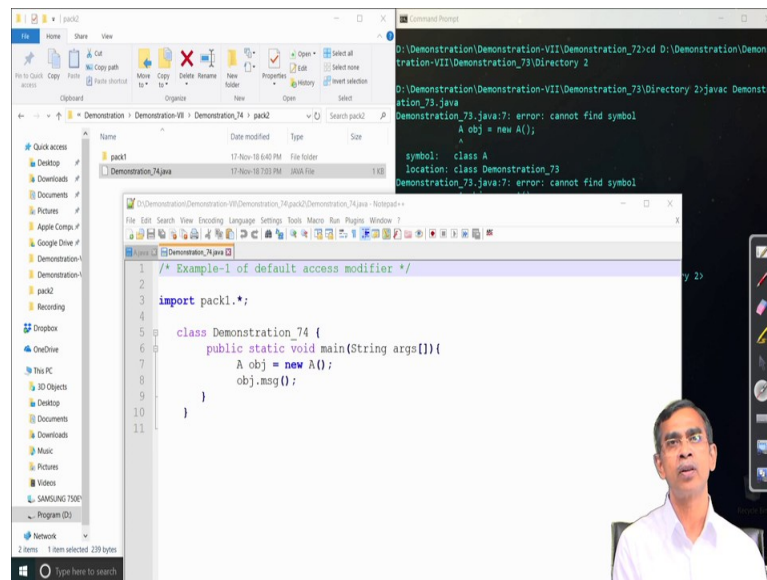
Now, so, this program shows that the last three programs that we have discussed shows how the access specification default works for us? Now, again we have some other demo here let us show the 7.4 programs here where you can see ok. So, this program basically creates a class A file in one directory as you have a directory is also can be termed as a package. So, we create one directory called pack 1 and under this pack 1 we store on file A.java let us see the A.java structure here fine.

(Refer Slide Time: 07:33)



A.java is very simple here we just mention that this A.java is in package pack 1 actually it is a directory this one. So, class and one thing you can see here we made it public. So, here class A is not accessing default access specifier by default, but it is made as public. Now in the last example that we have shown if it is default then it is not accessible to that, but if it is public then this class can be accessible outside to any file whichever it is whether the same directory or in a different directory. Now, let us have another program say demonstration program the main class which is stored in pack 2; pack 2 is a another directory and we will define here.

(Refer Slide Time: 08:21)

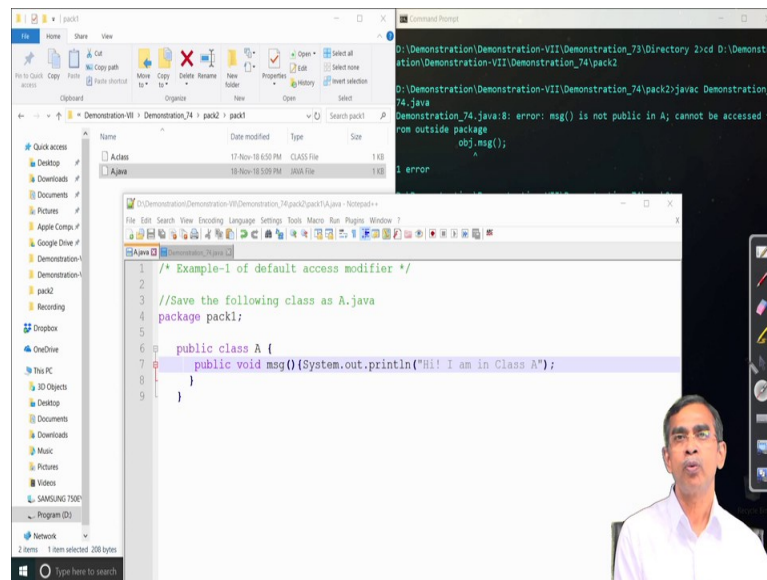


So, in under this pack 2 we store this program that name as this name is demonstration\_74.java file. And this is basically we have to give an input pack 1 showing that it has the input capability means, whatever the classes which declare there in pack 1 directive will be accessible to here because of this regarding this thing we learn about whenever you cover the package concept.

Now, here with this if we run this program earlier similar kind of program was not successfully compiled, but here you can see this program is all right it will run for you correctly.



(Refer Slide Time: 09:03)

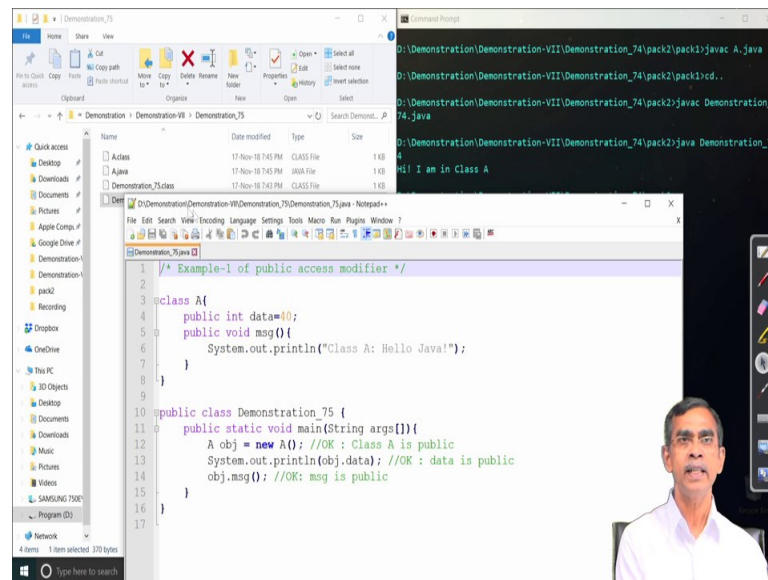


So, what is an error it is here? The message is not public ok. So, it is not working let's return to the class 1 yeah we cannot access because it is a message is default here. So, if we make it public yeah just make it public then this error is giving there because message although class A is public, but its method is not public. So, this is public is weaker restriction than the default 1. So, the default is supervoid here. So, default has the highest privilege this one so, that is why this message is not accessible.

Now, let us again go to the main method it's a compile it so, right yes. So, we have to compile because you have to changed it and the next go to the main method main class right run this and now it is successful and it is running fine.



(Refer Slide Time: 09:47)

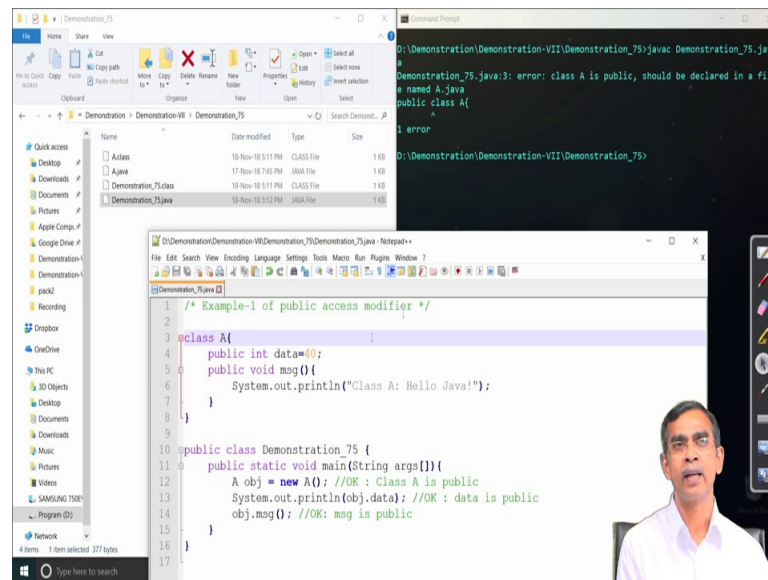


So, this now works for you. So, a public method and with the public the public class with its public members are highly accessible from outside any class that is the example explain us. Now, our next example showing on the public access specifier let us have a quick look another program let us go to the 7.5.java here yes. So, hah just load it yeah fine yeah. So, in this program we can see so, we have declared the two classes class A; where class A is declared as a default and it has members, 2 members, though both are declared as a public.

So, it public means as we know that this class is accessible to any other class belongs to the same directory; however, this class is not accessible outside to any file belongs to other directories that are fine so, but here this class a as it is in the same file of this demonstration\_75. So, no issue we can use it and here we can see we just create an object of this class here and so, object creation is and we also access the method here.

So, a public method is accessible anywhere, in the same file, in the different file, but in the same directory or in the different directory in the different file no issue. So, in this case, it is no issue. So, if we run this program it will run it correctly. Now, let us compile first this program and after the compilation will execute it.

(Refer Slide Time: 11:51)

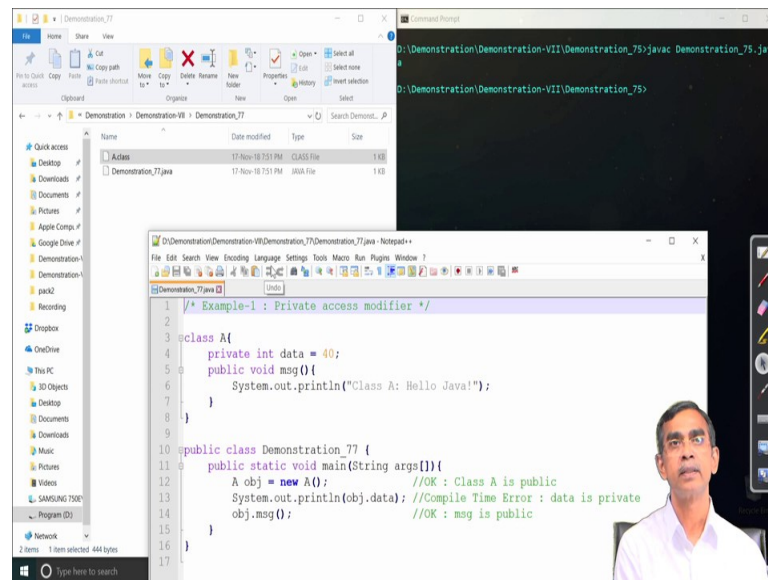


So, that it will a run it yeah so, it is running. So, fine it is running successfully yeah. So, it is running now let us switch to the program again I just want to do little bit modification. So, that we can see it a twist is here. So, A class which is in the same file whether we can declare with some other access modification, for example, the public let us specify the access for class A as public ok.

Now, let us see what will happen if we try to run a compile this program what will happen, we made it public and as you know the public is public. So, it can be accessed anywhere, but what will be the problem you can see yeah. So, here you can see here class A is public and should be declared in a file named A.java; that means, if you declare a class public then that class should be stored separately in a different file name. But, if you want to store in the same file name do not specify any other access specifier other than the default. So, it is not required here ok.

So, if you use two or more classes in the same file you should have the access specifier as a default no other access specifier is allowed there, but in the separate file then you can specify any access specifier. So, this is the example that we have discussed regarding the access specifier and then public specifier and let us have another example about the. So, private access modification let us go to the program 7.7 here we use the private access modification.

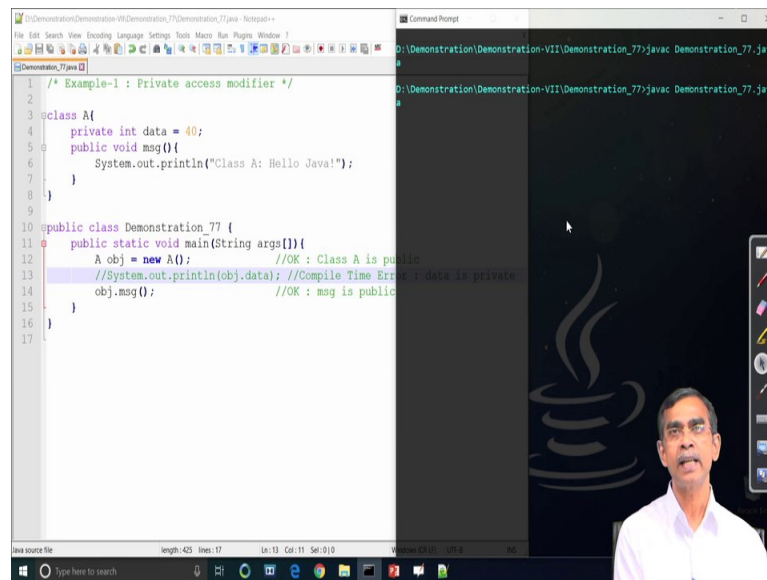
(Refer Slide Time: 13:49)



Now, let us have the program here yeah we can see one class yes ok. So, class A here is declared and it has 2 members; one method one method and 1 member the data is a 40 as a declared as a private and then the message is declared as a public. So, 2 members data and method one is private another is public. So, and then class A is a default with default access specifier.

Now, in the main method main class demonstration\_77 is the main class name here we create an object A. So, it is because a is accessible to its own file and then system.out.println() object.data. Now, here the comment is that compile-time error this is obvious because the data which is declared in class A is a private and then private data is not accessible outside this class A it is accessible to the class itself, but not outside. As the demonstration\_7 sees an outside class of class A so, we cannot access so; this is why it will give an error.

(Refer Slide Time: 15:17)

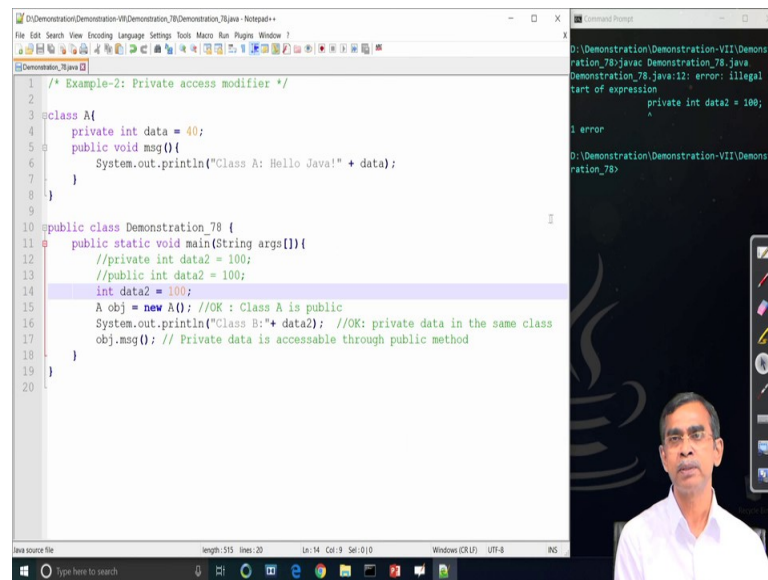


```
1  /* Example-1 : Private access modifier */
2
3  class A{
4      private int data = 40;
5      public void msg(){
6          System.out.println("Class A: Hello Java!");
7      }
8  }
9
10 public class Demonstration_77 {
11     public static void main(String args[]){
12         A obj = new A(); //OK : Class A is public
13         //System.out.println(obj.data); //Compile Time Error : data is private
14         obj.msg(); //OK : msg is public
15     }
16 }
17
```

Now, let us run this program and we see it is giving an error yeah. Now, here you can see error data has private access in A. So, as it private access so, we cannot access it this one if we comment it then definitely it will work go to the program and comment it yeah, comment it yeah fine no this one no system.out.println().

So, its comment yeah then it is work it will work because we are not accessing any private members to any other class fine. So, this actually shows that how private members access specifier is work there, let us have one more example of private access specification 7.8 please go to the 7.8. Now, this is an interesting program a program on you just note it.

(Refer Slide Time: 16:11)



```
1  /* Example-2: Private access modifier */
2
3  class A{
4      private int data = 40;
5      public void msg(){
6          System.out.println("Class A: Hello Java!" + data);
7      }
8  }
9
10 public class Demonstration_78 {
11     public static void main(String args[]){
12         //private int data2 = 100;
13         //public int data2 = 100;
14         int data2 = 100;
15         A obj = new A(); //OK : Class A is public
16         System.out.println("Class B: "+ data2); //OK: private data in the same class
17         obj.msg(); // Private data is accessible through public method
18     }
19 }
20
```

Terminal Output:

```
D:\Demonstration\Demonstration-VII\Demonstration_78>javac Demonstration_78.java
Demonstration_78.java:12: error: illegal start of expression
    //private int data2 = 100;
    ^
1 error
D:\Demonstration\Demonstration-VII\Demonstration_78>
```

It is the same problem as in the last program one, but it is a little bit different is there the same problem in the sense that class A has the same specification private and public a message method. Now, let us come to that main class demonstration\_71. .Now, here this is the private int data 2 1 private member is declared here we can declare a private member here no issue that is fine. And also we declare 1 public member data 2 whatever it is there now fine to let us see let us first declare about private int data two uncomment the private one uncomment this one ok.

Now, what we have done here is that in this class main class we declared one main method is there and then one method on data is that which is declared the private and we int data 2 ok, you know private is a comment there no that is not correct. So, let us a comment it private fine now. So, data two is declared as a with access specifier default. So, default and an object A is created here. So, object creation is not a problem because the default class is used here.

Now, system.out.println() class B data no issue because is public to this method and obj message is also not an issue because is a public method here. So, here, but here you can see if we run it then see that message m s g is basically accessed a private data no issue, but it is basically accessing a private data indirectly in the main method which is in class other than the class A.

So, this is quite now this is the one important point that you can note it there let us switch to the program again I just repeat it here again here you see the message being public is accessible to any other method. So, we can have the obj.message in the main method. As the message is public it accessed, but the message this message even accesses the private data which is private to class A, but here the as we can object message we can see the result. So, that 40 is now accessible to a method which is outside the class A actually ok. So, this is the idea about.

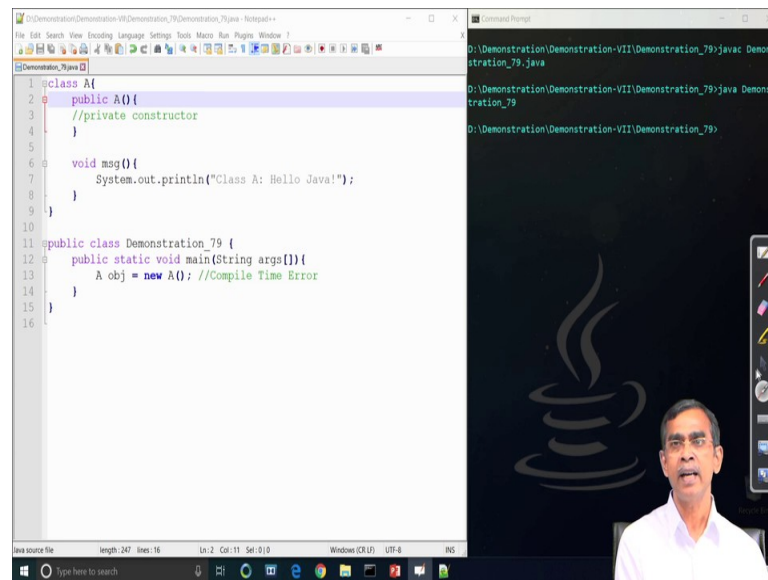
Now, here again, let us switch to the program here. Now can we declare a private method in this main method here now let us see integer data 2 may comment it please comment it and we declare public here. So, fine now let us declare here as public and let us see whether we can run this program or not yeah fine .Now, here it is an error you see why the error it is there. So, the error is that public int data 2 things cannot be declared as public only we can declare as the access specifier default because in the same file we cannot declare a method any member as public. So, public int, in this case, is giving error as it gives the public as an error a private should give an error also.

So; that means, we cannot declare one member in the main method or is the main class without access specifier any other specifier. So, let us see again private. So, private data now what is the lesson that we have learned from th

em in the demonstration is that in the main class if we declare any member this should be declared with default access specifier. No others access specifier is allowed to declare any member in the main method that is the important thing that you should note it ok.

So, this is the concept about public access specifier and in some sense the private access specification. Now our next demonstration is basically a constructor whether a constructor can be declared as a private or some other access specification other than public.

(Refer Slide Time: 20:37)



So, our next demonstration 7.9 includes this clarification here we can see one right yes 7.9 here we can declare a class as a default class, but its constructor is declared as a private. Now, we if we declare a constructor as a private what will happen that it will not be able to create any objects in any other class although it is accessible to some other class in the same program may be. So, here in the demonstration\_79 here you can see we are trying to create an object A whose constructor is private it is giving compile time error because the constructor is private. If it is a private constructor no object can give can be created in any other class.

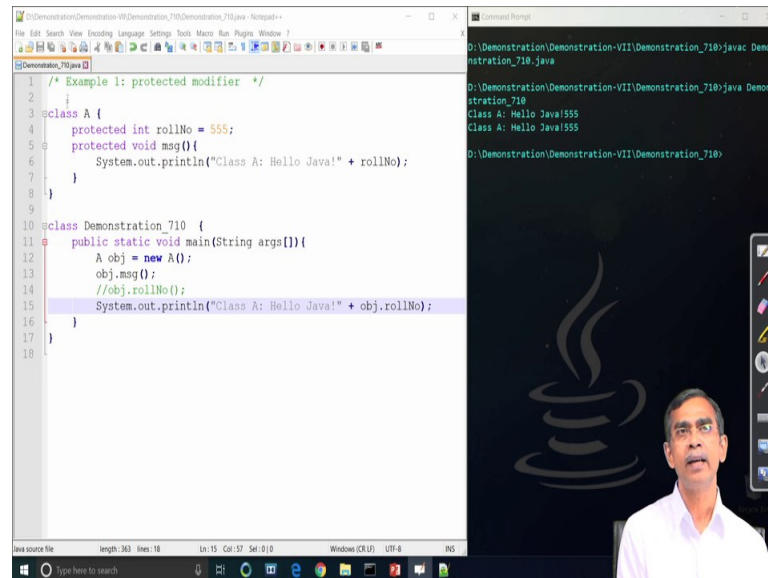
So, here is the error is a compile-time error showing. So, this basically gives an error because a constructor is private if we remove this constructor private may say a public right here public is are not required fine you can run it then save it and then run it first you have to compile yeah then compilation is successful so, it is running yeah. So, it is running there is no object new oh you do not have any free in the method. So, the message is there object right obj msg here.

Now, so, we can create an object which was not possible by means of private constructor, but if it is public it can be accessed here fine. So, this program shows that a constructor cannot be declared as a private in any class if it is there then no object can be created. Now our next demonstration showing the protected access modifier in a class, as



you know the protected access modifier is basically access limitation it limits its access to only the subclasses inherited classes.

(Refer Slide Time: 22:43)



The screenshot shows an IDE with two windows. The left window displays the source code for a Java program. It defines a class `A` with a protected integer attribute `rollNo` and a protected method `msg()`. A second class, `Demonstration_710`, contains a `main` method that creates an instance of `A`, calls `msg()`, and prints the `rollNo` attribute. The right window shows the output of the program, which prints the class name and the roll number. A small video inset in the bottom right corner shows a man speaking.

```
1  /* Example 1: protected modifier */
2
3  class A {
4      protected int rollNo = 555;
5      protected void msg() {
6          System.out.println("Class A: Hello Java!" + rollNo);
7      }
8  }
9
10
11 class Demonstration_710 {
12     public static void main(String args[]) {
13         A obj = new A();
14         obj.msg();
15         //obj.rollNo();
16         System.out.println("Class A: Hello Java!" + obj.rollNo);
17     }
18 }
```

Output:

```
D:\Demonstration\Demonstration-VII\Demonstration_710>javac Demonstration_710.java
D:\Demonstration\Demonstration-VII\Demonstration_710>java Demonstration_710
Class A: Hello Java!555
Class A: Hello Java!555
D:\Demonstration\Demonstration-VII\Demonstration_710>
```

Now, here is an example class regarding the access modification called protected. So, class A is default but, with its 2 members, the protected as an integer roll number and a message as a protected message. So, this is the one class here now have the main class say demonstration\_710 now so, far object creation is concerned. So, it is not an issue because it is a default class, but accessing the protected message obj msg or accessing a protected member value. So, roll number is a problem, let us run this program as obvious this program should not pass the compilation task step.

So, it will give a compilation error yeah as we see an object or roll number it giving say compilation error because cannot find symbol look like this one. So, it is the error is there, but the object is created, however. And another important thing obj.msg, it did not give any error report actually if a method is declared as a protected and if it is in the same file then protected is also accessible, but the protected member is not accessible via this one fine. So, it is not working, but here again, we have done some mistake actually.

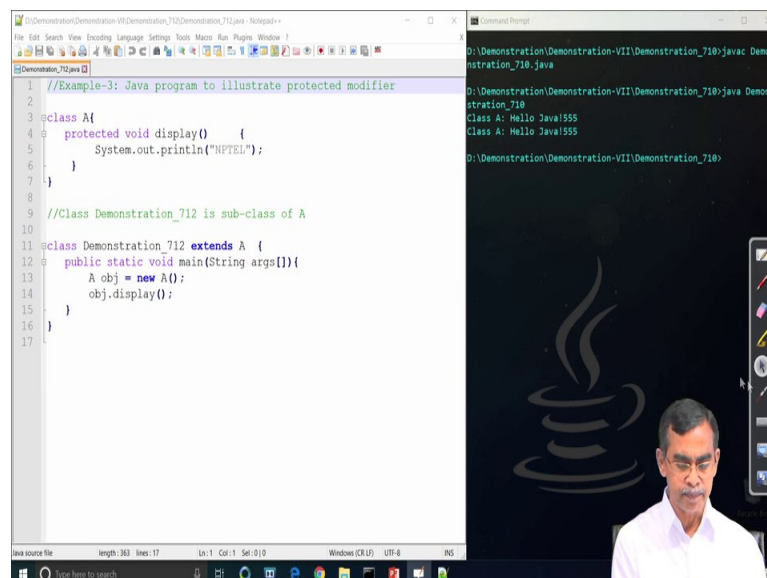
Now, here obj roll number you just see a roll number it is the method you have given not method just gives an obj roll number right. System.out.println() obj roll number just give a system that is the same statement you can call it there. System.out.println() just copy

the object roll number is not a method yeah correct you can type it here that is fine in yeah correct here and then commented object or roll number this is not a valid one.

Now, see what we have done here class A is a default class with protected, but if the class is in the said axis in the same file whatever it is a protected it will be accessible. So, so protected members are accessible in this program, in this case, this program will not give any error it will compile successfully what is the problem you have not run it successfully compile it. Roll number you object obj.roll number right obj.roll number here yeah obj.roll number yeah there is a simple mistake is there let us we have corrected it now let us run it.

So, it is now running correctly. So, what we have understood from this demonstration is that in the same file the protector members are accessible to different classes that are all. Now, again if we make this class in a different file then definitely it will not work as it works in this case. Now, here is the one example showing the same thing here the class a file is kept in the different file if it is a different file that this will not work for you let us have the second demonstration illustrating the protected member is more there what is the 7.12 program let us load it.

(Refer Slide Time: 26:31)



The screenshot shows a Java IDE with two windows. The left window displays the source code for a Java program. The right window shows the output of the program.

```
//Example-3: Java program to illustrate protected modifier
1
2
3 class A {
4     protected void display() {
5         System.out.println("NPTEL");
6     }
7 }
8
9 //Class Demonstration_712 is sub-class of A
10
11 class Demonstration_712 extends A {
12     public static void main(String args[]) {
13         A obj = new A();
14         obj.display();
15     }
16 }
17
```

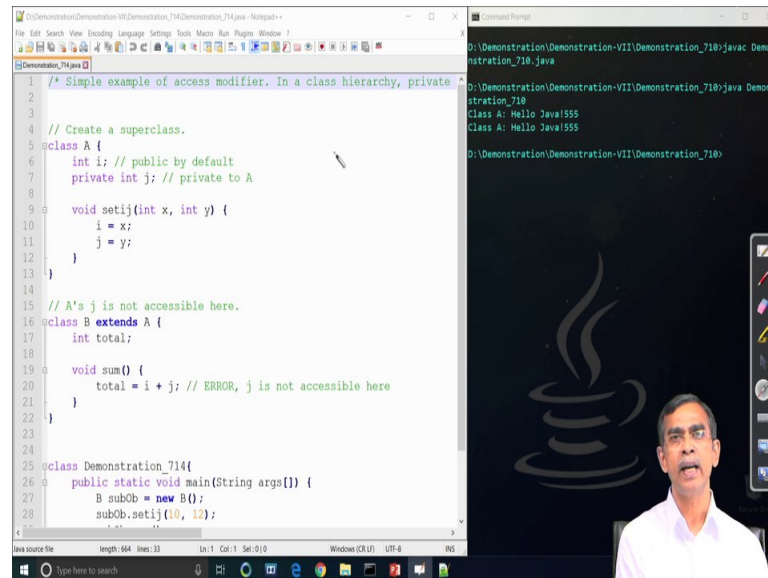
The output window shows the following text:

```
D:\Demonstration\Demonstration-VII\Demonstration_710>javac Demonstration_710.java
D:\Demonstration\Demonstration-VII\Demonstration_710>java Demonstration_710
Class A: Hello Java/555
Class A: Hello Java/555
D:\Demonstration\Demonstration-VII\Demonstration_710>
```

We have mentioned here protected void display and then yeah. So, this will work for us no issue protected method will be accessible to the same class, but if we store in the different class if we store the class A in a different class then it will not be there. Now,

let us see the inheritance whether we have through the inheritance we can access it or not  
let us go to the program 7.14 I think this is right we have to skip it go quick.

(Refer Slide Time: 27:07)



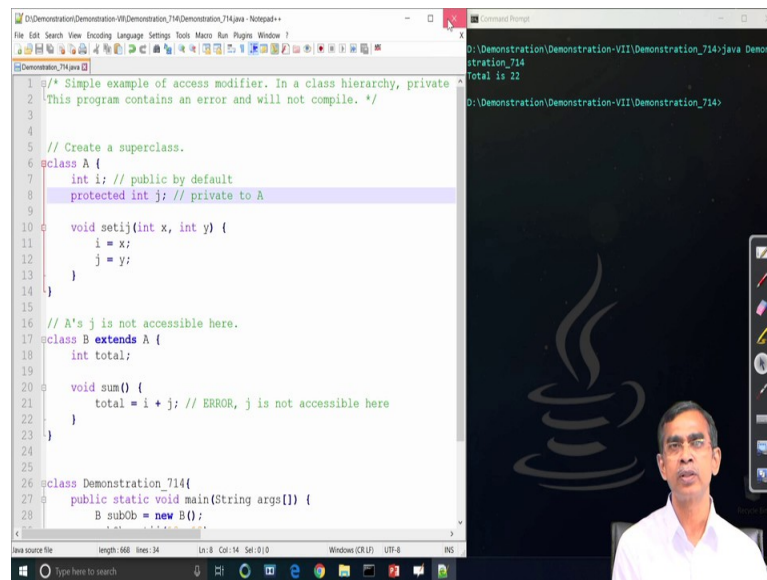
The screenshot shows a Java IDE with two windows. The left window displays a Java source file named 'Demonstration\_714.java'. The code defines a superclass 'A' with a public integer 'i' and a private integer 'j'. It includes a 'setij' method that sets 'i' and 'j'. A subclass 'B' extends 'A' and has a 'sum' method that attempts to calculate the sum of 'i' and 'j'. A comment in the 'sum' method states 'ERROR, j is not accessible here'. The 'main' method creates an instance of 'B' and calls 'setij(10, 12)'. The right window shows the command prompt output, which displays the class hierarchy: 'Class A: Hello Java/555' and 'Class B: Hello Java/555'. A small video inset in the bottom right corner shows a man speaking.

```
1  /* Simple example of access modifier. In a class hierarchy, private  
2  
3  
4  // Create a superclass.  
5  class A {  
6      int i; // public by default  
7      private int j; // private to A  
8  
9      void setij(int x, int y) {  
10         i = x;  
11         j = y;  
12     }  
13 }  
14  
15 // A's j is not accessible here.  
16 class B extends A {  
17     int total;  
18  
19     void sum() {  
20         total = i + j; // ERROR, j is not accessible here  
21     }  
22 }  
23  
24  
25 class Demonstration_714 {  
26     public static void main(String args[]) {  
27         B subOb = new B();  
28         subOb.setij(10, 12);  
29     }  
30 }
```

Now, so, the protected member has its more implication in the context of inheritance this is one example; so that how the member protected can be access to the inherited class derived class or subclass. So, class A is a superclass in this case where j is a private as is private it cannot be accessed to any other class other than this method in this class. And so, this is a method to initialize the object i and j in discussing. So, it is not an issue because it can access it on the member.

Now, here B class extends A so, here and it has its own member one total as an integer and its method sum .Now, here total I plus j as you understand that j being a private. So, this is and compilation error this leads to a compilation error. So, it will not a work here so, a private method cannot be accessible to any derived class right here. So, this will give an error ok.

(Refer Slide Time: 28:15)



The screenshot shows a Java IDE with a code editor on the left and a terminal window on the right. The code editor displays a Java program with the following content:

```
1  /* Simple example of access modifier. In a class hierarchy, private  
2  * This program contains an error and will not compile. */  
3  
4  
5  // Create a superclass.  
6  class A {  
7      int i; // public by default  
8      protected int j; // private to A  
9  
10     void setij(int x, int y) {  
11         i = x;  
12         j = y;  
13     }  
14 }  
15  
16 // A's j is not accessible here.  
17 class B extends A {  
18     int total;  
19  
20     void sum() {  
21         total = i + j; // ERROR, j is not accessible here  
22     }  
23 }  
24  
25  
26 class Demonstration_714 {  
27     public static void main(String args[]) {  
28         B subObj = new B();  
29     }  
30 }
```

The terminal window on the right shows the output of the program:

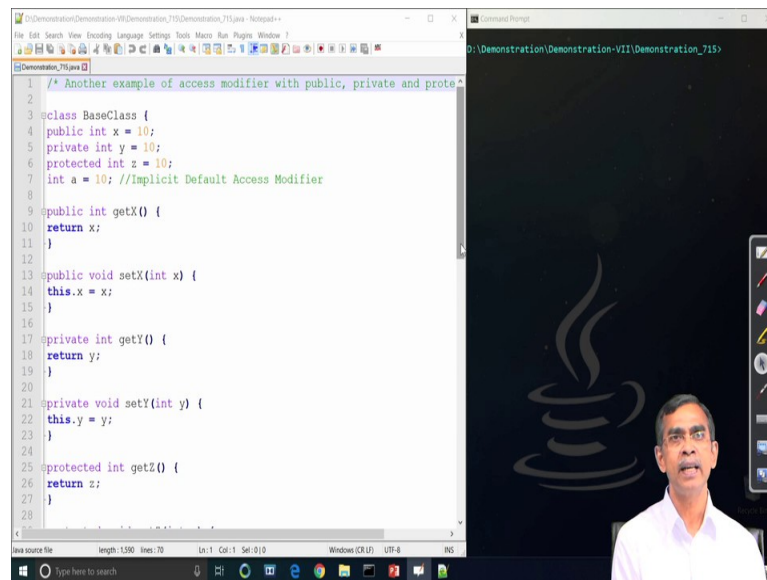
```
D:\Demonstration\Demonstration-VII\Demonstration_714>java Demon  
stration_714  
Total is 22  
D:\Demonstration\Demonstration-VII\Demonstration_714>
```

A small video inset in the bottom right corner shows a man speaking.

So, it is giving an error because the `j` is not accessible there. Now, if we make it protected for example, so, instead of private let us go here protected now see this error can be eliminated here in this case because protected member is readily accessible to the derived class here. So, run this program and now this program will work for you.

So, this program works correctly and we can see that a protected member is accessible to the inherited class. Now, let us have a little bit bigger one example who which includes many other methods with many access modifier 7.15 is the last demo last, but one demo fine let us see the program here.

(Refer Slide Time: 29:15)



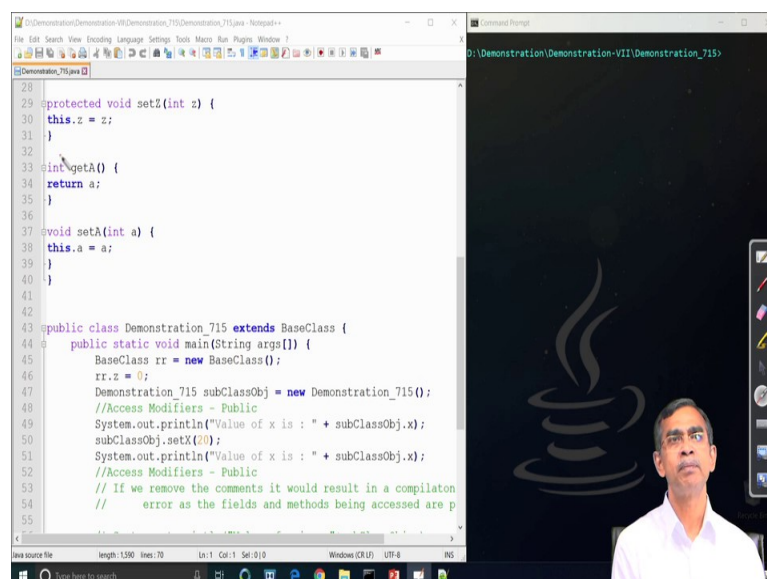
The screenshot shows a video lecture interface. On the left, a code editor displays the following Java code:

```
1  /* Another example of access modifier with public, private and protected */
2
3  class BaseClass {
4      public int x = 10;
5      private int y = 10;
6      protected int z = 10;
7      int a = 10; //Implicit Default Access Modifier
8
9      public int getX() {
10         return x;
11     }
12
13     public void setX(int x) {
14         this.x = x;
15     }
16
17     private int getY() {
18         return y;
19     }
20
21     private void setY(int y) {
22         this.y = y;
23     }
24
25     protected int getZ() {
26         return z;
27     }
28 }
```

On the right, a presenter is visible in a small window, with a large Java logo in the background.

Little bit large program here, but the program is readily understandable here. So, we have declared one class called the base class which its members public one, private and protected and integer is a default. Then here is a method public declare as a public getX and then another method setX is also public and another method getY to private it is basically initialized to private value it is a private method and setY is also another private method that basically to initialize the private elements in it. And then getZ is a protected; that means, dealing with the protected members there and then setting the value and then getA return A.

(Refer Slide Time: 29:59)



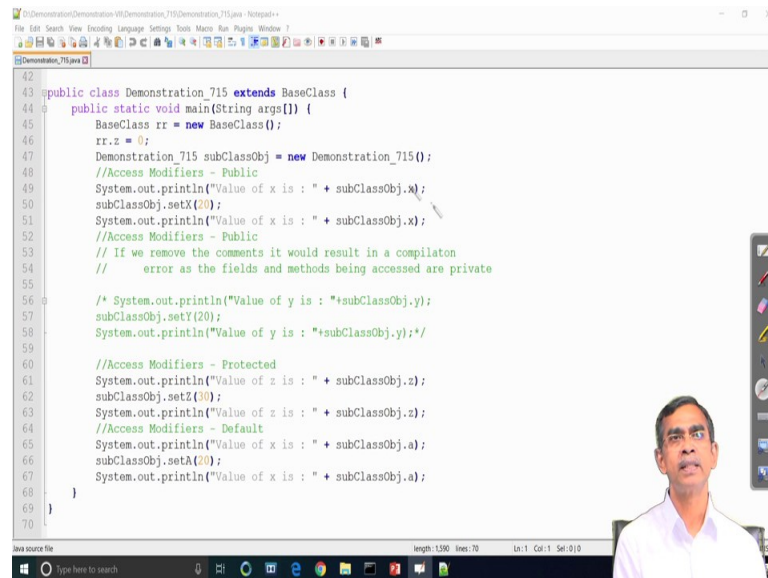
The screenshot shows a video lecture interface. On the left, a code editor displays the following Java code:

```
28
29 protected void setZ(int z) {
30     this.z = z;
31 }
32
33 int getA() {
34     return a;
35 }
36
37 void setA(int a) {
38     this.a = a;
39 }
40
41
42
43 public class Demonstration_715 extends BaseClass {
44     public static void main(String args[]) {
45         BaseClass rr = new BaseClass();
46         rr.z = 0;
47         Demonstration_715 subClassObj = new Demonstration_715();
48         //Access Modifiers - Public
49         System.out.println("Value of x is : " + subClassObj.x);
50         subClassObj.setX(10);
51         System.out.println("Value of x is : " + subClassObj.x);
52         //Access Modifiers - Private
53         // If we remove the comments it would result in a compiler
54         // error as the fields and methods being accessed are p
55     }
56 }
```

On the right, a presenter is visible in a small window, with a large Java logo in the background.

A is basically the one default values are set default here. So, we have declared four members public, private, protected and default and then eight different methods regarding they are accessing.

(Refer Slide Time: 30:19)



```
42
43 public class Demonstration_715 extends BaseClass {
44     public static void main(String args[]) {
45         BaseClass rr = new BaseClass();
46         rr.z = 0;
47         Demonstration_715 subClassObj = new Demonstration_715();
48         //Access Modifiers - Public
49         System.out.println("Value of x is : " + subClassObj.x);
50         subClassObj.setX(20);
51         System.out.println("Value of x is : " + subClassObj.x);
52         //Access Modifiers - Public
53         // If we remove the comments it would result in a compiler
54         // error as the fields and methods being accessed are private
55
56         /* System.out.println("Value of y is : "+subClassObj.y);
57         subClassObj.setY(20);
58         System.out.println("Value of y is : "+subClassObj.y);*/
59
60         //Access Modifiers - Protected
61         System.out.println("Value of z is : " + subClassObj.z);
62         subClassObj.setZ(30);
63         System.out.println("Value of z is : " + subClassObj.z);
64         //Access Modifiers - Default
65         System.out.println("Value of x is : " + subClassObj.a);
66         subClassObj.setA(20);
67         System.out.println("Value of x is : " + subClassObj.a);
68     }
69
70 }
```

And here is basically if we remove the comment then those are the comments basically creates a problem. Now, let us have a quick look about the main method in this main method we create an object of the class base class r r and then r r.z as you know z is a default 1. So, rr.z is accessible here so, this is not an error so, it will first.

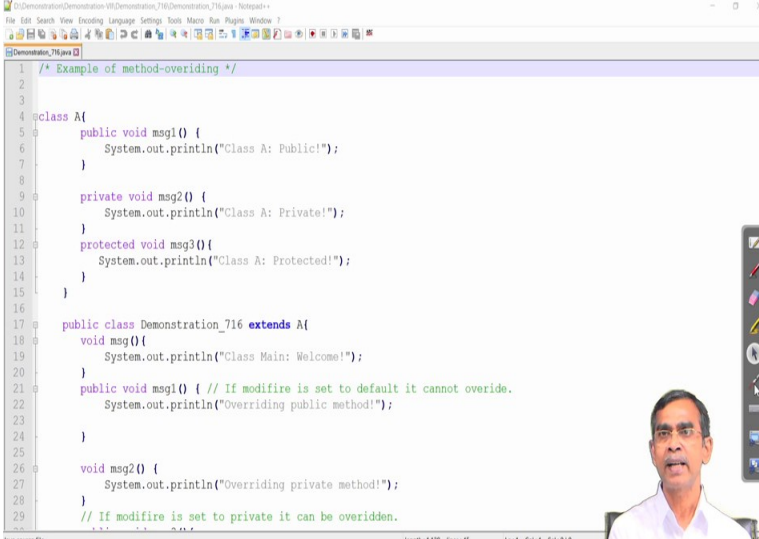
Now, we create another object of the main class method say demonstration 71 subclass object new this one and for the subclass object we are trying to access the different method get to set all these things are there. So, for the access modifier is public is concerned system out.println() it is not an issue it will access because public method public values are accessible to here. So, setX is being a public method it will accessible here similarly getX also will be accessible here. So, this is not an issue.

Now, so, far the access modifier public is concerned they are accessible; however, there will be an error if we removed the following comments in the following line right. For example, system.out.println() value of y is subclass object y y here y is a private as if the private cannot be accessed by any subclass object here. So, that cannot be accessed here. So, if we remove this comment so, it will set an error. So, in that case, subclass object setY also not a privileged a method for this class and then we cannot access it here.

Then so, far the protected is concerned. So, as it is a derived one so, extent because you see demonstration\_715 extent base class. So, derived class this means that all the protected member and the protected methods are accessible. So, they all will access here. Now so, far the default access specification is concerned in this class default is in the same file. So, it will not be an error so, it will execute successfully.

So, this is the understanding of the different access modification. So, far the program in Java is concerned now I will just conclude this demonstration with the last example this is related to the method overriding; whether you can override some method having some access specification which is already specified with some other access specifier.

(Refer Slide Time: 32:37)



```
1  /* Example of method-overriding */
2
3
4  class A{
5      public void msg1() {
6          System.out.println("Class A: Public!");
7      }
8
9      private void msg2() {
10         System.out.println("Class A: Private!");
11     }
12     protected void msg3(){
13         System.out.println("Class A: Protected!");
14     }
15 }
16
17 public class Demonstration_716 extends A{
18     void msg(){
19         System.out.println("Class Main: Welcome!");
20     }
21     public void msg1() { // If modifier is set to default it cannot override.
22         System.out.println("Overriding public method!");
23     }
24
25
26     void msg2() {
27         System.out.println("Overriding private method!");
28     }
29     // If modifier is set to private it can be overridden.
```

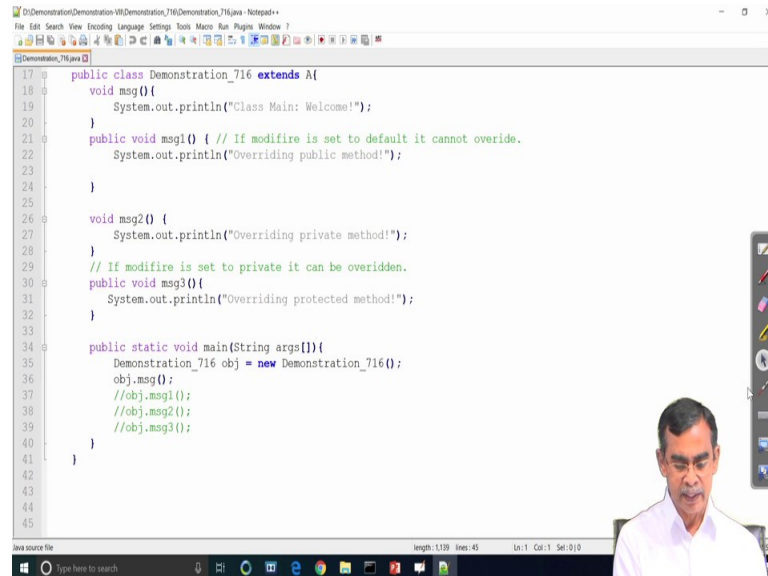
The basic concept is that a method can override it by one weakest one if it is by if an access specifier is weakest one then the strongest one then it can be specified by the weakest one, but the reverse is not possible. So, this is the concept it is basically upward compatibility it is there.

Now, so, far the different access is concerned private is the strongest and after the private protected and after the protected default and the weakest is the public there. So, if we declare a public, later on, we can method override as a default if we declare is a public overridden as a private can be possible. But if it is a protected and then accessing that is a default it is not possible because the default is weaker then the protected and vice versa



similarly public is weakest. So, public default is there then may overriding as a default is possible, but the reverse is not possible.

(Refer Slide Time: 33:49)



```
17 public class Demonstration_716 extends A{
18     void msg(){
19         System.out.println("Class Main: Welcome!");
20     }
21     public void msg1() { // If modifier is set to default it cannot override.
22         System.out.println("Overriding public method!");
23     }
24
25
26     void msg2() {
27         System.out.println("Overriding private method!");
28     }
29     // If modifier is set to private it can be overridden.
30     public void msg3(){
31         System.out.println("Overriding protected method!");
32     }
33
34     public static void main(String args[]){
35         Demonstration_716 obj = new Demonstration_716();
36         obj.msg();
37         //obj.msg1();
38         //obj.msg2();
39         //obj.msg3();
40     }
41 }
42
43
44
45
```

So, this demonstration has a quick one example that giving the same notation here. Here we declare one class here the class A s class A and in this class, we have the method message one as a public msg 2 method as a private and msg 3 is another method as protected. So, the three different methods with three different access specifier we have not used any default access specifier here in this case, but we could do that anyway.

So, let us come to the inherited class extends a now by virtue of inheritance in this class all the method is available except the private method protected is available and then the public is also accessible there. Now, in this method, we have declared one method is a default method message it is not an issue. Now, here if you see msg one which is declared in its base class as a private, but here we are declaring as a public.

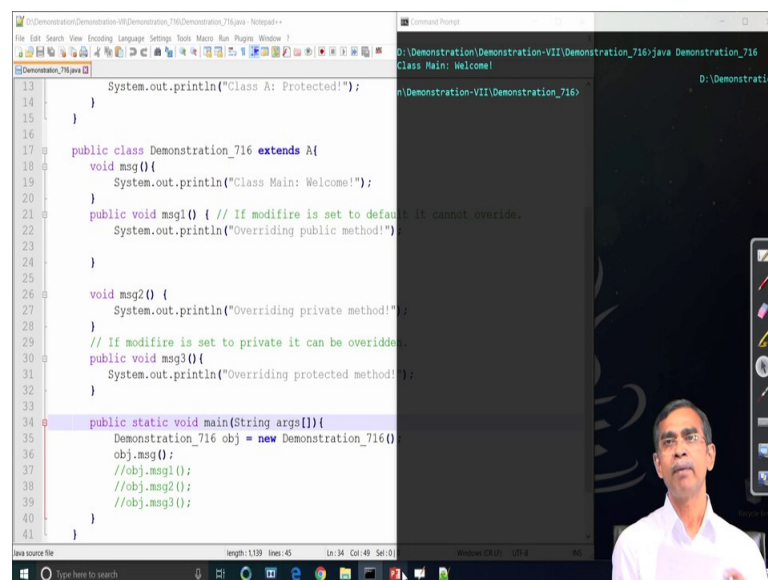
So, that overriding the method you go to the previous class the base class please message 1 yeah. So, message 1 you see message 1 is declared as public. So, if it is a public overriding is quite possible public to public same access level method overriding is possible let us go to the main class here main method main class yeah so fine.

So, message 1 overriding is possible, .Now, here msg 2 in base class this method msg 2 is coming as private right go their private method. So, here we are going to an override it

as a private we can override it, but other than private we cannot override because it is the highest access phase one it cannot be overridden. Now let us go to the next one yeah. So, msg 3 which is a protected declare as a protected, but here we can override as a what is called the public? So, that is possible.

Now, so, with these things we can override it and then once the overwritten is there, we can create the object of these classes and then access it. Now let us compile this and let us see what is the output this program it gives you. So, this basically gives a hierarchy of access overriding method overriding with the different access specifier.

(Refer Slide Time: 36:15)



The screenshot shows a Java IDE with a source file named 'Demonstration\_716.java'. The code defines a class hierarchy where 'Demonstration\_716' extends 'A'. It contains several methods: 'msg()', 'msg1()', 'msg2()', 'msg3()', and a 'main' method. Comments indicate that 'msg1()' cannot be overridden (default), 'msg2()' can be overridden (private), and 'msg3()' can be overridden (protected). The 'main' method creates an object of 'Demonstration\_716' and calls these methods. To the right, a 'Command Prompt' window shows the output: 'Class A: Protected!', 'Class Main: Welcome!', and 'Overriding public method!'. A small video inset in the bottom right corner shows a man speaking.

```
13 System.out.println("Class A: Protected!");
14 }
15 }
16
17 public class Demonstration_716 extends A{
18     void msg(){
19         System.out.println("Class Main: Welcome!");
20     }
21     public void msg1() { // If modifier is set to default it cannot override.
22         System.out.println("Overriding public method!");
23     }
24
25     void msg2() {
26         System.out.println("Overriding private method!");
27     }
28     // If modifier is set to private it can be overridden.
29     public void msg3(){
30         System.out.println("Overriding protected method!");
31     }
32 }
33
34 public static void main(String args[]){
35     Demonstration_716 obj = new Demonstration_716();
36     obj.msg();
37     //obj.msg1();
38     //obj.msg2();
39     //obj.msg3();
40 }
41 }
```

Command Prompt Output:

```
D:\Demonstration\Demonstration-VII\Demonstration_716> java Demonstration_716
Class A: Protected!
Class Main: Welcome!
Overriding public method!
```

So, actually for a so, here you can see you know there is a problem you can just see there is a problem in the 716 compile is successful. Now, what I want to mention as the last notice note is that access modification is not a critical job for there you specify which method which member you want to give access to for which one. But the compilation itself take care of all this invalid or illegitimate any access specification. For a Java program, it is a great relief that if we have wrong things and then the program is built up, but this is not a violation because compilation time only it will resolve if we specify according to one.

So, this is the demonstration about the access specifications and we have learned about how the information hiding in access specification is can be done. And; obviously,

practice will improve your understanding I advise you to go for rigorous practice for this.  
So, with this, I want to stop it here today.

Thank you very much.