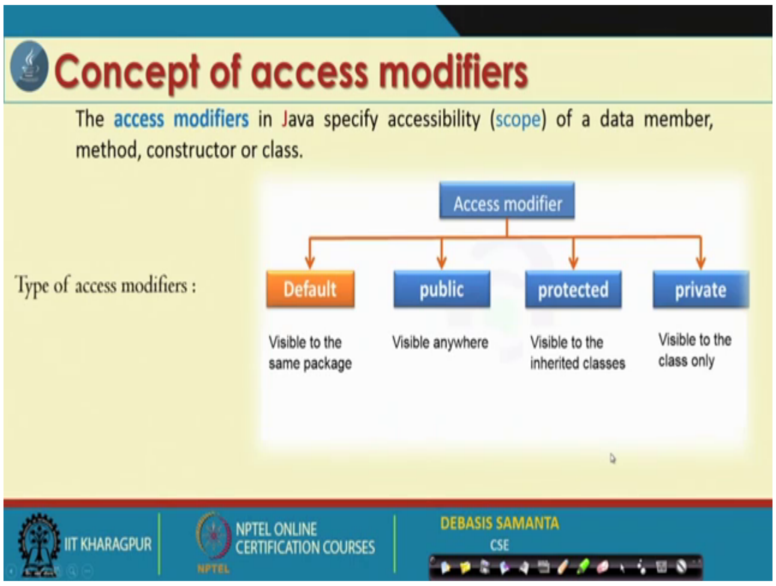
Programming in Java
Prof. Debasis Samanta
Department of Computer Science and Engineering
Indian Institute of Technology, Kharagpur

Lecture – 15
Information Hiding

So, in this module we will learn about Information Hiding in Java. The concept information hiding is a very interesting as well as important concept. So, our today's basically the information hiding; that means, what are the different way that a programmer can control the access of different elements, different elements means the method and the members that belongs to a class.

(Refer Slide Time: 01:01)



Concept of access modifiers

The access modifiers in Java specify accessibility (scope) of a data member, method, constructor or class.

Type of access modifiers :

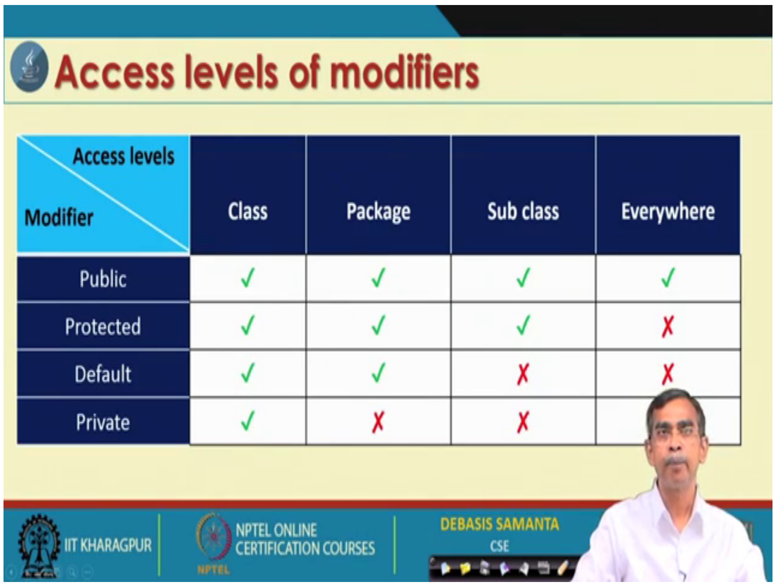
Access modifier
Default
Visible to the same package
public
Visible anywhere
protected
Visible to the inherited classes
private
Visible to the class only
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE

So, for this the Java developer has coined a number of modifiers, they are called access modifiers. So, in this lectures we will try to understand the different access modifiers and then their usage. Now so far the access modifier are concerned there are 4 types, they are called default, public, protected and private. So, if we fixed one access modifier before a member for example, before a field or a value or a data or a method then according to the access specifier that value or the method will be accessed. So, if it is public then it should be visible to anywhere so; that means, any other class from any other class or any method belongs to any class can access that kind of members.

On the other hand, if it is protected then that member cannot be accessed by all classes that member can be accessed by any method that belongs to any inherited sub classes. On the other hand, if it is a private than that member cannot accessed by any other method other than the method in that class itself. So, these are the 3 public protected and private and then default, there are possibilities that without making any specifier explicitly then it is called the default access specifier. That means, if we do not use public protected private, just keep it blank void then the access specifier is called the default. If it is a default then that members will be accessible to any class that belongs to the same directory, where that class belongs or it is basically belong to the same package.

So it is basically, either same file or it is a same class belongs to the same directory the access will be available otherwise, it cannot be accessed. So, these are the 3 4 different access modifiers and we will just discuss about all these modifiers one by one with examples. So, then you can understand about it. So, we can say in other words that access modifier specify the scope that a member have, whether scope is within that class or scope is within that directory or scope is anywhere. So, this is the concept that is there.

(Refer Slide Time: 03:41)



Access levels of modifiers

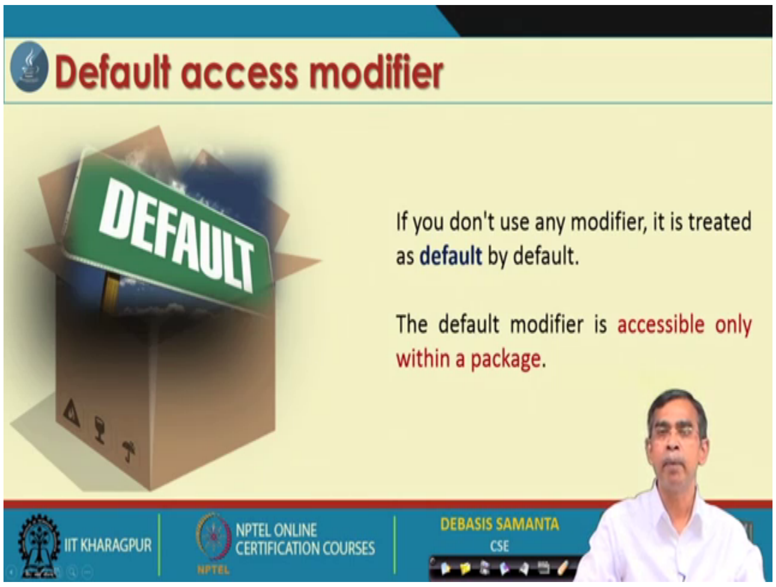
Access levels / Modifier	Class	Package	Sub class	Everywhere
Public	✓	✓	✓	✓
Protected	✓	✓	✓	✗
Default	✓	✓	✗	✗
Private	✓	✗	✗	[illegible]

Now here, the scope can be understood from this table, we have mentioned that access level and the different modifiers that we have discussed 4 modifiers. Now so far the public is concerned as we see any class, they have the access to public member, any package also it has the access to that member any sub class and it can be from anywhere.

So, if it is a public means it is a global look like. So, anybody can see it, can use it, can print it, can change the value whatever it is there. On the other hand, if it is a protected within the same class any method can access this protected data then protected member within the same package, the protected is accessible within the sub class the protected is accessible; however, protected is not accessible, other than these 3 things from anywhere it cannot be. So, if the two class files are same package then protected member can be accessed in any classing, it belongs to that same package.

Now, let us come to the private as I told you private is the stricter access specification, it is accessible a member which is declared as a private is accessible to that class only where that member is declared as a private outside this class that member cannot be accessible either it is packaged or it is sub class or anywhere. Now the default as I told you, default member is accessible to the same class, where it is declared as a default and also in the same directory it is called the package, where that class is that member is declared as a default access specification. So, this table I suggest you to remember it is very informatic table, it can give you the whole summary of the different modifiers.

(Refer Slide Time: 05:45)



Default access modifier

If you don't use any modifier, it is treated as **default** by default.

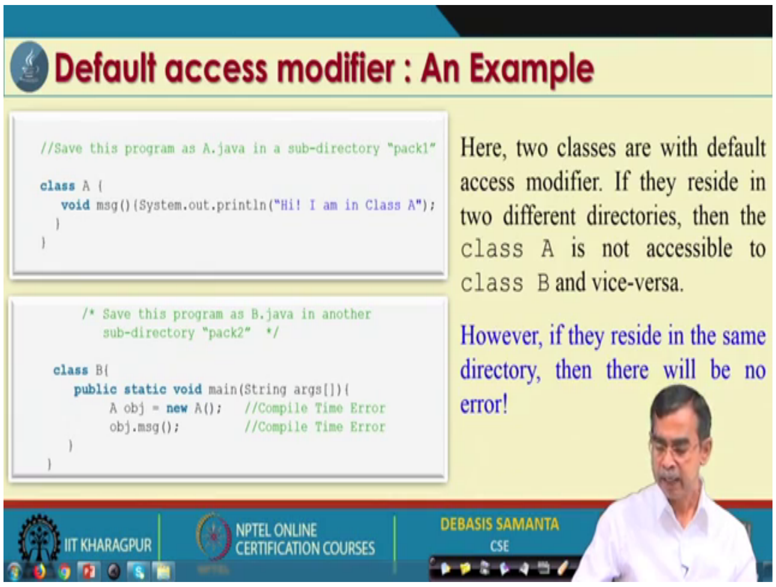
The default modifier is accessible only within a package.

DEFAULT
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE

Now, let us have a quick look over the different modifiers one by one, first we shall discuss about default access modifier. So, it is a default means we do not have to mention any access specification explicitly before a member that is a data or a method it is without mentioning these things it can be and as I told you if the default is an access specifier for a member; that means, it is accessible only within the same class or within the same package; that means, package concept is little bit not known to you, regarding this package our next lecture has been planned.

So, there we will discuss about the package, package is a collection from Java files actually. So, there maybe all files belongs to one subdirectory or a directory. So, a directory we can say roughly as a package, but package has many more other implication. So now, you can understand a package, if we refer to it means that a package is a set of files, which are belongs to a common directory, one directory.

(Refer Slide Time: 06:53)



Default access modifier : An Example

```
//Save this program as A.java in a sub-directory "pack1"

class A {
   void msg(){System.out.println("Hi! I am in Class A");
  }
}
```

```
      /* Save this program as B.java in another
         sub-directory "pack2"  */

  class B{
     public static void main(String args[]){
          A obj = new A();   //Compile Time Error
          obj.msg();         //Compile Time Error
     }
  }
```

Here, two classes are with default access modifier. If they reside in two different directories, then the `class A` is not accessible to `class B` and vice-versa.

However, if they reside in the same directory, then there will be no error!

Anyway so, the default access specifier, we are going to discuss about it, let us have a quick demo a quick look of the two classifies that we have discussed here. Now, here we discussed two class file and in one class file and we say that it is the on one class file, we declare one class this class is called class A and it is stored in a file, let us name of the file be A dot java.

So, this is the one class, we have declared and stored it is in the form in the file called A dot java. Now this is a very simple one class that we have discussed for only demonstration or you can say for illustration. So, actually this class can be very complex anyway. So, this a simple class A stored in A dot java file and here is the another file another file, we can say the name of the file as B that B dot java and here the class B is declared here.

Now here, you can note two things here, the class that we have declared here and these does not have any specifier nor this or also does not have any specifier. So, here we can see both the class a without any access specification. So, what I want to say in other word is that, access specifier can be prefixed before the class declaration, before a method and before a data member as in this case only the class and then one method belongs to this. So, we can specify the access specifier either here or here or both whatever it is there.

Now, in this case as we see there is no access specifier mentioned. So, if no access specifier mention then it does mean that the class A is declared as a default similarly, the method message which return type is void does not have any access specification this means that it is default by default. So, access specification for both class A and method msg belong to that class A is default access specification.

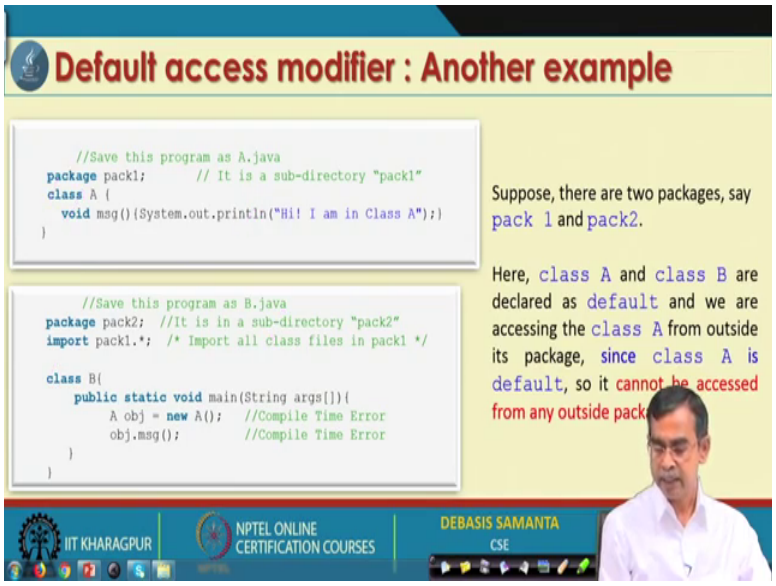
Now, let us come to the class B again, here there is no access specification. So, there is no access specification, it means that class B is also by default it is a default access specification. Now this is the user method that public, static, void as we have already discussed about it and here always we have to mention that public access specification, this is obvious and without any hesitation you have to do it, because without these things no one can run the program particularly the main class, if it is there.

So, this is the public access specifier by default is there and thus rest of the things are just simply the accessing of the different. Now in the first statement, we see we create an object of class A here and in this method main, we call the method msg, which belongs to the class object; that means, which belongs to class A. So, if we run this program it will print Hi I am in class A, this kind of message is there.

Now so, if these are the two files belongs to the different directory then access specifier will not work for you, if they are in the same file, if we maintain these and these in the same file say B dot java then access specifier default acts will work for you then in this case, these two will give will not give any error, but if they maintain in the different file, but not in the same directory then it will produce an error. So, these will be an error.

But if again, it produce into the same sub different sub directory, it is an error and if it is in the same directory there is no error. So, there are two things that you should consider default access, where if both the classes are belong to the same file. So, there will be no error, if they are belong to the different sub directory then there will be an error, if they are belong to the same directory then there will be no error. So, these are the 3 things that you should consider with access specification. So, this is about the default access modifier.

(Refer Slide Time: 11:31)



Default access modifier : Another example

```
    //Save this program as A.java
package pack1;        // It is a sub-directory "pack1"
class A {
  void msg(){System.out.println("Hi! I am in Class A");}
}
```

```
    //Save this program as B.java
package pack2;  //It is in a sub-directory "pack2"
import pack1.*;  /* Import all class files in pack1 */

class B{
    public static void main(String args[]){
        A obj = new A();    //Compile Time Error
        obj.msg();          //Compile Time Error
    }
}
```

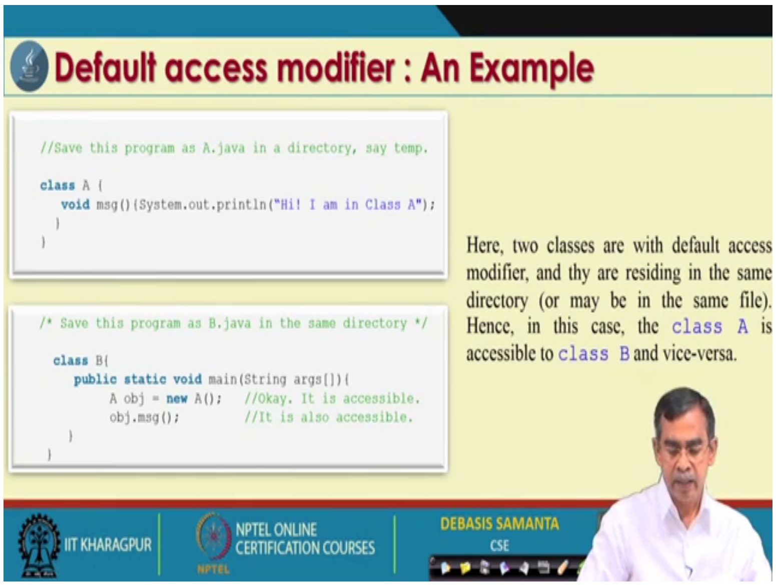
Suppose, there are two packages, say pack 1 and pack2.

Here, class A and class B are declared as default and we are accessing the class A from outside its package, since class A is default, so it cannot be accessed from any outside pack

Now, let us have another example of this one and here we can see we have little bit different concept I have introduced here ok. So, here is a package my package as I told you, in which directory this class belongs, we have to specify this one. So, that is why this concept is there this indicates that this file that mean A dot java belongs to one sub directory called pack 1. So, that is why the package now regarding package statement, we will learn a lot later on. So, this in these case, what we have done is that this program is shaped as A dot java in the sub directory pack 1.

Now again, this is another program same save this program as B dot java, it is in the sub directory pack 2. Now, as you see this program and this program, they are belong to the different sub directory that mean default access specifier, this one or this one will leads to a compilation error. So, this is the things that we have already discussed earlier. So, this is the concept.

(Refer Slide Time: 12:41)



Default access modifier : An Example

```
//Save this program as A.java in a directory, say temp.

class A {
   void msg(){System.out.println("Hi! I am in Class A");
   }
}
```

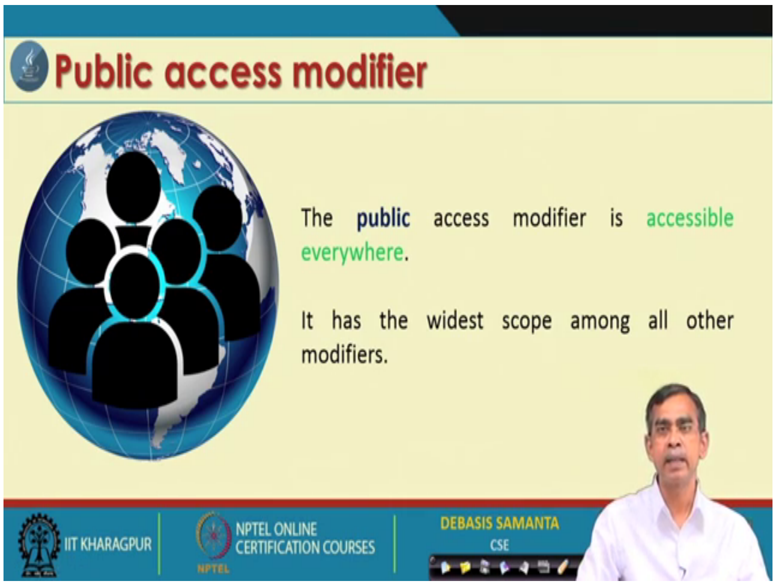
```
/* Save this program as B.java in the same directory */

  class B{
     public static void main(String args[]){
          A obj = new A();    //Okay. It is accessible.
          obj.msg();          //It is also accessible.
     }
  }
```

Here, two classes are with default access modifier, and thy are residing in the same directory (or may be in the same file). Hence, in this case, the class A is accessible to class B and vice-versa.

So now, access specification is all about this one. Now here, again you can understand that in this case, there will be no error this is because, if you see here this program, we save A dot java in a sub directory say 10 and here this program B jot java is a same directory as both the class files are in the same directory. So, default access specifier works; that means, in this case they will not give any error.

So, this are the 3 things that we have to consider about or is the same class files in the same directory files in different directory. So, this is the concept and then access specifier will work for you.

(Refer Slide Time: 13:17)



Public access modifier

The public access modifier is accessible everywhere.

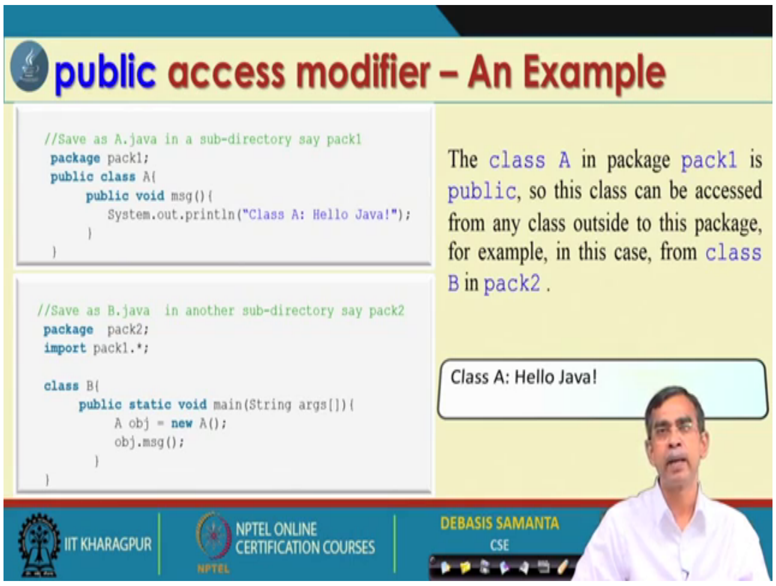
It has the widest scope among all other modifiers.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE

Now, let us come to discuss about another modifier, it is called the public access modifier. Now so far, the stricter since is concerned public specifier is basically in a very strict access specification very weak. Next after public the protected then private and finally, default is basically sorry public then protected default then private in that sense. So, as you know private is a most stricter access specification compared to the public, public is the weakest access specification in this range, you can consider the table that I have already discussed that table, you can follow it.

Any way so, if we declare any class or any data method as a public this means that that class will accessible to any other class, whether this class belongs to the same directory a class into the same directory or belongs to class in any other directory. So, that is the concept it is here. So, it has the scope among all it has a weakest scope among all other modifier.

(Refer Slide Time: 14:35)



public access modifier – An Example

```
//Save as A.java in a sub-directory say pack1
 package pack1;
 public class A{
      public void msg(){
         System.out.println("Class A: Hello Java!");
      }
 }
```

```
//Save as B.java  in another sub-directory say pack2
 package  pack2;
 import pack1.*;

 class B{
      public static void main(String args[]){
           A obj = new A();
           obj.msg();
       }
 }
```

The class A in package pack1 is public, so this class can be accessed from any class outside to this package, for example, in this case, from class B in pack2 .

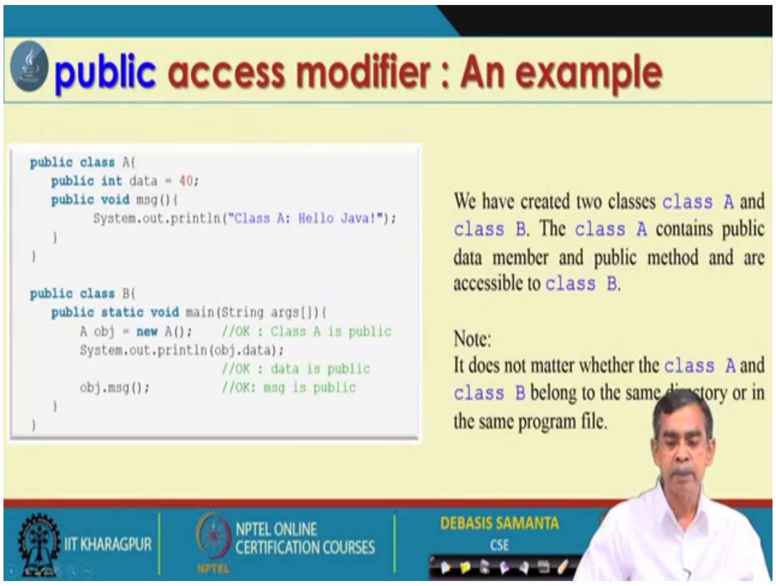
```
Class A: Hello Java!
```

Now, let us consider a simple example first here or we can see this is the class A we have declared A jot java and in a sub directory say pack 1 and here we can see we have declared the class A as public and now another class B, which is an default access specifier say and these class is the save as B dot java in another directory say pack 2. So, there are two classes class A and class B both belongs to the different directory whereas, class A having the public access specifier.

Now, if it is like this then let us come to this one accessing of the members, which belong to the class A. So in that case, this will not be an error because, this is public and this method is also declared as a public as the method is public, whatever then this method is accessible by any class which belong to the same directory or any other directory outside that directory. So, this is the concept about the public. So, public is the weakest access specification as we have already mentioned. So, in that case you can see this is the output that it will give, I have to the obvious. So, there will be no compilation error.

Now again, if there is a what is called the restriction, which is not met then this will be reported not during the run time rather during the compiled time; that means, you will not be able to compile this program successfully, if it highlights some access specifier as per the rule.

(Refer Slide Time: 16:21)



public access modifier : An example

```
public class A{
   public int data = 40;
   public void msg(){
         System.out.println("Class A: Hello Java!");
   }
}

public class B{
   public static void main(String args[]){
       A obj = new A();     //OK : Class A is public
       System.out.println(obj.data);
                            //OK : data is public
       obj.msg();           //OK: msg is public
   }
}
```

We have created two classes `class A` and `class B`. The `class A` contains public data member and public method and are accessible to `class B`.

Note:
It does not matter whether the `class A` and `class B` belong to the same directory or in the same program file.

Now so, this is the public access modifier, let us come to another example here again, we declare class A as a public and there is again another class B both are declared as a public. So here, we create take two class classes class A and class B and in class A, there are public data member and then public method, this means that the entire class is basically accessible to any other classes anywhere actually any classes. So, according to this if this class is in the same file not necessarily to be same file rather, if it is in the same directory rather or we can say in the other directory also no problem, all these are quite there will be no compilation error.

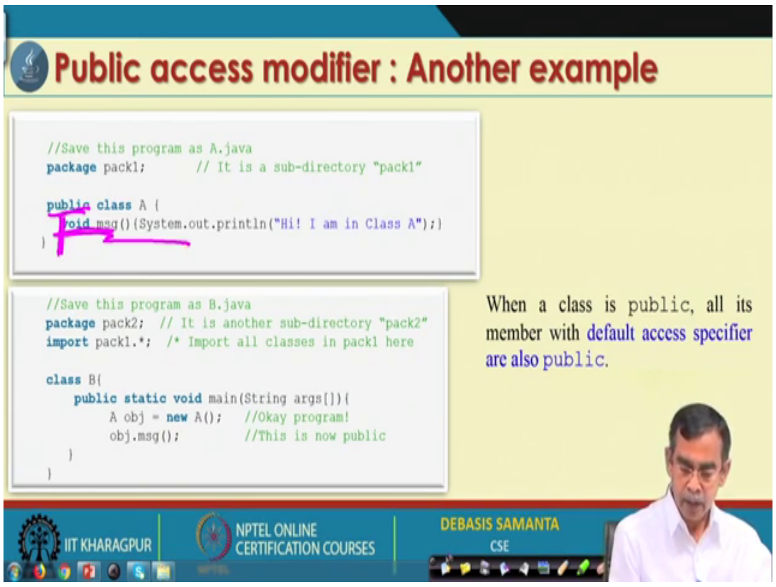
Now again, I just want to mention one more important thing that you may face sometimes that if we maintain these two classes here, these classes and these classes, in the same file then the Java compiler does not allow to specify all the classes, which are belong to the same class by any other access specifier; that means, if class A and class B, if we type using same program and save as a say class B dot java then that public specifier, you cannot mention because, in the same file there is no access restriction.

Actually because, you are a programmer, you are writing this class, you are using this class then why you should mention any access specification, there is no meaning actually. So, in that case the Java developers suggest you to make it default. So, no access specific provided that if you declare all classes belong to the same file.

So again, my advice is that whenever you write two or more classes and save all the classes in the same dot Java file, you should have give the access to an default do not put any access specification other than default that is the concept you will be able to you will not be able to run this program. In fact, compile this program in fact successfully. So, these will report an error usually people follow this one.

So, public is not required until and it is a not a good practice actually, it is a good practice for any programmer is that if you create any class all these class, you should maintain in a separate file. So, all files can be maintained individually isolated way and then all these files can be accessed in a your main class that is the practice good practice of course. So, in that case all those access modification matters actually, if you store all the files in one file, all access modification does not matters too much because, it is public in any way.

(Refer Slide Time: 19:21)


Public access modifier : Another example

```
//Save this program as A.java
package pack1;        // It is a sub-directory "pack1"

public class A {
  void msg(){System.out.println("Hi! I am in Class A");}
}
```

```
//Save this program as B.java
package pack2;  // It is another sub-directory "pack2"
import pack1.*;  /* Import all classes in pack1 here

class B{
    public static void main(String args[]){
        A obj = new A();   //Okay program!
        obj.msg();         //This is now public
    }
}
```

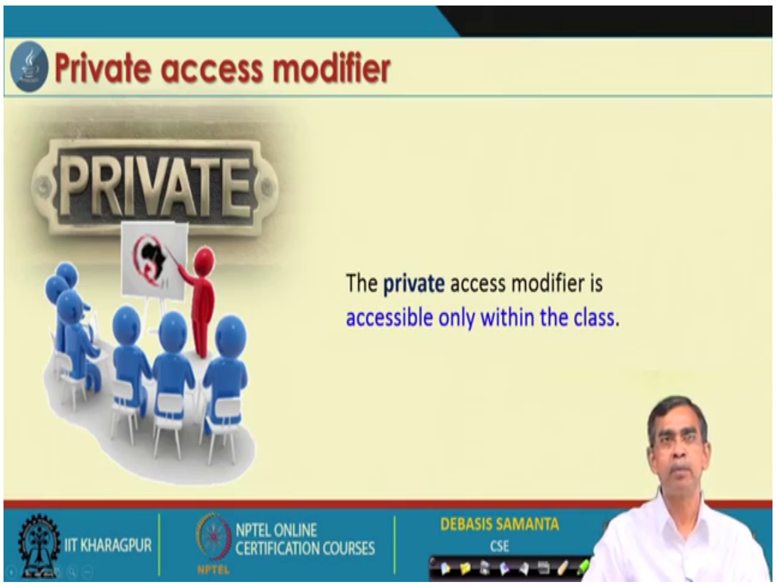
When a class is public, all its member with default access specifier are also public.

Anyway so, this is the concept about the public access modifier and here is the same, you will be able to understand here, we define the class A as A dot java in one sub directory and here the class B also saved as another sub directory and as we main make here is a public. So, it is a public; that means, from any class which belongs to any other sub directory, this class A with all it is member will be accessible.

So, this is the concept here so, that is this one is look what like this. So, this is the public access modification is very easy. So, public if you mentioned it then it can be accessible to anyone there. Now one more thing that I want to mention here is that, if you declare a class as a public for example, here we declare a class as a public, if we declare a class as a public and if you have the access the other members under this class without any access specification; that means, if they are the default access specifier did not mean that if they are the default they are also public.

So, here for example, it is declared as a public and this method is default access specifier, this means that this message is also public. That means this method can be accessible can be accessed by any other class in anywhere. So, this is the idea about the public access modifier.

(Refer Slide Time: 20:51)



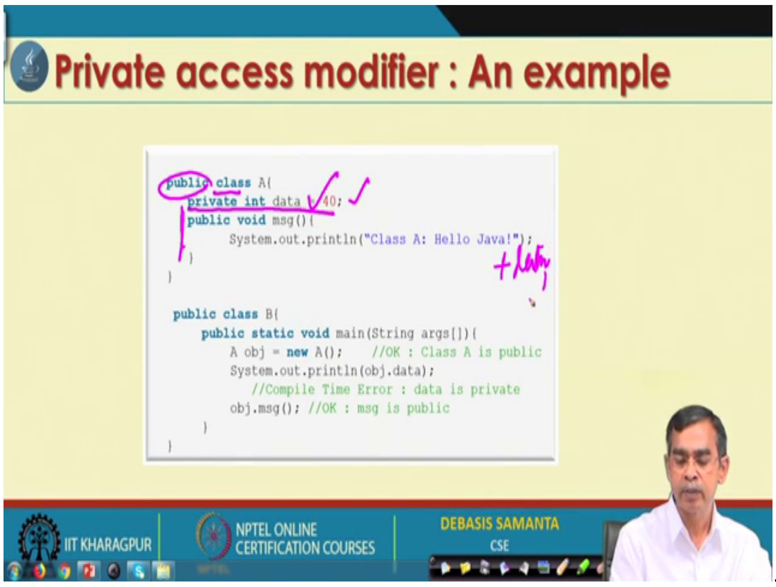
Private access modifier

The **private** access modifier is accessible only within the class.

PRIVATE
IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE

Now, let us come to the discussion of our next modifier, which is the strictest one the private access modifier. So, this modifier whenever you specify for a class, this means that any method which belongs to the same class can access it and outside that class no method can access that member. So, this is the concept of this private access modifier. Now let us have an example here.

(Refer Slide Time: 21:17)



Private access modifier : An example

```
public class A{
   private int data = 40;
   public void msg(){
        System.out.println("Class A: Hello Java!");
   }
}

 public class B{
     public static void main(String args[]){
        A obj = new A();     //OK : Class A is public
        System.out.println(obj.data);
           //Compile Time Error : data is private
        obj.msg(); //OK : msg is public
     }
}
```

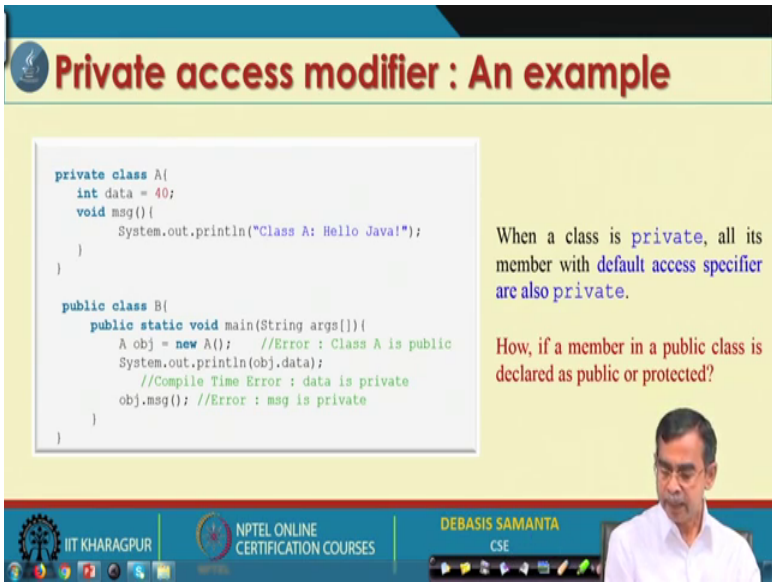
Now, we declare class A and access vision is public. So, public means it is a it can this class A can be accessed from anywhere; however, if we see this is the member, which is declared as a private that mean int data is declared as a private. So, being a private member. So, any method which belong to this class can use it. So, these method message if I write say this one plus data, if you write say plus data no issue. So, it can use it.

Now, let us consider the class B maybe in the same file, it if it is in the different file in same directory or sub directory it also equally applicable. Anyway so, this class B also declared as a public; that means, these class can be accessed from outside to any other class. Now here, if you see let us declare this class as a class B and this whole the program is saved in a one file called B dot Java. Now so here, as it is public class so we can create a object so because, public class is accessible so no problem. So, this is not an error and here you see system dot out dot print ln obj dot data.

Now here is a problem because, data is declared as a private and you want to access it from the class B. So, this will not allow you that mean you cannot access the data, you cannot read the value of data by creating an object of class A so there is an error. On the other hand as this message is declared as a private pubic because, this is a public and this is also public then this is also accessible that mean it will give this print message. So, you can understand that ok.

So, private means it belongs to if it is a private then only this one. So, outside this one it does not have any access that is the concept. Now so, this is a simple example now let us have consider another example.

(Refer Slide Time: 23:25)



Private access modifier : An example

```java
private class A{
   int data = 40;
   void msg(){
         System.out.println("Class A: Hello Java!");
   }
}

 public class B{
     public static void main(String args[]){
         A obj = new A();    //Error : Class A is public
         System.out.println(obj.data);
            //Compile Time Error : data is private
         obj.msg(); //Error : msg is private
     }
}
```

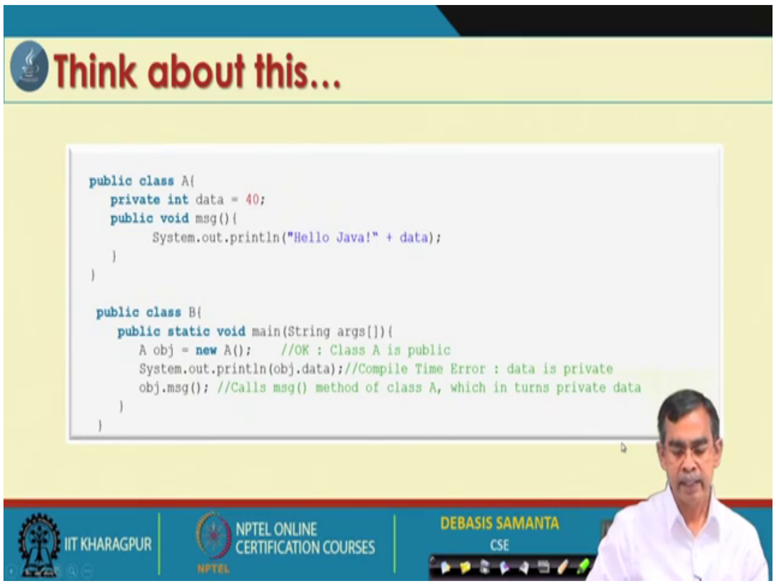
When a class is private, all its member with default access specifier are also private.

How, if a member in a public class is declared as public or protected?

In this example, what we have to do is that again, we declare this is a private. Now class is private, if class is declared as a private then all the member that is there with the default access specifier also become private. So, in this case as this is a private and there is a default. So, by default all these members are private.

Now, let us come to this program again here, class B the main class let us save this program as a B dot java, which include class A and class B file in the same file and here we create an A object this one, you cannot create because class A is declared private and for any private class no object can be declared outside. So, this is an error now here again, system dot out dot print ln object dot data, we are accessing the private data it is an error and object this is also an error. So, you will not be able to access any method here. So, you can understand that private, if it is declared here then it is not accessible outside it.

(Refer Slide Time: 24:43)



Think about this...

```
public class A{
   private int data = 40;
   public void msg(){
         System.out.println("Hello Java!" + data);
   }
}

 public class B{
    public static void main(String args[]){
       A obj = new A();    //OK : Class A is public
       System.out.println(obj.data);//Compile Time Error : data is private
       obj.msg(); //Calls msg() method of class A, which in turns private data
    }
 }
```

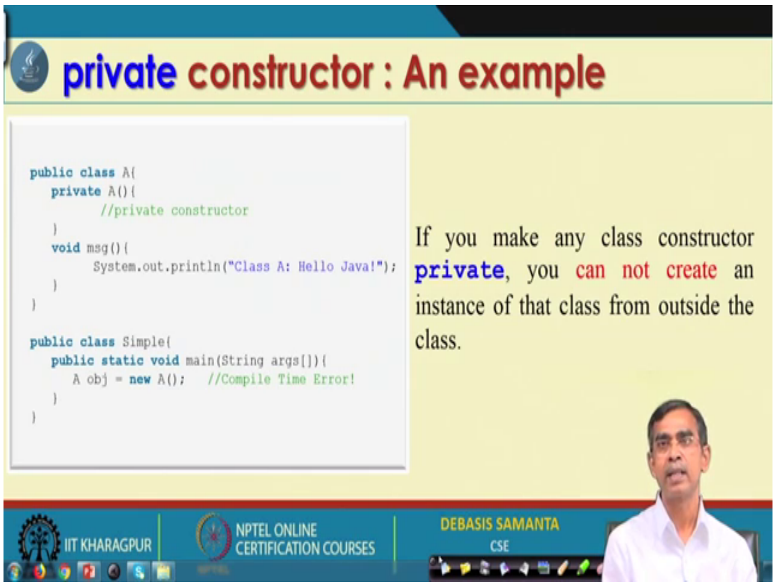
Now so, this is our example that we have discussed about the private access modification and this is the another example and this is a very interesting example and you should note it very carefully, what we have done it here. So, this class A is declared as a public this means we can create an object that is no issue. Now here, this method is public this means that we can call this method from here because, if the public method can be accessible from any other class from any other method belongs to any class whatever it is there.

Now, let us come to the discussion of this one. So, private in data so for so far this method is concerned this is no error no compilation error, because data is accessible to any method any private method, which is declared within the same class. Now let us come here, whether we can use this one, see here fallacy is that the data is a private and we access this private by means of this method. Now if we can you access this method then internally basically or indirectly I can say indirectly this data is accessible to this class B although data is a private but here is actually in Java you can do that.

So, if a method which is in the same class, where a private member is declared access that one and if that method; that means, this method is declared as a public and then public can be accessible. So, via this public all the private can be accessible. So, this is basically accessible. So, here basically we can access the private member in other way indirectly.

So here is a problem is that this method is the main critical. So, if you develop one class, where you can declare a method as a public this means that you allow any other programmer to access this. So, it is accessible whatever the data member, everything is there whatever id you can act, I will you are giving that accessibility to others so that now, if you do not want to give access then definitely you should declare this as a private; that means, this is strictly private, I do not want to give acts to any data or any method belongs to my class. So, this is the concept that is there so for the private the data is concerned.

(Refer Slide Time: 27:01)



private constructor : An example

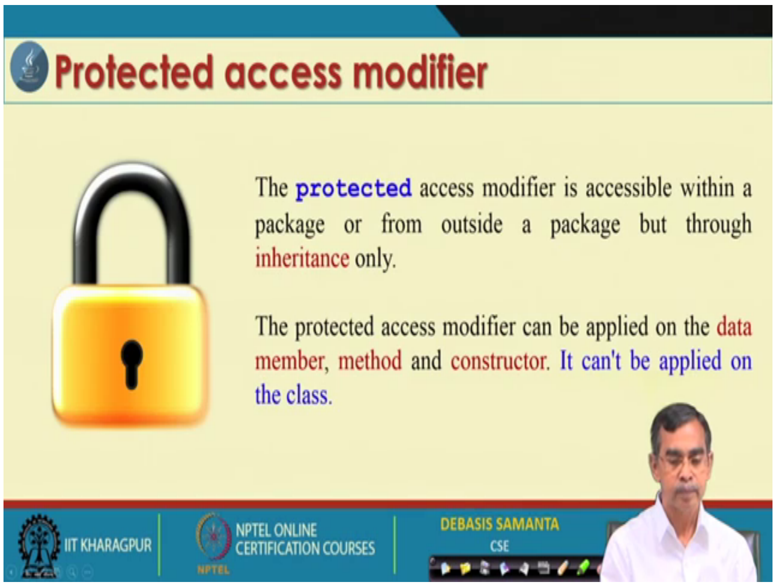
```
public class A{
   private A(){
         //private constructor
   }
   void msg(){
        System.out.println("Class A: Hello Java!");
   }
}

public class Simple{
   public static void main(String args[]){
      A obj = new A();   //Compile Time Error!
   }
}
```

If you make any class constructor **private**, you can not create an instance of that class from outside the class.

Now, let us another aspects so far the private data is concerned id is a constructor. Now what will happen, if we declare a constructor as a private? Now in this case, you see this class is declared as a public. So, anybody should can access this class does not create an object of this class; however, here you declare as a private constructor is declared as a private and then this measure by default is a public measure because, it is a public class belongs to public class.

Now here, if you create a constructor then means whenever, we create an object then constructor will be called here. Now when the constructor is called in this case as it is a private, private cannot be executed from anywhere right. So, this will lead to a compilation error; that means, if you declare a class as a public, but if constructor as a private then no one can create any object of that class. So, usually this is a very harder one rule, you should if you want if you want to make a class as public; so, better that you should not create any constructor, which is with private constructor. So, usually we do not do. So, constructor should be declared as a public like so that object can of that class can be created any way, but once you can create an object, it means that you can access all the members those are public or protected or access default it is there. Any way so, this is the idea about that private constructor is not advisable in actual situation actually.

(Refer Slide Time: 28:39)



Protected access modifier

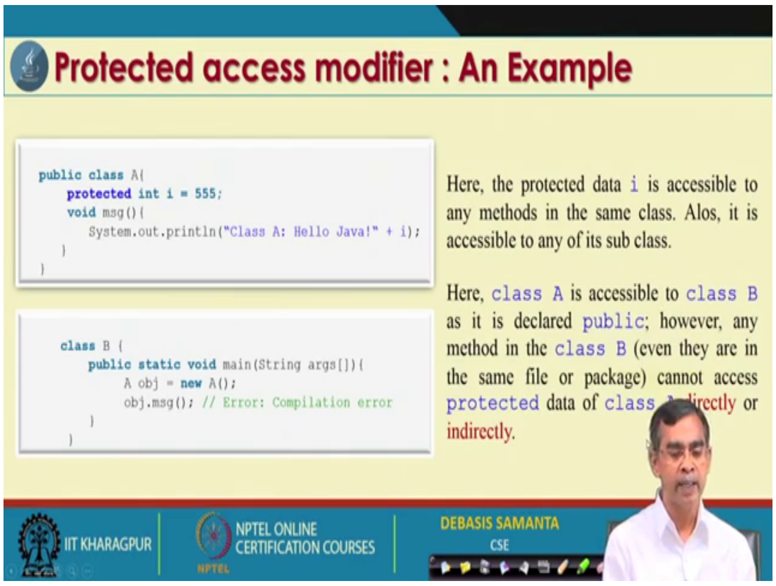
The **protected** access modifier is accessible within a package or from outside a package but through inheritance only.

The protected access modifier can be applied on the data member, method and constructor. It can't be applied on the class.

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
DEBASIS SAMANTA
CSE

Now, I am in the step of discussing the protected access in a modifier as I already told you that protected access modification is limited to the inherited class. That means, if a member is declared as a protected in it is super class then that member is accessible to any derived class or any subclass, which can be obtained from that.

(Refer Slide Time: 29:05)



Protected access modifier : An Example

```
public class A{
    protected int i = 555;
    void msg(){
        System.out.println("Class A: Hello Java!" + i);
    }
}
```

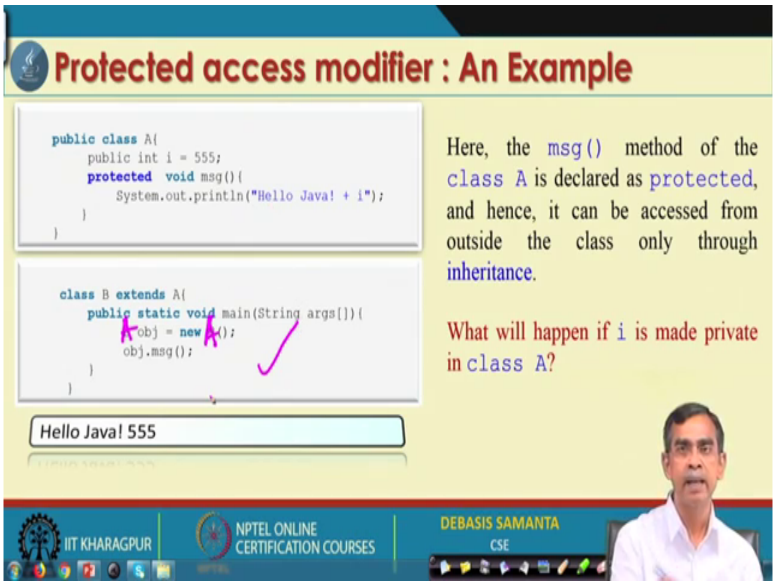
```
class B {
    public static void main(String args[]){
        A obj = new A();
        obj.msg(); // Error: Compilation error
    }
}
```

Here, the protected data i is accessible to any methods in the same class. Alos, it is accessible to any of its sub class.

Here, class A is accessible to class B as it is declared public; however, any method in the class B (even they are in the same file or package) cannot access protected data of class A directly or indirectly.

So, this is a concept of protected now here, let us have a quick look about it class A one class that we have declared here as a public and here one data we have declared as protected. Now the class B here, if we want to access this protected data then it will be a compilation error, you cannot access this protected error, but if they are in the same file in the same then this is there is no compilation error. So, in the same file it can allow, but in the different file outside or any class cannot access the protected data.

(Refer Slide Time: 29:39)



Protected access modifier : An Example

```
public class A{
    public int i = 555;
    protected  void msg(){
        System.out.println("Hello Java! + i");
    }
}
```

```
class B extends A{
    public static void void main(String args[]){
        A obj = new A();
        obj.msg();
    }
}
```

```
Hello Java! 555
```

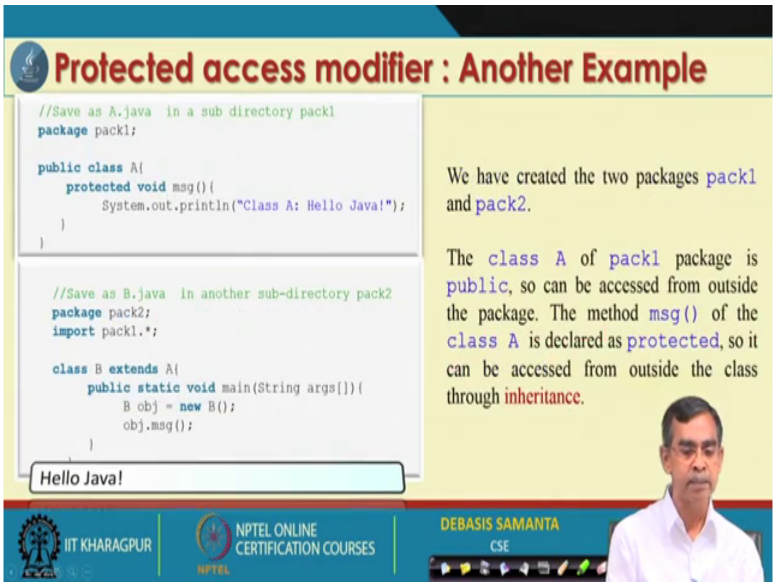
Here, the msg() method of the class A is declared as protected, and hence, it can be accessed from outside the class only through inheritance.

What will happen if i is made private in class A?

Now so this is a concept as a protected now here is another example, let us see whether it will work for you or not now class A and here is a public, this is a protected method and here you see class B extensive that mean B is a derived class subclass of super class A. So, by virtue of subclass, it can access any methods those are public and protected into this one. So, here that i which is declared as a public is accessible through this one. So, this is also correct and as it is a public and then B object new B so no problem B is a class itself. So, it is also create object.

Now, if I write say A object and A also that is also correct no problem because, it is a public a public object can be created. However, all the protected message cannot be accessed outside this one, but it if they are in the same, it can be accessible here now in this case. So, it is an extent. So, this is a valid compilation is successful as well as output is valid.

(Refer Slide Time: 30:45)



Protected access modifier : Another Example

```
//Save as A.java  in a sub directory pack1
package pack1;

public class A{
    protected void msg(){
        System.out.println("Class A: Hello Java!");
    }
}
```

```
//Save as B.java  in another sub-directory pack2
package pack2;
import pack1.*;

class B extends A{
    public static void main(String args[]){
        B obj = new B();
        obj.msg();
    }
```

```
Hello Java!
```

We have created the two packages pack1 and pack2.

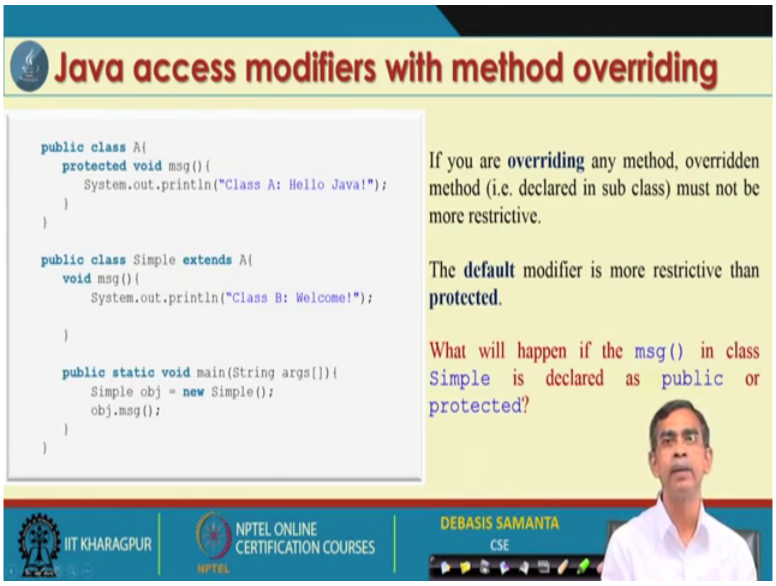
The class A of pack1 package is public, so can be accessed from outside the package. The method msg() of the class A is declared as protected, so it can be accessed from outside the class through inheritance.

IIT KHARAGPUR | NPTEL ONLINE CERTIFICATION COURSES | DEBASIS SAMANTA CSE

So, this is the protected access modifier and here is another. So, this is the same example little bit in a twisted manner, we create this class save in A dot java here another class save in a B dot java class a in a different directory and class B is in a different directory class B extends A. So, class B is the subclass of super class A. So, no problem all protected members for example, this message is accessible to this and so this is a valid output.

So, this is a concept of protected. So, very simple the protected means only sub class that inherits a super class can access it whether this is in the same file in thus different file, but in the same directory or the file that is there in outside any other directory no issue. So, this is the protected access modification.

(Refer Slide Time: 31:35)



Java access modifiers with method overriding

```
public class A{
   protected void msg(){
      System.out.println("Class A: Hello Java!");
   }
}

public class Simple extends A{
   void msg(){
       System.out.println("Class B: Welcome!");

   }

   public static void main(String args[]){
       Simple obj = new Simple();
       obj.msg();
   }
}
```

If you are **overriding** any method, overridden method (i.e. declared in sub class) must not be more restrictive.

The **default** modifier is more restrictive than **protected**.

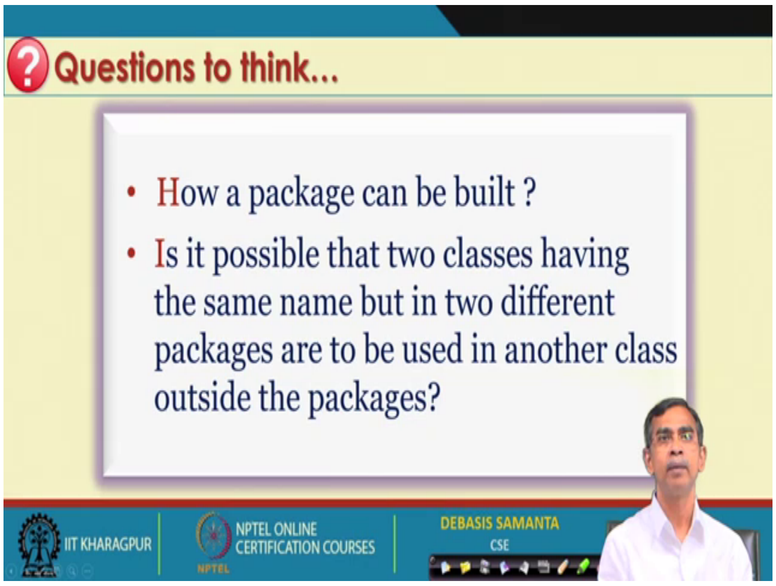
What will happen if the `msg()` in class `Simple` is declared as `public` or `protected`?

So, protected is basically a little bit limits the access specification. Now here again, this is a simple example that you can think about in this example, we declare class A and this is the simple one extends that is simple is a derived class of this super class and within this class the main method is declared. So, you can see if we do it then simple object, new simple if we create an object of this class and we call the object measure here. So, this will be accessible this one.

So, all these things are there. Now so, class a basically here again one thing you can notice that method overriding. So, here we see that this message is declared as a protected in class A and again the message is overridden here in the subclass this one. So, if it is a protected and if it is a subclass of that; so, overridden is possible, but if it is not supposed no access specification is there. So, it is a say default and then and then if you declare as a protected here is a protected then the problem.

The problem is that if it is a default one method, you want to protect it, override it as a protected one then it will not allow you to do that. So, this is stricter one since, that you cannot do that. So, all these things if the access vision not allowed, it will not successfully compile your program. So, during compilation time if any access specification is violated in your any program so far access is concerned it will be there if whatever the expression that you use it and if they are compiled it successfully; that means, you are allowing the program to give the access to all those things are there.

(Refer Slide Time: 33:17)



Questions to think...

- How a package can be built ?
- Is it possible that two classes having the same name but in two different packages are to be used in another class outside the packages?

IIT KHARAGPUR
NPTEL ONLINE CERTIFICATION COURSES
NPTEL
DEBASIS SAMANTA
CSE

So, we have discussed about the access specification in order to hide the information here information in the term of data as well as method, the whole class is also considered as an information. Now our next topic is basically that package, we have referred to many times package, package, package. Now we will discuss about in Java, how a package can be created.

So, this is a very important because as a programmer more specifically as a Java programmer, it is your responsibility to build very large software and whenever you have to be is very very complex the voluminous large software then definitely you have to manage all the classes very carefully. So, a package concept in Java gives you that kind of skill. So, that you can manage very large software configure your software very efficiently.

Now, if I ask you one question, what is your idea, is it possible that two classes having the same name, but in two different packages are to be used in another class outside the package? So, answer to these questions will be understood once, we discuss the concept of package in our next module ok.

Thank you very much.