

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

Lecture – 19
Demonstration – VIII

So, in Java package is very important concept. That we have mention explicitly in our last module that we have discussed on packages we cover this discussion of package in two modules. Now this Demonstration we will completes the understanding of the package concept in Java.

(Refer Slide Time: 00:40)

In today's demonstration...

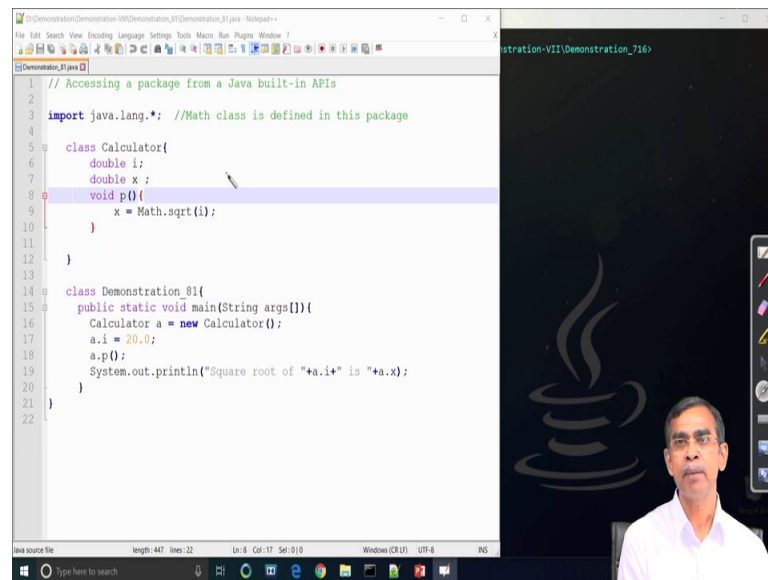
1. Importing a Java built-in API package.
2. Creating a user's own package.
3. Package with default access specifier for its classes.
4. Utilization of a package in a Java program.
5. Inheritance with a class in a package.
6. Access protection of classes in package.

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

So, in this today's demonstration we are going to explain how the built in API package can be imported in your Java program. So, importing a Java built in API packages. Then we will discussed about how we can create our own package and use this package in our own program and then package with access modification is really a big job. So, we will discuss about the packages with access protection.

And then packages which are there and the classes are there in this packages whether they can be used in our inheritance procedure. So, we will discuss about inheritance mechanism with some classes which are there in the package. And then finally, we will clear our concept about the different access specification that we have known in the context of package only.

(Refer Slide Time: 01:42)



So, let us have the first demonstration in this concept of package our first demo says that how we can access an API package. So, we know Java have 9 API packages all packages are readily accessible to a Java program to access a package we have already used one statement call the import. So, if we write import then the name of the package then the package will be imported into this program.

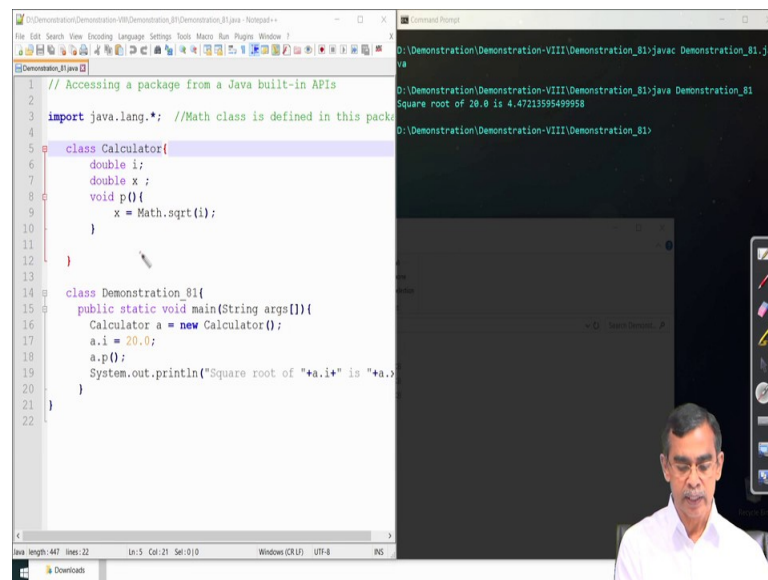
Here for example, import Java dot lang dot star java.lang.* this means that it will import all the classes which are declared in the lang package will be there will be accessible here. Here accessible means it is like this that in one the same file all the classes those are there in dot lang package will be pasted here like. So, it is the concept you can thing like that. So, by means of import it is basically this concept ok. So, you will be able to paste all the classes which are already define their in the long package they are the classes test it and compile successfully because, if we want to may store one class in our one package we should test it compile it then store it this is the concept actually there.

Anyway so, in this case as we see we import one class which belongs to lang package and actually this class as you see in the program the name of the class is Math.sqrt() Now here this is the simple class declaration calculator has the two members i, x and one method p; the p basically use the method Math.sqrt(); a square root is a method which is defined in the math class. Now, one thing also you can notice for the math class we do not have to create any object because this math is a static class which is declared in this

lang package as you know if a class is declared is a static or a method is declared is a static.

So, no object needs to be created see basically is a class method, is a class variable instants variable we have discussed similarly, the class method and instants method. So, here sqrt in fed is a class method. So, for a class method need not to call create any object here. Anyway so, math dot square root is basically pass an integer i if it is like this. So, it will a passed any value i then it will calculate the square root and then it return the result and store in the value x. So, this program here and running is successfully fine it is the obvious that this program will run it is there let us run this program 20.0 square root it will give you the result.

(Refer Slide Time: 04:31)



The screenshot displays a Java IDE on the left and a Command Prompt on the right. The IDE shows a file named 'Demonstration_81.java' with the following code:

```
1 // Accessing a package from a Java built-in APIs
2
3 import java.lang.*; //Math class is defined in this pack
4
5 class Calculator{
6     double i;
7     double x ;
8     void p(){
9         x = Math.sqrt(i);
10    }
11
12 }
13
14 class Demonstration_81{
15     public static void main(String args[]){
16         Calculator a = new Calculator();
17         a.i = 20.0;
18         a.p();
19         System.out.println("Square root of "+a.i+" is "+a.x);
20     }
21 }
22
```

The Command Prompt on the right shows the execution of the program:

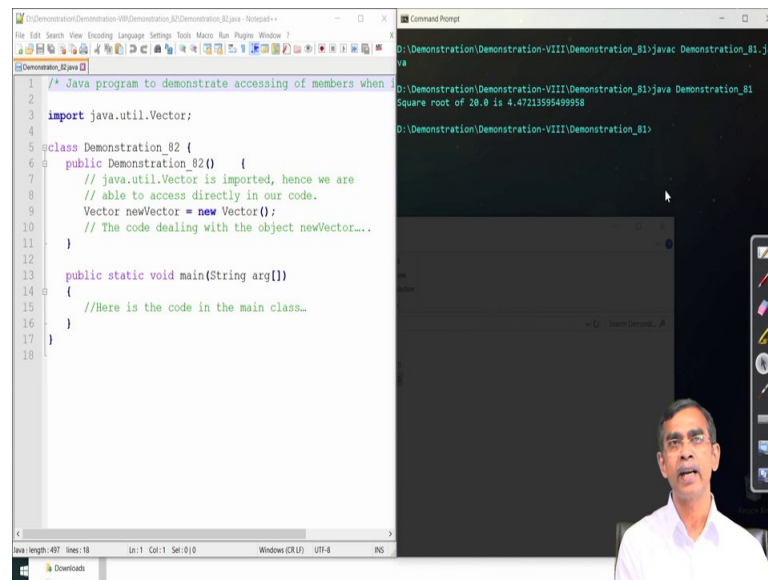
```
D:\Demonstration\Demonstration-VIII\Demonstration_81>javac Demonstration_81.java
D:\Demonstration\Demonstration-VIII\Demonstration_81>java Demonstration_81
Square root of 20.0 is 4.47213595499958
D:\Demonstration\Demonstration-VIII\Demonstration_81>
```

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

So, you can understand about it. So, these examples shows a very simple one so, that any class can be access there that class may be any method also belong to that class also readily accessible in your program. So, this demonstration is there. So, 20.0 is the value that has been passed to the Math.sqrt() giving the square root as 4.47 this result is there ok.

So, this example show that how we can access the package which is there like this the many packages all packages method in any packages is also can be access.

(Refer Slide Time: 05:15)



```
1  /* Java program to demonstrate accessing of members when
2
3  4  import java.util.Vector;
5
6  class Demonstration_82 {
7      public Demonstration_82() {
8          // java.util.Vector is imported, hence we are
9          // able to access directly in our code.
10         Vector newVector = new Vector();
11         // The code dealing with the object newVector....
12     }
13
14     public static void main(String arg[])
15     {
16         //Here is the code in the main class..
17     }
18 }
```

```
D:\Demonstration\Demonstration-VIII\Demonstration_81>javac Demonstration_81.java
D:\Demonstration\Demonstration-VIII\Demonstration_81>java Demonstration_81
Square root of 28.8 is 4.47213595499958
D:\Demonstration\Demonstration-VIII\Demonstration_81>
```

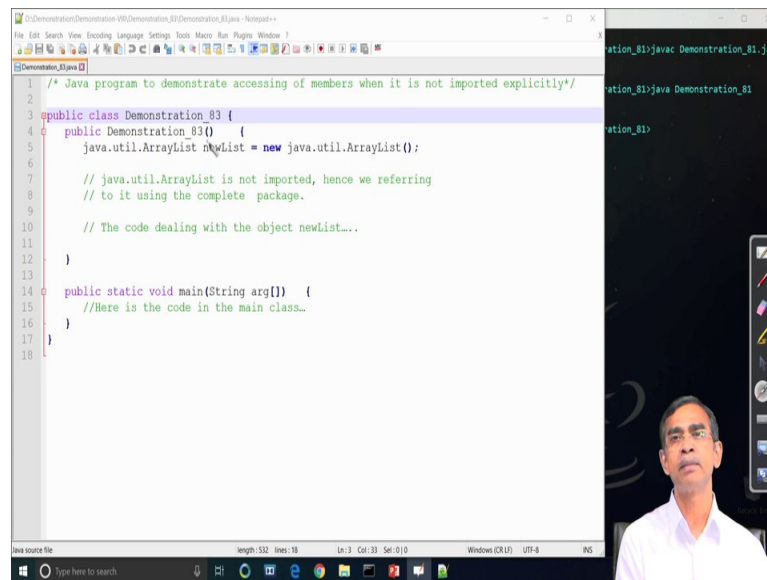
Now, let us have to another demo that we can explicitly mentioned here `java.lang.*`. That means, the entire package will be entire a package is very heavy very large; so not necessary to include the entire package. So, in lieu of the entire package we can explicitly mentioned a particular class if we know that it is required in our program.

Here these example includes this fact here you see `import java.util.Vector` you can understand what it does mean it is basically accessing one class the name of the class is vector and this class is define in the package util. So, in Java there is a package called util and there many class like vector array list array all this things are there it is there.

Now here, if you see. So, in this program in this class the demonstration _ 82 here we can see the vector class is imported once it is imported then this class can be used to create any object as we see `Vector newVector= new Vector()` . So, basically we create a create and object of class vector which is basically in util package and then this vector can be used for many other function that code is not here given.

So, this is a one example that a particular vector can be created. Now import is not necessary limited to only importing one class in our next example we can see that we can create without importing some method also.

(Refer Slide Time: 06:44)



```
1  /* Java program to demonstrate accessing of members when it is not imported explicitly*/
2
3  public class Demonstration_83 {
4      public Demonstration_83() {
5          java.util.ArrayList newList = new java.util.ArrayList();
6
7          // java.util.ArrayList is not imported, hence we referring
8          // to it using the complete package.
9
10         // The code dealing with the object newList....
11     }
12
13
14     public static void main(String arg[]) {
15         //Here is the code in the main class..
16     }
17 }
18
```

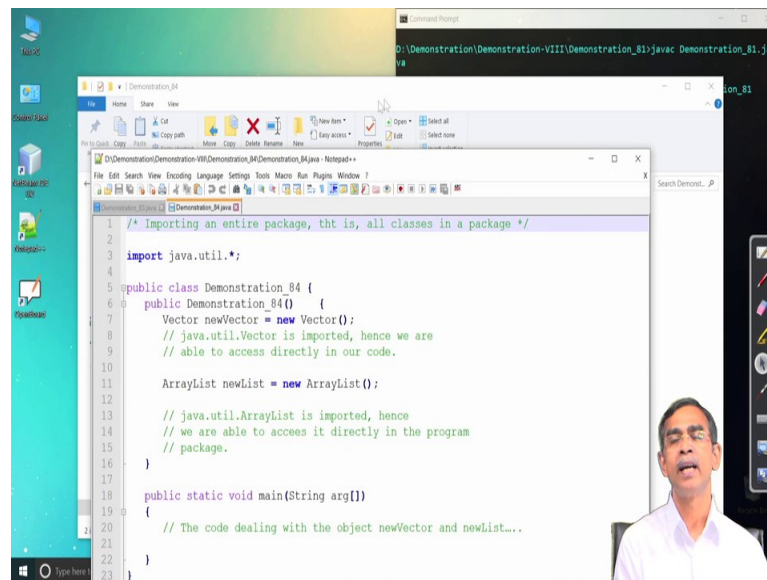
Terminal Output:

```
ation_81>javac Demonstration_81.java
ation_81>
```

For these things we have to explicitly mention the location of that class. So, without importing also we can have an accessed to some method which belongs to a package. So, this example includes this fact again. So, let us see the class here demonstration_ 83 where we can see here we have created one object the name of the object in newList. But this object of type ArrayList .ArrayList is already define in the class util and here you can see how we have done it explicitly mention by means of dot that java.util.ArrayList =new java.util.ArrayList () this means that we want to create an object which belongs to util the class type is array list.

So, this is also way here you can note that we did not use any import statement. So, these are the things are there, but the good practice is that import statement to be used in your program. If we use import; obviously, program will learn little bit slowly that you cannot understand, but if you do it explicitly this is first or of course, for this speed is not so important and it is a negligible the speed improvement if we say so. Now, our next example that we can already have a demonstration that we can instead only one package we can include many packages.

(Refer Slide Time: 08:01)

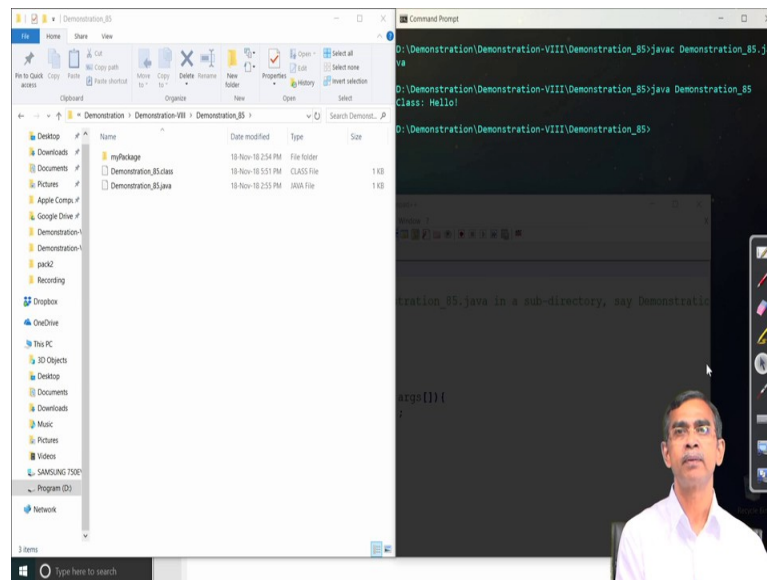


```
1  /* Importing an entire package, that is, all classes in a package */
2
3  import java.util.*;
4
5  public class Demonstration_84 {
6      public Demonstration_84() {
7          Vector newVector = new Vector();
8          // java.util.Vector is imported, hence we are
9          // able to access directly in our code.
10
11          ArrayList newList = new ArrayList();
12
13          // java.util.ArrayList is imported, hence
14          // we are able to access it directly in the program
15          // package.
16      }
17
18      public static void main(String arg[])
19      {
20          // The code dealing with the object newVector and newList...
21      }
22  }
```

For example Vector and ArrayList both of the classes if you want to include it. So, what we do we can do two things import java.util.Vector, import java.util.ArrayList like this. But instead of this `.*` is a short form if we do this all the classes belongs to these package will be accessible to this program here we can see vector new vector similarly array list new list we can access it.

Similarly, all other methods all other classes which is declare in this util can be access like this one. So, this is not a complete program quote related to dealing with Vector and dealing with ArrayList can be included here to make it complete. Our next extra demonstration is most important that is how a user can create his own package.

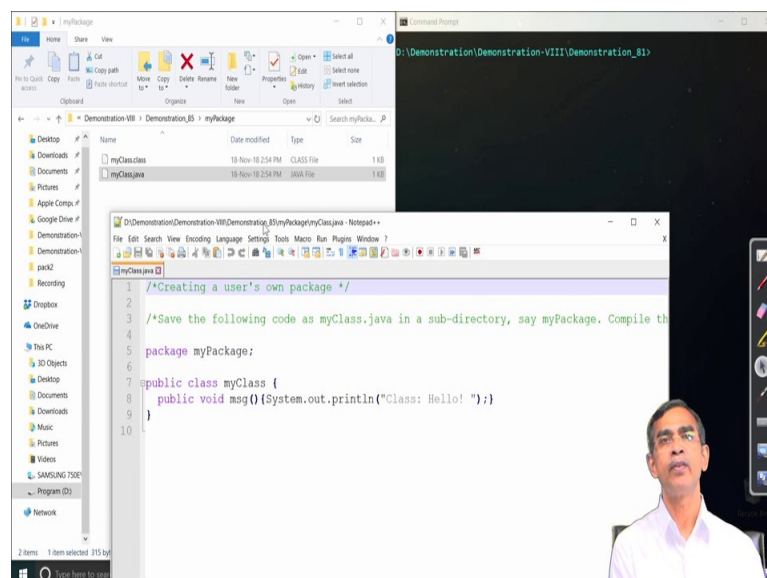
(Refer Slide Time: 08:57)



This is very important concept because, if we have to develop your own application software you have to build many packages of your own and maintain them this things. So, next discussion next demonstration onwards basically emphasize on this fact there how we can create our own package, how we can maintain them, how we can access them and how the different protection can be maintain in there all this facts.

Now, here you can see the simple program here where we want to maintain one class in a packages let us have the demo of first from in this line 8.5 open these one.

(Refer Slide Time: 09:31)



So, first now here you can see we have created one class name of the class is myClass and this class I want to put it into a package the name of the package is myPackage. So, for this thing we have to write the statement that package myPackage indicating that this class will be moved to the package myPackage. What is myPackage? A myPackage is a directory that means we have to build a directory the name of the directory should be myPackage and under this myPackage you should store this file the name of the file should be myClass that is the class name myClass.java.

As we see here we have created one directory myPackage here and then under this directory we have stored myClass dot Java it is not finished, we have to also compile this myClass and then store this myClass into that directories. So, we have created myClass dot class the compiled version of this Java file and store there.

And one more important thing is that if we want to create if we want to store one class in a package then that class should be declared as a public access specifier. So, here we can see we have given the access specifier as a public class myClass. So, these complete that one class is ready to be put into a package called the myPackage here. So, this class this class is now after showing computing.

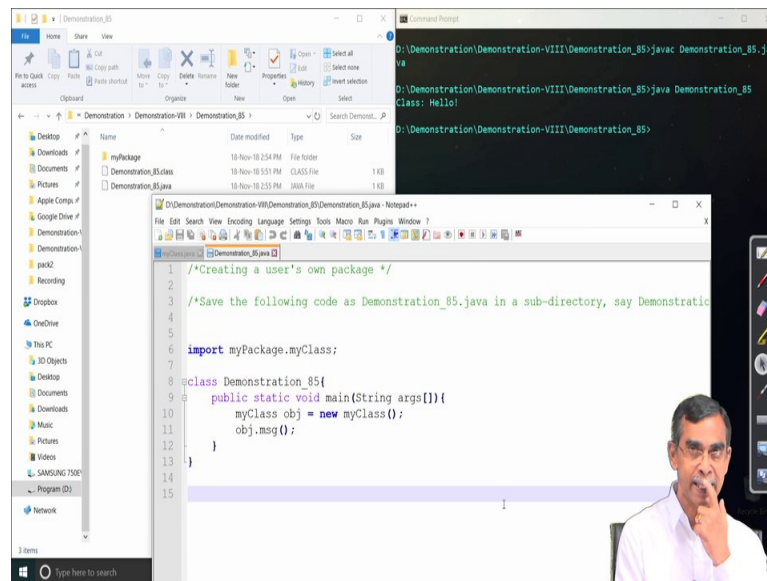
So, many steps are now one package is created I hope you have understood that how a package can be created. So, you have to create a file access specification should be public and then keep this file and compile this and store this in a subdirectory myPackage. And finally, put the package statement the package myPackage. So, this will complete to creating your own package.

Now, once this package is created then we can access this package in our own other program. So, this is an example showing little bit [FL] it is a [FL] fine. So, here we can see we are just attempting to access the package in order to access this one we have to use again input, it is just same as the user concept we have already learned, but in this case we want to access the package which belongs to the myPackage and the name of the myClass.

So, import myPackage.myClass and then this is the rest of the things accessing this myClass object create an object this one. So, this is complicated, but another important thing that you have to note it where this package should be stored.

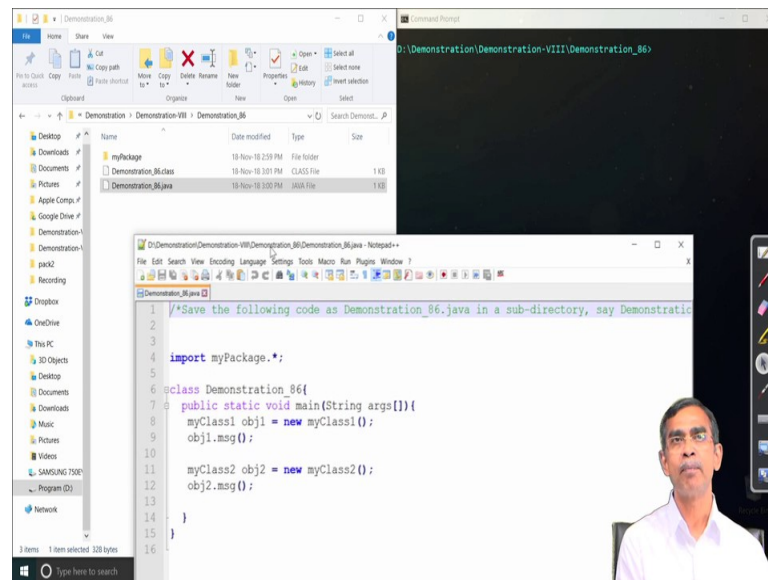
Now, this class the demonstration_85 if it is store in working directory let it be demonstration_85, then your package should be under this working directory that mean your myPackage is under the directory demonstration_85 then only you can access it. Now let us see whether this program that mean demonstration_85 is the main class is compilable and then executable. So, it is compile successfully and then running also successfully ok. So, this classes.

(Refer Slide Time: 12:48)



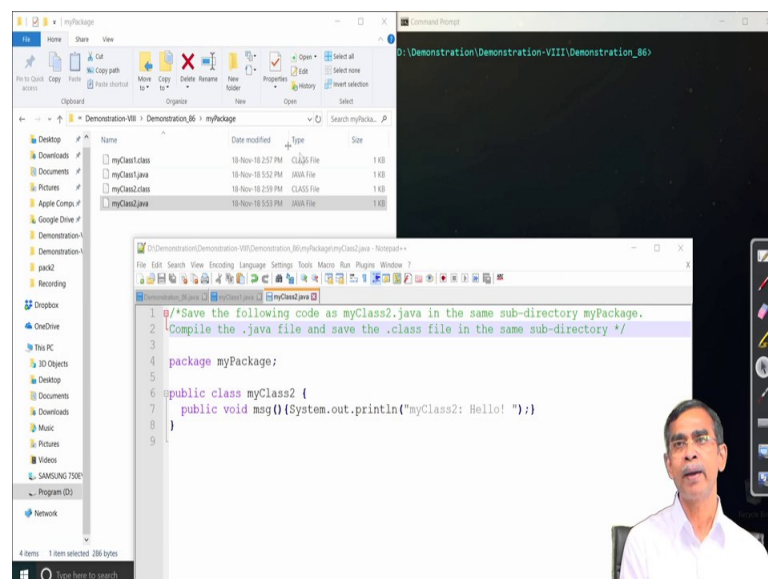
Now, we can see how a package can be created and a component which is belong to a particular package can be accessed in one file that is also this is basically creating a users own package here now our next demonstration. So, I have shown an example about that how we can create one we can place or put one package one class file into a package similar extension can be done if we want to have more than one file.

(Refer Slide Time: 13:22)



Same things can be repeated again like here in this demonstration let us see we create one package called the myPackage there. So, let us the myPackage directory here, now here we can see in this my package we have store two class file myClass 1, myClass 2.

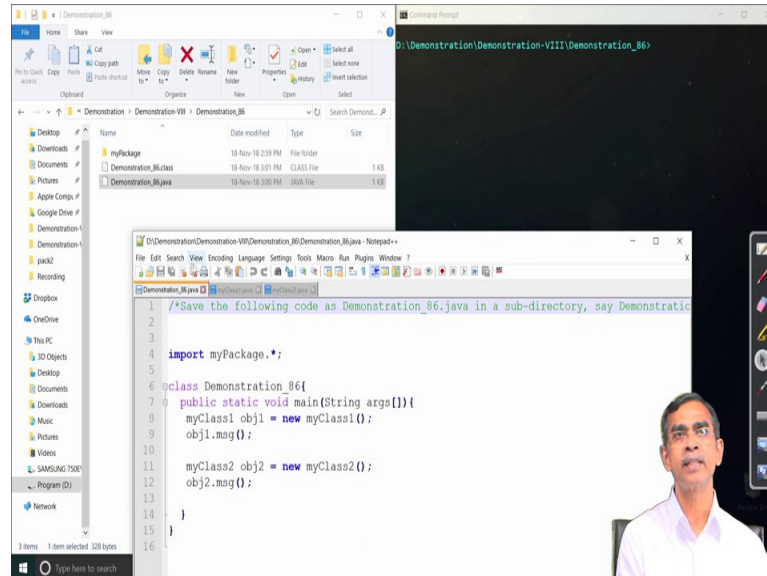
(Refer Slide Time: 13:35)



So, all these things again repeated just like myClass program there. So, myClass 1 here just package statement compile and then shape in the same directory. So, this completes the putting the myClass 1 class into the package similarly myClass 2 class into the same package there.

After creating this package with two files or more files we can access them into one class file that is class file should be a root above this myPackage.

(Refer Slide Time: 14:10)

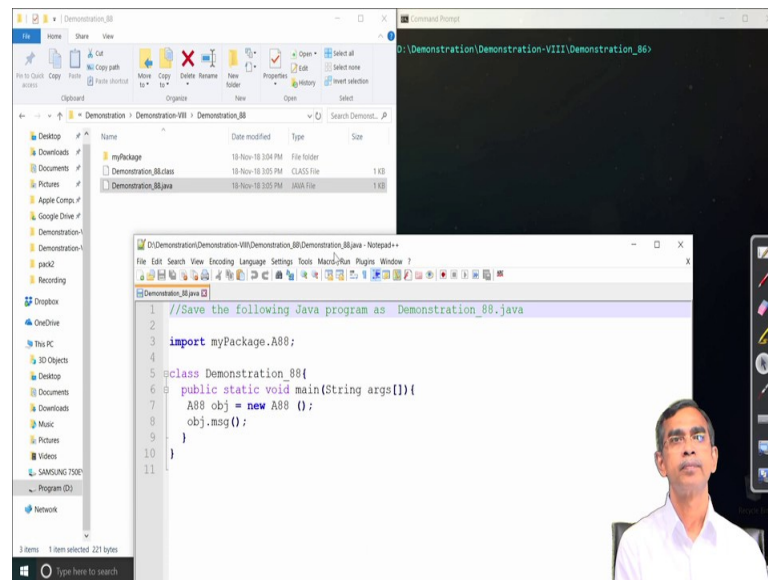


Now, here the working directory here is demonstration_85 under this is just above the myPackage directory. Now here we can see myClass 1 object is created myClass 2 object is created and this classes are imported from the myPackage packet. So, myPackage.* means it imports all the classes.

So, here again I repeat that myClass one as if pasted here myClass two class as if pasted here as if all the classes are here and all classes as you see that declare there in the package as a public access specification. So, this way two or many more classes can be put into package and package can be builder package can growth as the programmer will at many more component into the package. So, this concept is basically how a user can create the package here.

Now implicitly as we have already told that implicitly using import statement a package can be accessed just like user package also know a exception that a package also can be explicitly located and then there can be access in a program. Our next demonstration regarding this one let us go to the 8.8 demonstration here which basically explain everything here these program is basically here.

(Refer Slide Time: 15:32)



Here we can see a dot 88 (A88) is the one class file which is already there in myPackage that is all now let us go to the main program here which basically accessing this one. So, here so, let us go the demonstration yes.

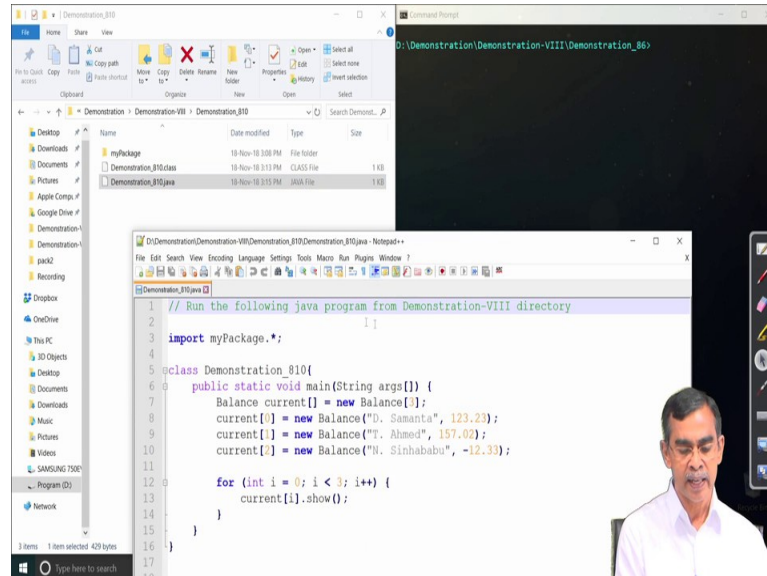
Now, here we can see after importing this A88 as if it is one class right. So, we can create an object obj for example, for the class A88; as if A88 is declared in this class. So, this is the explicitly A88 can be access as if the package the class it belongs to the programmers one class look like. So, package gives the access so, easily to the programmer.

Now, let us have the next demonstration 8.9 using the same concept as the explanation it is there, but using the method it is there. Now let us have the 8.9 demonstration it is little bit twist about the 8.8 extension. As if the 88 A88 class is already stored their which one method it is here and we want to access the same method here. So, that method will be accessible here ok. So, this is the A88 method and we are accessing this method by an application program class the main class here and then we can use it and using the fully qualified name here we can use myPackage.A88.

So, that mean fully qualified name because A88 is belongs to the mypackage here we do not have any import statement included here we have commented it without any import also explicitly we have given the similar demonstration earlier also. So, that explicitly we can mention the name or location of a particular class and by means of that we can

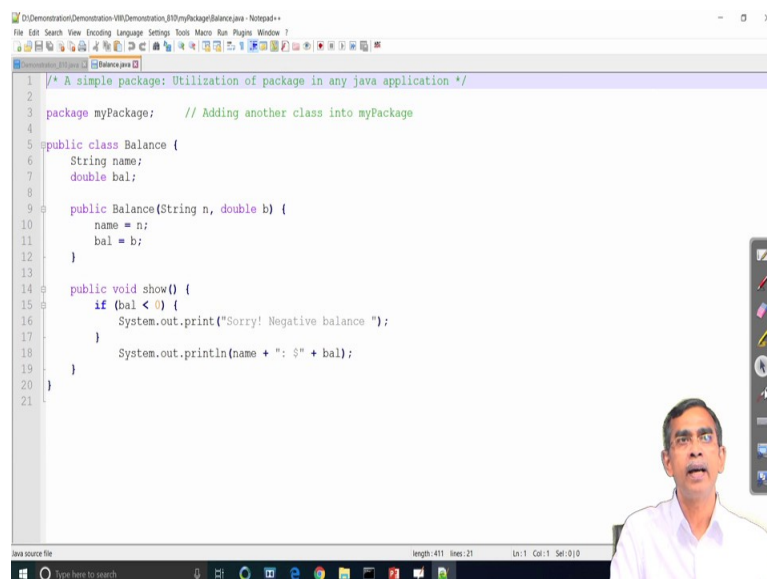
access the class and then object can be creator of that. So, this example; in fact, a concludes this one.

(Refer Slide Time: 17:44)



Now, let us have the another demonstration about it basically utilization of package in Java application there mean how we can use utilize this one now for this things let us consider one class first this is in the package let the name of the class we balance.

(Refer Slide Time: 17:55)



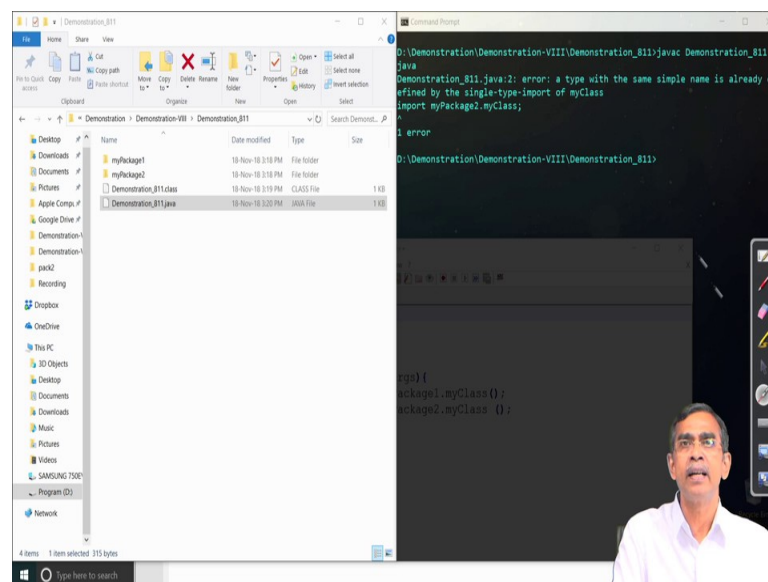
Let us go to this one we have created one package called the myPackage; under this myPackage there is a one class as you see name of the class is Balance dot Java and let

us have a quick look of the class this Balance.Java here. So, fine and we have declare this class belongs to a package myPackage and this class is very simple the Balance has the variable name as string and balance has a float.

And then it has one constructor the balance and then it is initializing the members and also it has one method the show method declare as a public and as you see the method all are method is public and then construct is also declare as a public and then this class is also declare as a public. If we do not declare as a public it will compilation error because non public any method cannot be access to other outside classes it is there because public is accessible ok.

So, this is basically the class lets have the main class that is your program main program. So, if this is the main program import this myPackage dot mean balance class is imported here after this imported here we can create objects as we see we can create a array of objects size of the array is three here. So, we have created initialized and then we can display the different elements in it for example, the show method we can call the show method for each object and it is.

(Refer Slide Time: 19:37)



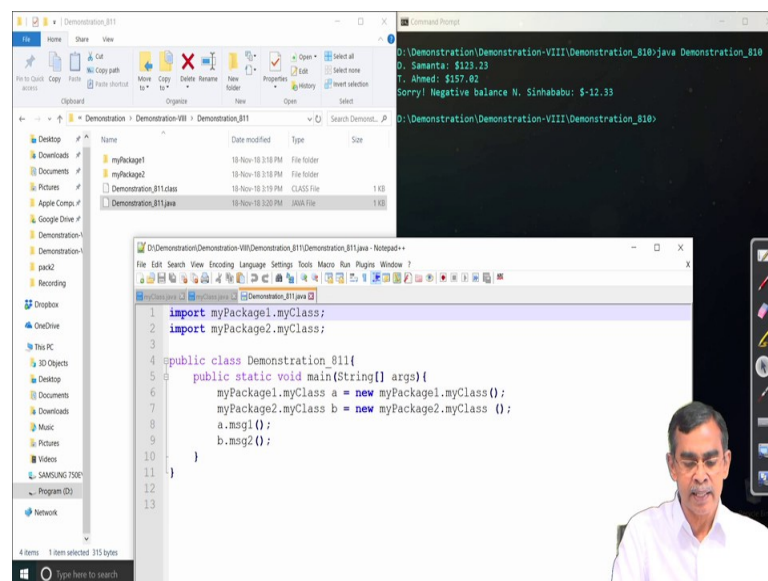
Now, let us have a quick run for this program. So, that it can the objects are created for this package class, and then the method belong to that package class is now accessible to the method. So, we can see that objects three objects in this case are created there initialized and their methods there members are accessible through there method. So, this

basically shows that how we can utilize the different methods, different classes are there in the package if there in your custodian ok.

Now, our next example is basically give the concept that whether same class name can be given to two different classes into belong to two different packages. Now, here let us consider one package say myPackage 1, now here you see under this demonstration 8.1 there are two packages myPackage 1 and then myPackage 2. Here we can say myPackage 1 and myPackage 2 are the two packages under this working directory and in myPackage 1 there is a class called myClass.Java same name, but may be structure different it is the same name with the method msg 2.

Now, let us go to the myClass a message myPackaged 1 method here also same name, but the different method; so two classes, but the two classes have the same name and there define in two different packages.

(Refer Slide Time: 21:20)



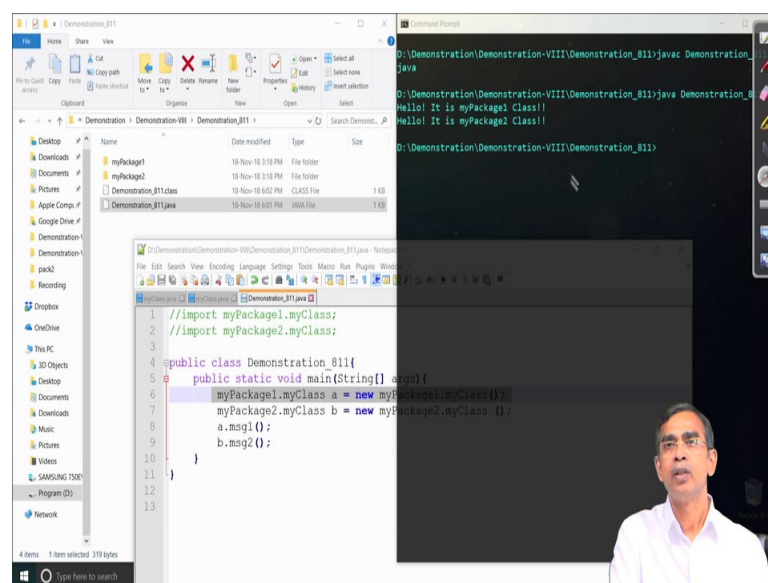
Now, let us come to our main class main class is here demonstration_811. Now here we can see I just want to access the two classes which belongs to the two different packages, but having the same name problem is here we can import it. Now I show import myPackage1 my dot class there now here is a problem. As I told you this import will paste or merge two class myClass.

Now, in one program you cannot give same name two different class. So, this way if we give this kind of import it basically leads to a compilation error if we run this things you will see it, will give an error because it will not take this kind of things because it is syntactically not valid. So, it is given an error that a type with the same simple name is already define by this.

So, what is the way out? Way out is here the remedy is that we should stop import here we cannot do that, but instead of import we can have the full qualified name I have given the full qualified name by say myPackage one dot myClass. So, it will result that myClass this myClass is belong to myPackage 1 and we can then successful to create an object a for that class.

Similarly, for another class myClass, but belongs to my package 2 we can create successfully able to create another object b here, now if we run it after saving.

(Refer Slide Time: 22:58)

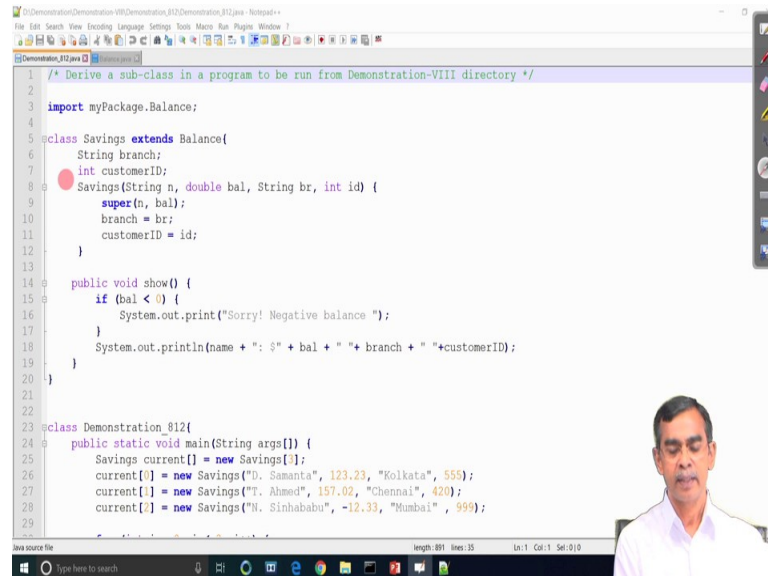


Let us compile and then we will see the `a.msg1()` ,`b.msg2()` is resolved although they belong to the different classes having the same name, but accessible here. So, the compilation is successful and then we run it also successfully this is the way out.

So, we can understand that the same class name can be used, but while you have to use it in your program we have to use the full qualified name to avoid the collision. So, this is

the one example that we have discussed about it now let us have another example about this example is basically discussing example 8.12 right.

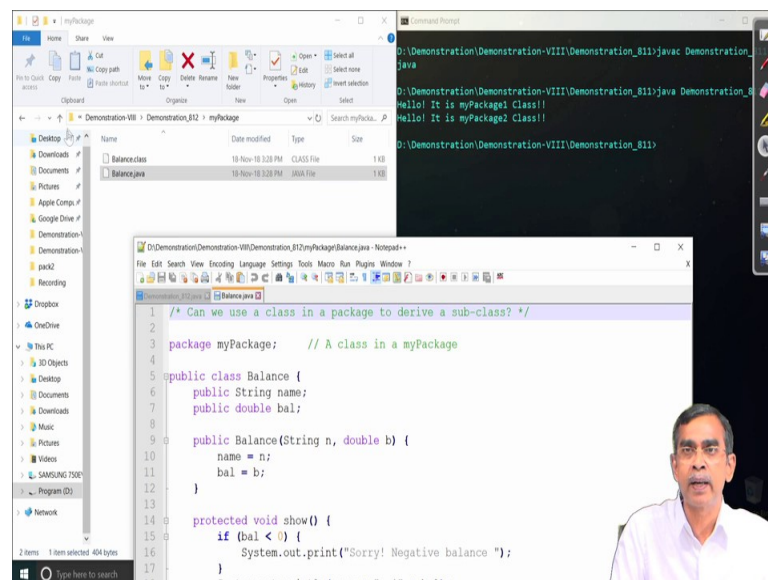
(Refer Slide Time: 23:36)



```
1  /* Derive a sub-class in a program to be run from Demonstration-VIII directory */
2
3  import myPackage.Balance;
4
5  class Savings extends Balance{
6      String branch;
7      int customerID;
8      Savings(String n, double bal, String br, int id) {
9          super(n, bal);
10         branch = br;
11         customerID = id;
12     }
13
14     public void show() {
15         if (bal < 0) {
16             System.out.print("Sorry! Negative balance ");
17         }
18         System.out.println(name + ": $" + bal + " " + branch + " " + customerID);
19     }
20 }
21
22
23 class Demonstration_812{
24     public static void main(String args[]) {
25         Savings current[] = new Savings[3];
26         current[0] = new Savings("D. Samanta", 123.23, "Kolkata", 555);
27         current[1] = new Savings("T. Ahmed", 157.02, "Chennai", 420);
28         current[2] = new Savings("N. Sinhababu", -12.33, "Mumbai", 999);
29     }
30 }
```

We can whether a class which is defined in a package can be used to inherit I mean can be used as a super class to derive some sub class. So, this demonstration we will clear this doubt here.

(Refer Slide Time: 23:42)



```
1  /* Can we use a class in a package to derive a sub-class? */
2
3  package myPackage; // A class in a myPackage
4
5  public class Balance {
6      public String name;
7      public double bal;
8
9      public Balance(String n, double b) {
10         name = n;
11         bal = b;
12     }
13
14     protected void show() {
15         if (bal < 0) {
16             System.out.print("Sorry! Negative balance ");
17         }
18         System.out.println(name + " : $" + bal);
19     }
20 }
```

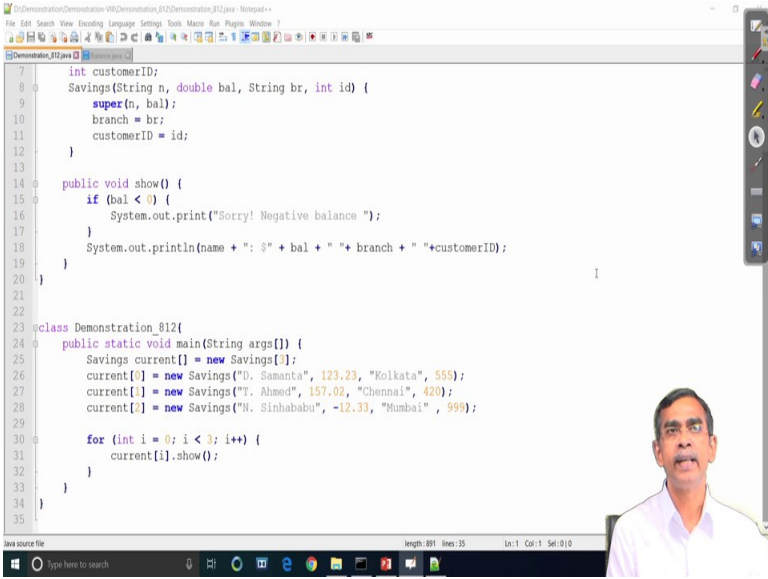
Now, here let us consider a myPackage package which includes one class belong to this myPackage the name is Balance class. So, a class is already there in myPackage

directory the name of the class is Balance. Now our object is that we want to derive some classes from this class Balance as a super class. So, let us have the main class where we can try to inherit this one. So, let us go to the main method. So, this is at demonstration_812 is the main class here. Now in this main class you can see we derive a sub class the name of the sub class is savings; Savings it basically inherited from the super class balance.

Now, whatever the member elements are there in the balance namely string name then float balance all this things are readily available here, because of the public it is there, because this class as made public and all methods. And then member are also public there otherwise we cannot access through package anyway.

So, Savings and in the Savings class we can declare one member as a branch and another integer as customer id. So, two fields are declared here in addition to this the constructor is also declare here to initialize the object of type savings and then a method. So, definitely this method is basically overriding method, because the show method that is there to display only two elements, but here we have to in addition to earlier two balance and name we have to display a branch. So, show is basically also overridden method. So, this completes the inheritance.

(Refer Slide Time: 25:29)

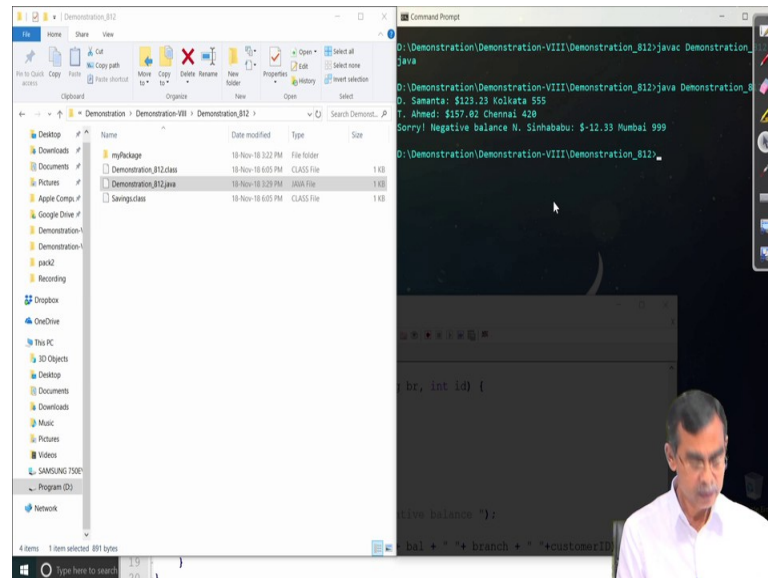


```
7      int customerID;
8      Savings(String n, double bal, String br, int id) {
9          super(n, bal);
10         branch = br;
11         customerID = id;
12     }
13
14     public void show() {
15         if (bal < 0) {
16             System.out.print("Sorry! Negative balance ");
17         }
18         System.out.println(name + ": $" + bal + " " + branch + " " + customerID);
19     }
20 }
21
22
23 class Demonstration_812 {
24     public static void main(String args[]) {
25         Savings current[] = new Savings[3];
26         current[0] = new Savings("D. Samanta", 123.23, "Kolkata", 555);
27         current[1] = new Savings("T. Ahmed", 157.02, "Chennai", 420);
28         current[2] = new Savings("N. Sinhababu", -12.33, "Mumbai", 999);
29
30         for (int i = 0; i < 3; i++) {
31             current[i].show();
32         }
33     }
34 }
35 }
```

Now, we could inherit it now we have to do it at whether we successfully inherit it or not. So, to understand these things our task is to create three objects of type inherited

class. Now in this see the current is an array of objects of type savings and we have created three objects in this case. So, three objects are created and initialize subsequently those are stored are current 0, current 1, current 2 like this. And now for(int i=0;i<3;i++) for int i currents to 0, i less than 3, i plus plus we are trying to run this method so, for the savings class object displaying the elements.

(Refer Slide Time: 26:12)



So, this completes the complete class. Now, let us have the compilation and subsequently execution then we can see that a method a class which are there in a package can be used to inherit into some application package application class. So, here we can see 81 fine. So now, we can successfully compile and execute that inheritance is possible. So, what I want to emphasize it again that a class if it is there in a package.

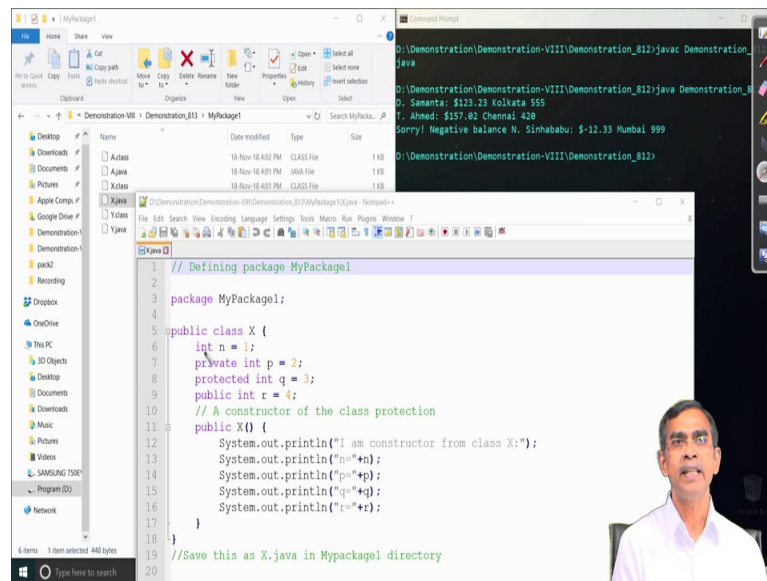
So, inheritance is not an issue, and if it is public all this things; so all members will be accessible if it is private they are then obviously private member cannot be accessible. But those are the protected those protected member are accessible in the inherited class, those are the obvious concept we have already learn. So, I do not want to explain it again.

Now, let us have the next demo. So, our next demo is basically that if the same class name, but in two different packages. And the class are then referred in a Java program that we have discussed already right not is there 8.12 we have declared that is 8 point our

next demonstration is basically combining all access modification in one file and how it works.

Now, this program is bit complex and little bit large actually. Now here let us consider two packages myPackage 1 and myPackage 2; now myPackage 1 composition is that it has three classes; one class A class X and class Y and Y is a derive class that mean Y is an inherited class from X. So, the combination is like this.

(Refer Slide Time: 28:04)



The screenshot shows a Java IDE with a file explorer on the left, a code editor in the center, and a command prompt on the right. The file explorer shows a directory structure with 'MyPackage1' and 'MyPackage2'. The code editor shows the following code:

```
1 // Defining package MyPackage1
2
3 package MyPackage1;
4
5 public class X {
6     int n = 1;
7     private int p = 2;
8     protected int q = 3;
9     public int r = 4;
10    // A constructor of the class protection
11    public X() {
12        System.out.println("I am constructor from class X:");
13        System.out.println("n="+n);
14        System.out.println("p="+p);
15        System.out.println("q="+q);
16        System.out.println("r="+r);
17    }
18 }
19 //Save this as X.java in MyPackage1 directory
20
```

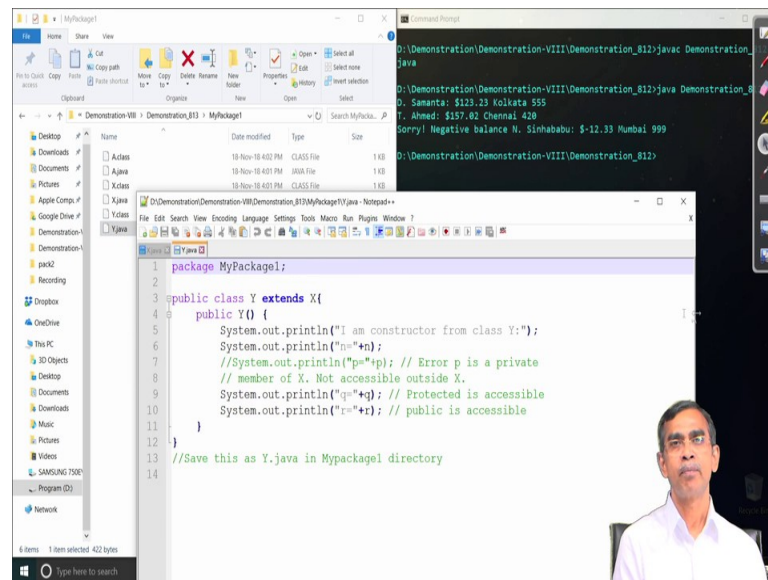
The command prompt shows the following output:

```
D:\Demonstration\Demonstration-VIII\Demonstration_812>javac Demonstration_812.java
D:\Demonstration\Demonstration-VIII\Demonstration_812>java Demonstration_812
D. Samanta: $123.23 Kolkata 555
D. Ahmed: $157.02 Chennai 420
Sorry! Negative balance N. Sinhababu: $-12.33 Mumbai 999
D:\Demonstration\Demonstration-VIII\Demonstration_812>
```

Now, we will see the three different classes there in this package one by one. So, let us first have the class X declaration. So now, this is the class X as we see in this class X n is declared as a default axis specification p is a private, q is protected and r is public and so no problem.

So, constructor is made as public the class itself X is made as public. So, that we can use it in other classes, other application through package concept and then these are the simple concept that we can print it in this case all members are accessible because they are belong to the same. So, whether is a public private protected it is accessible within the same class. So, this class X is successfully compliable and it will executes no error. Now, let us go to the class X after the class is declaration is over class Y as I told you here class Y is basically derived class from the X.

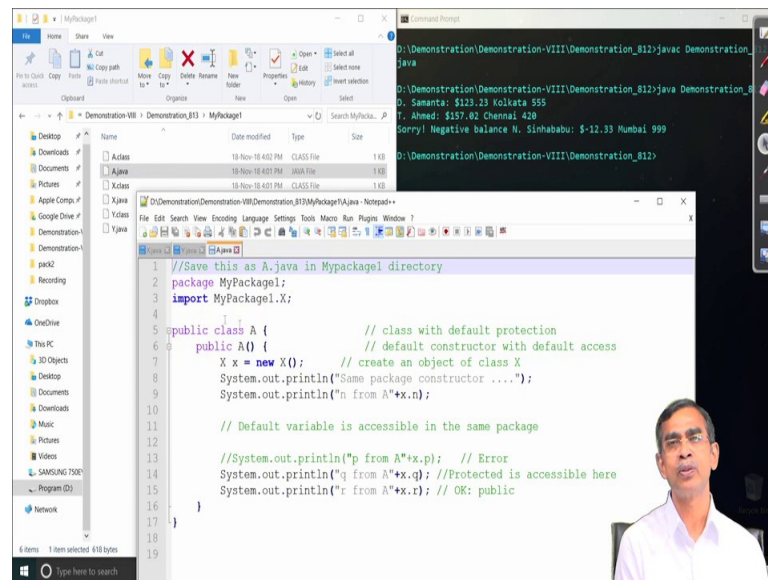
(Refer Slide Time: 28:59)



So, we made the class Y is a public and then extends X and constructor also a made it public Y and then we create the things when here now we have put some comments there because, if we remove the comment then these class will not be executable. Here we can see that System.out.println attempting to print the n which is the default value there. So, default is accessible to any class because it is derived one. So, default is accessible to the derived so, it is not an error there.

Now, let us come to the p; p is declared as a private there in class X. So, now, if we want to access it in the derive class. So, private is not accessible so, this will give an error. Now member X is not access now System.out.println q here q which is the protected by virtue of protected because it is an inherited class so, q is accessible. So, this is ok. So, protect is accessible and r being a public so, public is accessible. So, after commenting so, this program will run successfully; that means the access is fine.

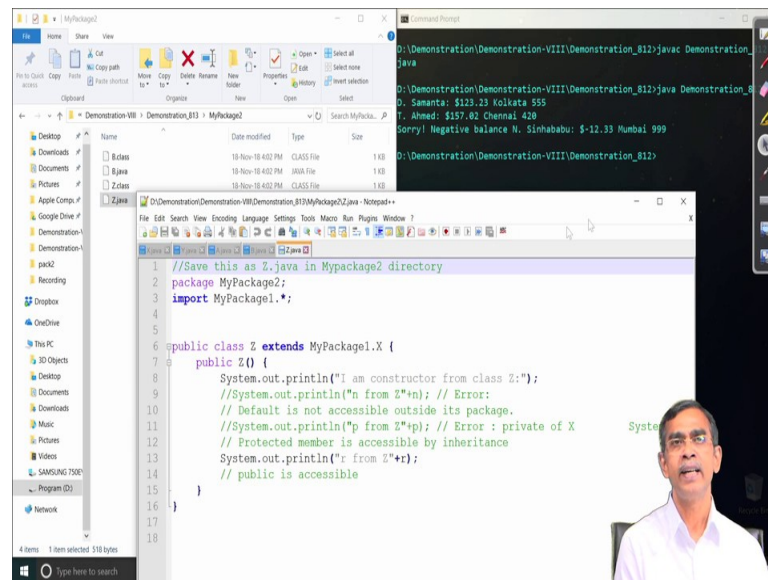
(Refer Slide Time: 30:22)



Now, let us come to the class A here the class A if we want to print the different members which is there in class X whether this is accessible or not. So, we import the myPackage dot X, in this class A because it to access it now public A we can declared a constructor class A and create and object X of type X. And through X we can see here the default is accessible two because it is in the same file same folder same directory. So, no issue A dot X x dot n is accessible.

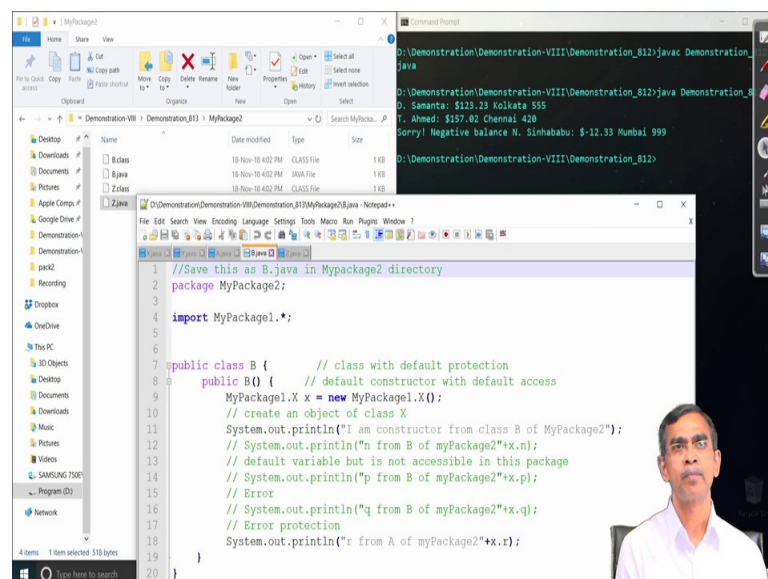
Now come to the x dot p it is not accessible via A because p is private. So, it is an error; however, protector is also not accessible because Ax is not an inherited class here it is X class is there through X the protected class cannot be accessible public is accessible. So, public is there so, this completes the working of the different classes in the myPackage 1. Now, similarly go to the myPackage 2 it is little bit different, but in the different sense that myPackage 2 is has two different classes B class B and class Z.

(Refer Slide Time: 31:26)



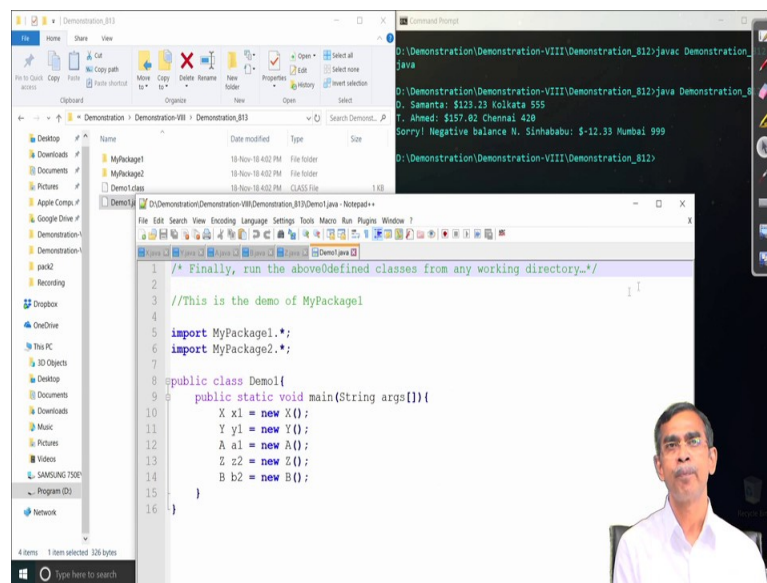
Class Z is an inherited of class X. So, myPackage 1 inherited like this one. So, if we can inherited this concept using myPackage 1 so, import myPackage 1 extends this one. Now, so, for this Z is concerned; here we can see as it is in the different file, because this myPackage two in a different directive can say. So, the default axis is not accessible so, it will give an error. So, dot n is giving an error and then the protected is also not accessible private is not accessible; however, public is accessible. So, in under this only the r is accessible others are not accessible.

(Refer Slide Time: 32:07)



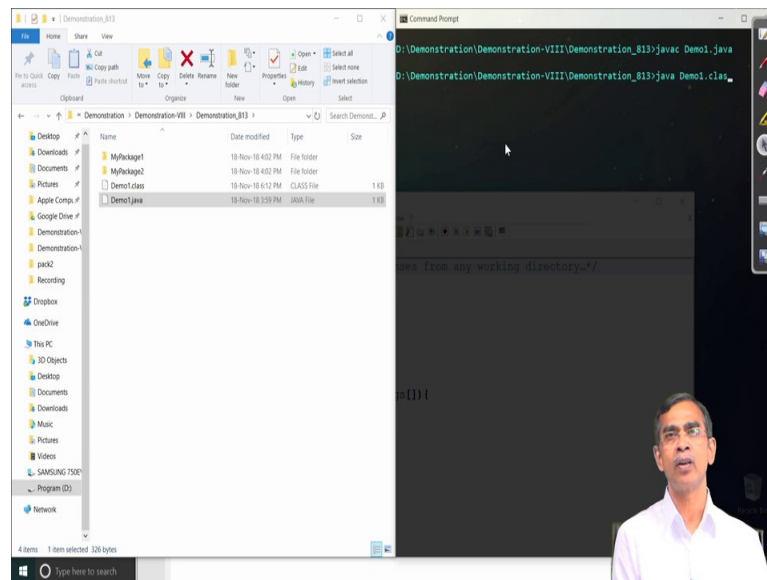
Now, let us go to the B class B object; class B object is same here we can see default is not accessible, private is not accessible, protector is not accessible, only public is accessible. So, these will give the idea about the different access specification. So, for package is concern, now once after creating this package we can use this package if they are successful in the package form then they can be accessible here the demo program here.

(Refer Slide Time: 32:32)



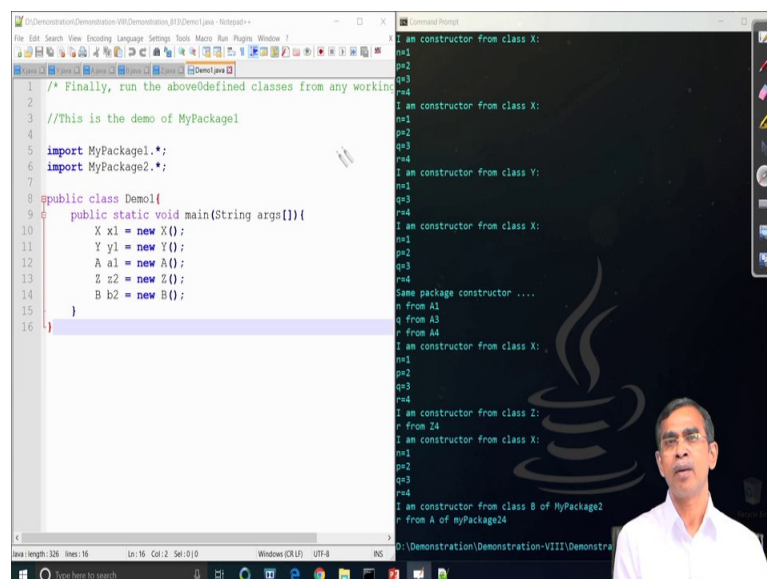
So, in this demo we are trying to access the different classes those are there in different packages and we can create the objects here the objects are readily available because, the classes are successfully compile, and then and legitimately accessed here. So, this completes the access specification so, for the package is concern.

(Refer Slide Time: 32:56)



So, there is no different issue; so, so for the access specification is concern other than information hiding concept. If we have understood the information hiding concept clearly then you will be able to understand about the concept those are there in package.

(Refer Slide Time: 33:05)



So, package is basically is a more advance version of the information hiding there is the information that is there ok. So, this is the complete discussion about the package; obviously, package can be matured the learning about this package concept can be matured. Once you are involved in the developing of large software, but this is the basic

the beginning of this package concept and that is fine. So, this is ok. So, this is the program I advise you to go for further rigorous practice with the demo program that we have used here in this program. So, that you can completely you can get more confidence about this programming. Thank you.

Thank you very much.