Lesson 04 Demo 05

Creating Packages and Access Modifiers

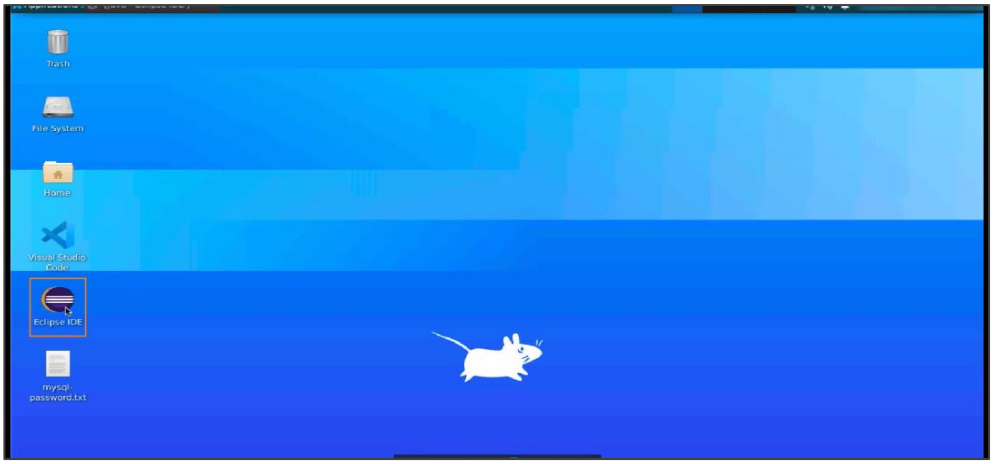
Objective - To implement packages and access modifiers in Java **Tools required** - Eclipse IDE **Prerequisite** - None

Steps to be followed:

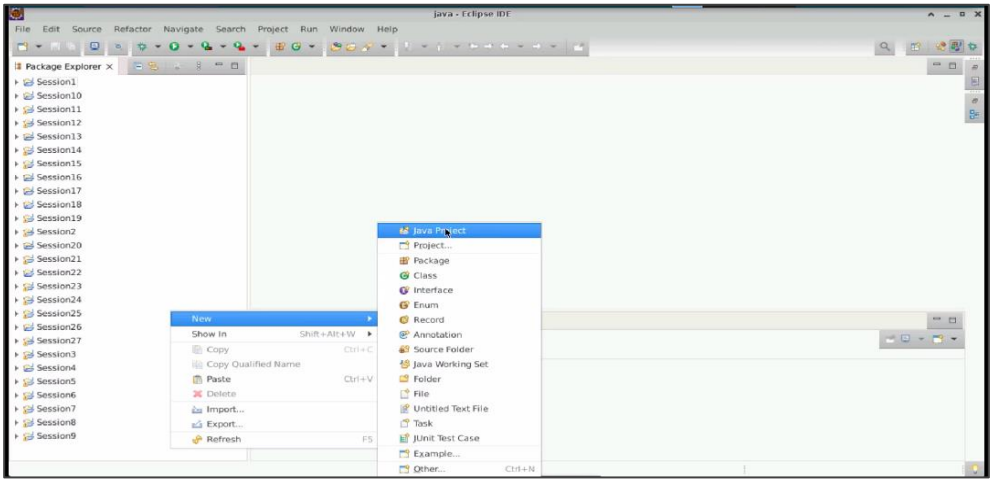
1. Create java package and class
2. Implement the use of access modifiers in class declaration
3. Implement the use of access modifiers inside the class
4. Implement the use of access modifiers outside the class
5. Implement the difference between protected and default access modifier

Step 1: Create java package and class

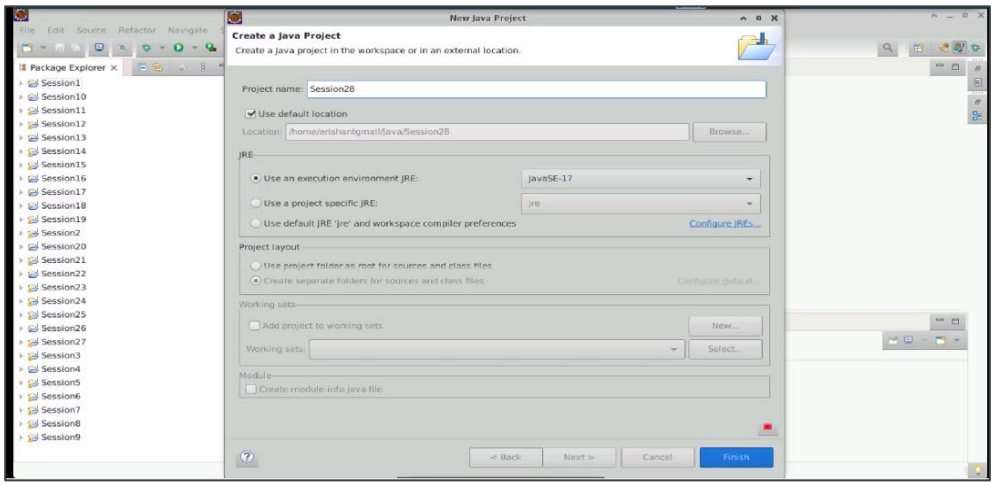
1.1 Open the **Eclipse IDE.**



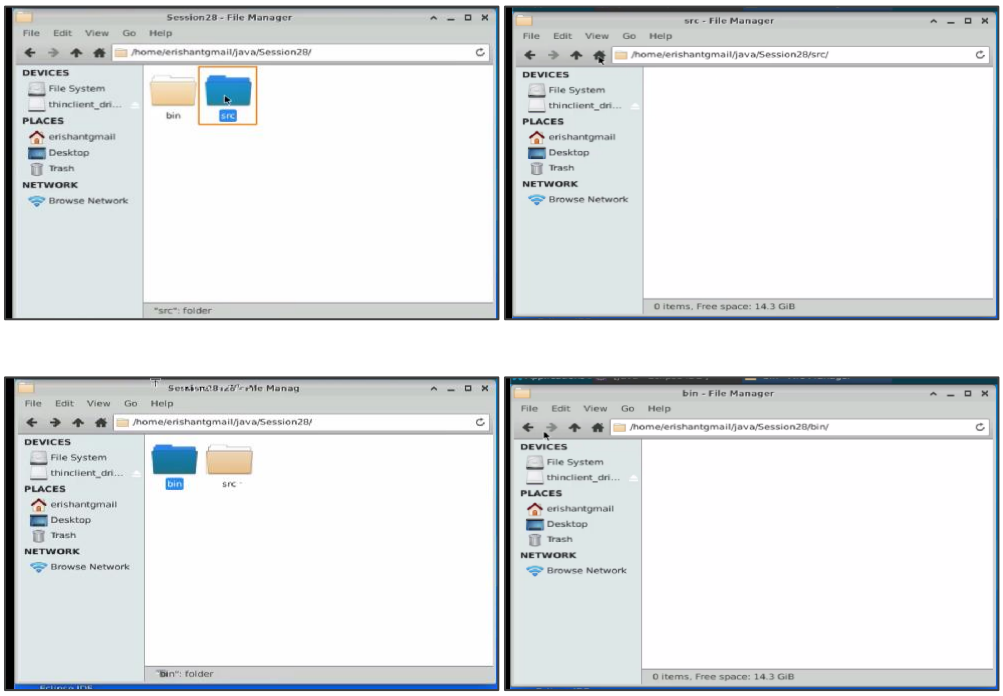
1.2 Select **File,** then **New**, and then **Java Project.**



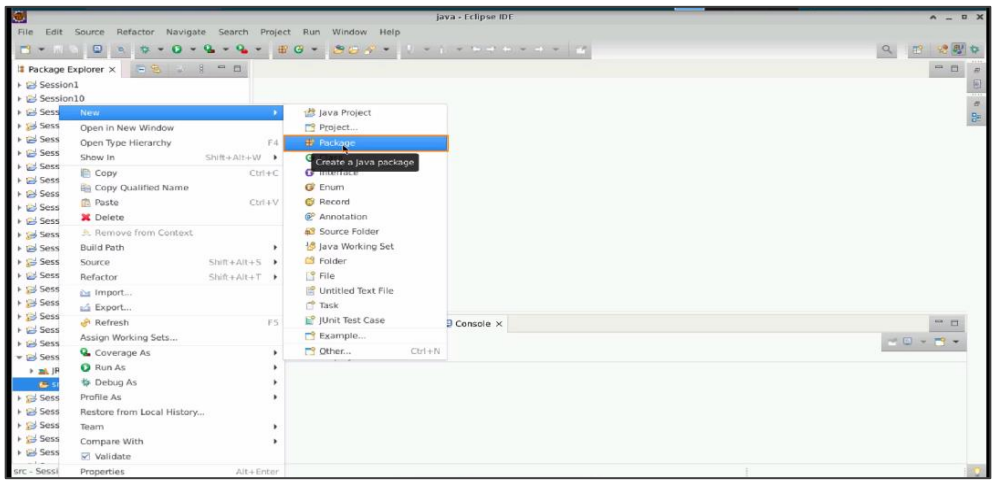
1.3 Name the project as "**Session28"** and select **Finish**.



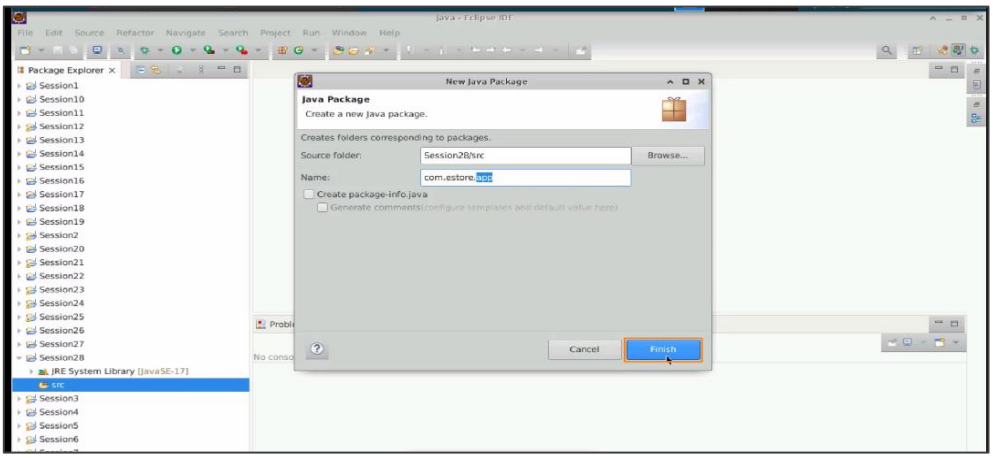
1.4 In your **File Manager,** click on the java workspace and open **Session28** folder. Two folders are placed inside, **a source folder** and **a bin folder**. As of now, both the folders are empty.



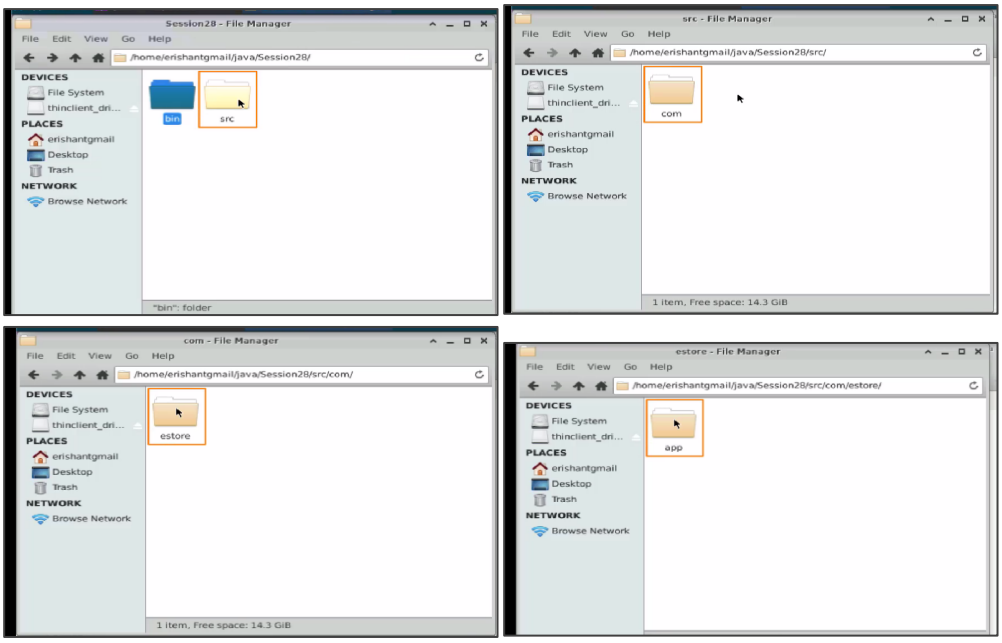
1.5 In the left panel, inside the project **Session28**, right click on the **src** folder, select **New,** and then **Package** in order to create a java package. A java package is a directory in which the source files are placed.



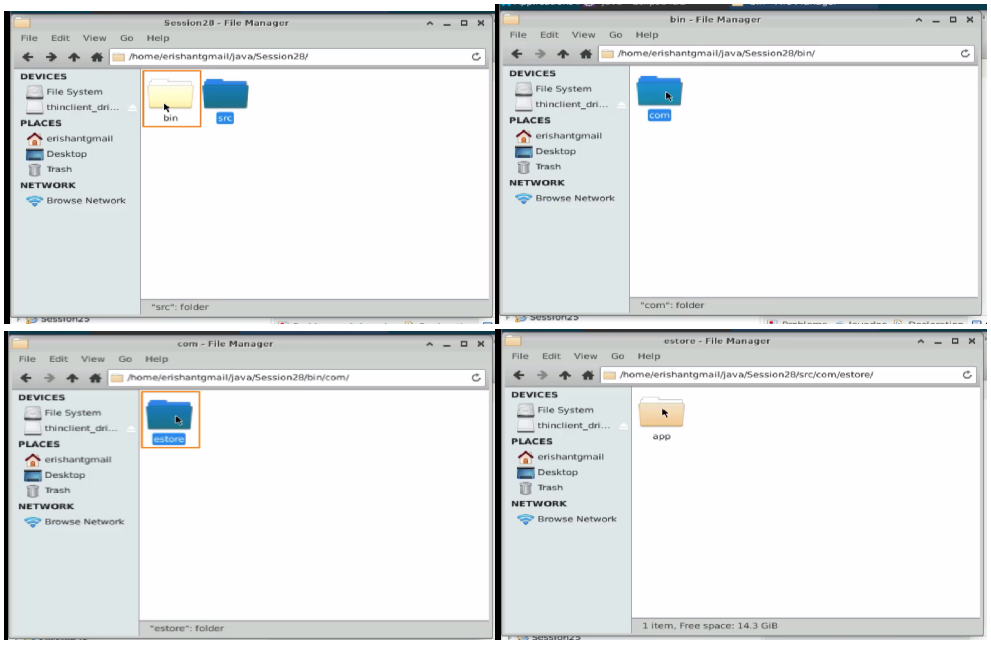
1.6 Name the java package as **com.estore.app.** The structure of the package name is the company's domain name in the reverse order. Select **Finish** and hence a package is created.



1.7 Open the **File Manager** and in the **src** folder of **Session28,** we will have a **com** folder, then the **estore** and then the **app** folder. Only the directory structure has been created in the app.

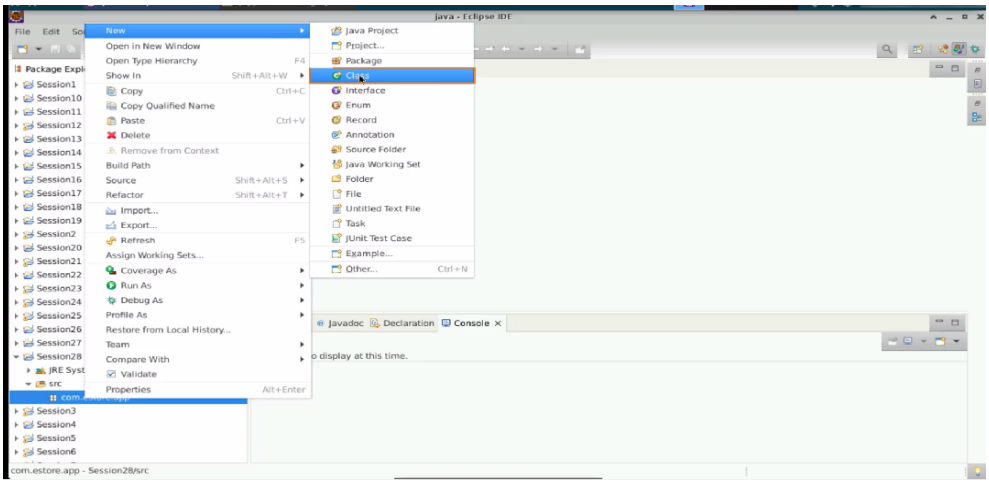


1.8 In the **bin** folder of **Session28** also**,** we will have a **com** folder, then the **estore** and then the **app** folder.

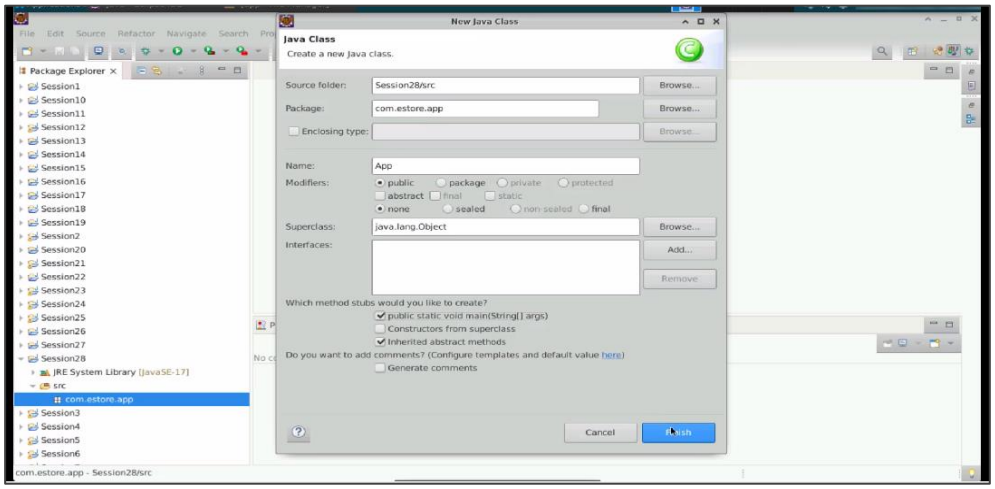


Note:- When you build a package, you are basically creating folders, and when you use the dot operator, you are creating nested folders.

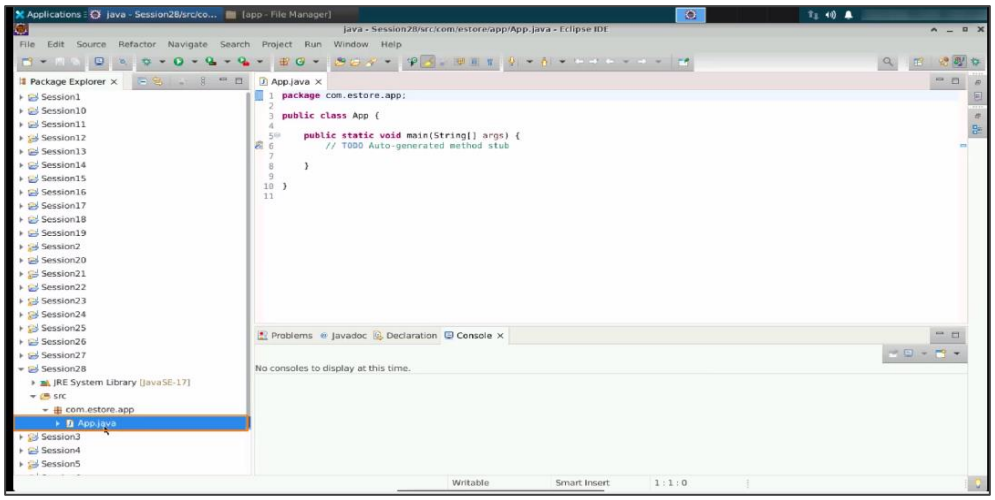
1.9 Right click on the package, select **New**, and then select **Class**.

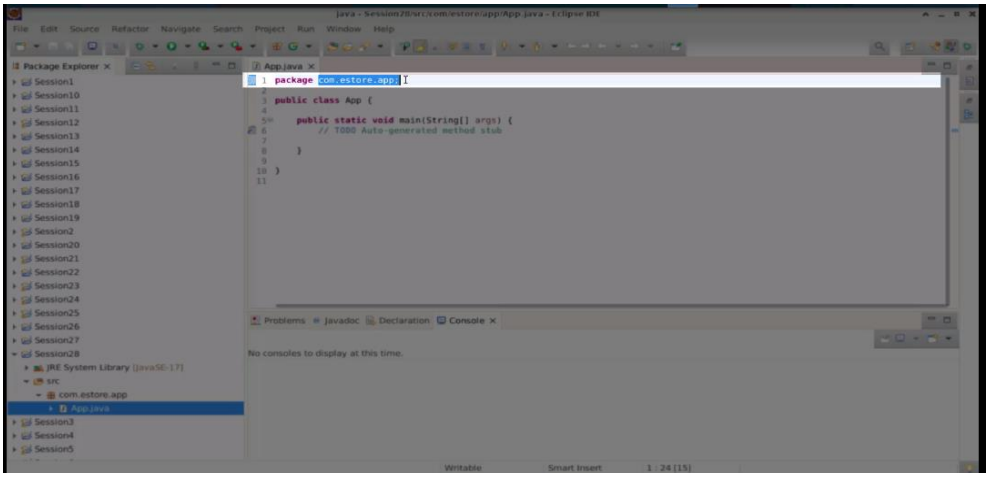


1.10 Name the class as **App** and select the option **public static void main(String[] args)**. Select **Finish**. This will create a java class with a **main()** method.

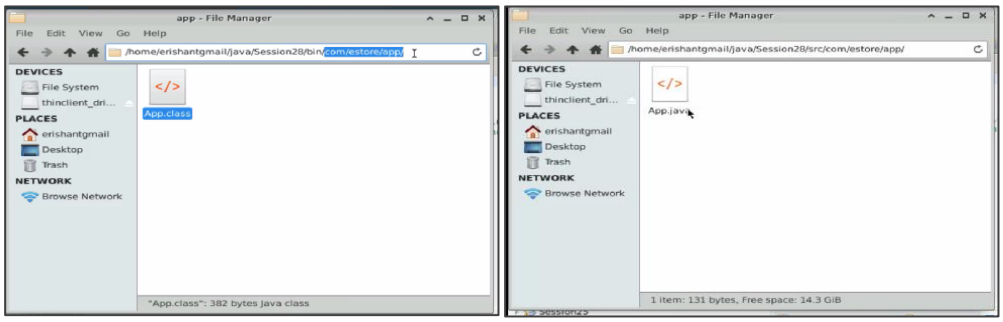


1.11 Inside the package, an **App.java** has been created and when you open this class, you can see a declaration on the top as **package com.estore.app** which means that when this class will be compiled, it will be created inside this package.





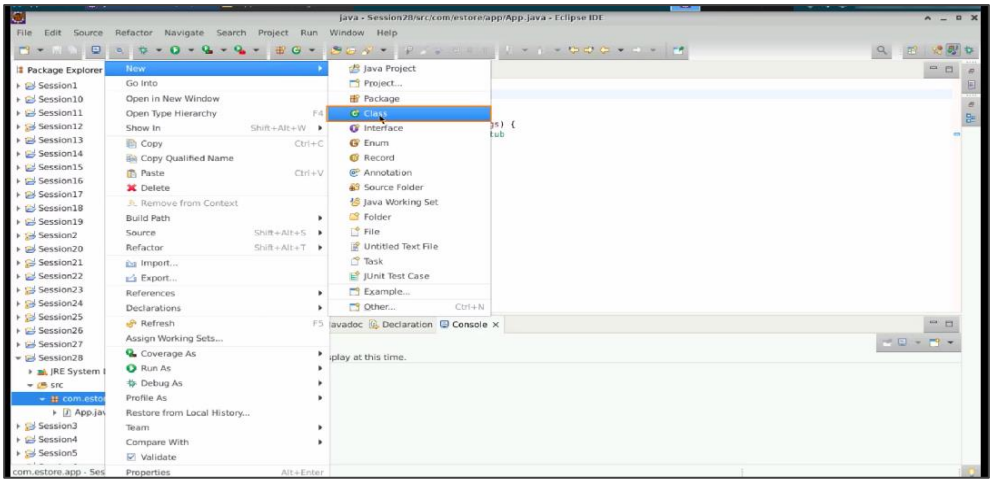
1.12 In the **File manager**, inside the **bin** folder, we will have **App.class** and inside the **src** folder, we will have **App.java**.



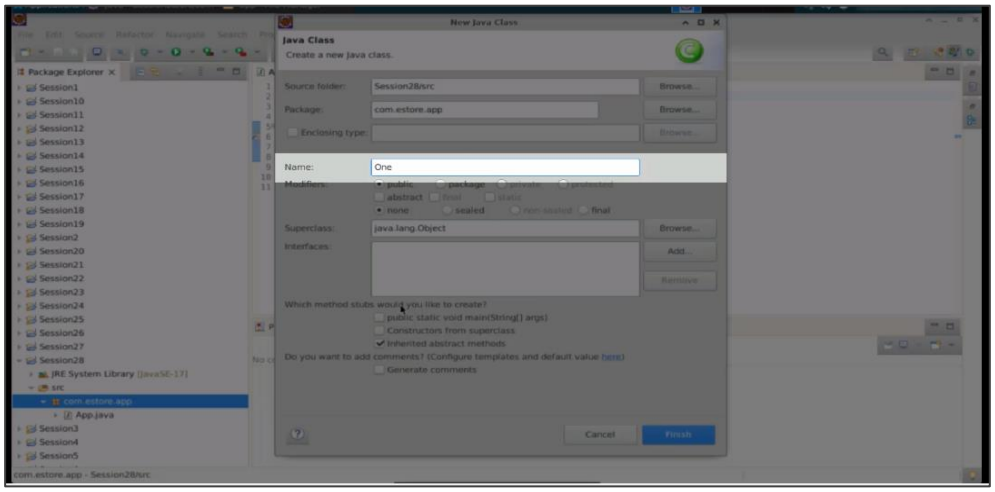
Note:- Classes and interfaces are categorised using Java packages to make maintenance easier. Java packages make the code organized.

Step 2 Implement the use of access modifiers in class declaration

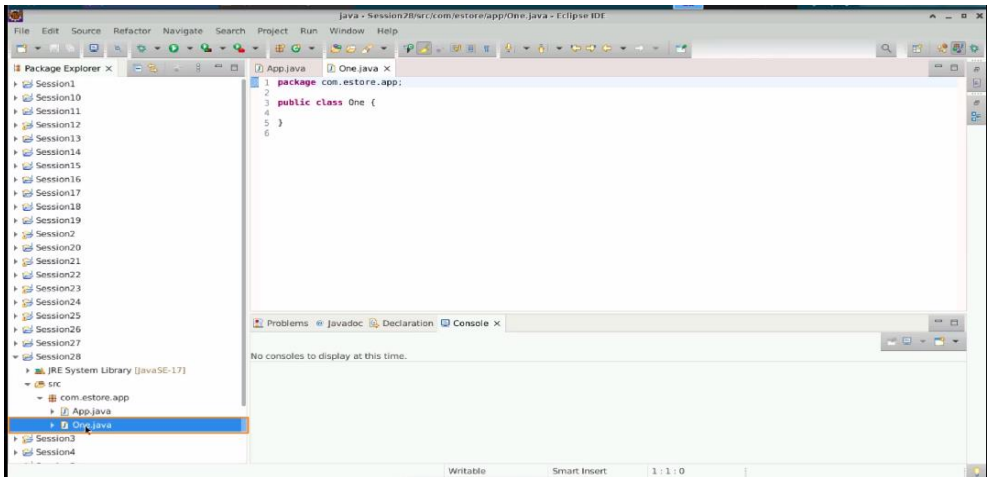
2.1 In the same package, right click, select **New** and then select **Class.**



2.2 Name the class as **One** to create a class without a main method.



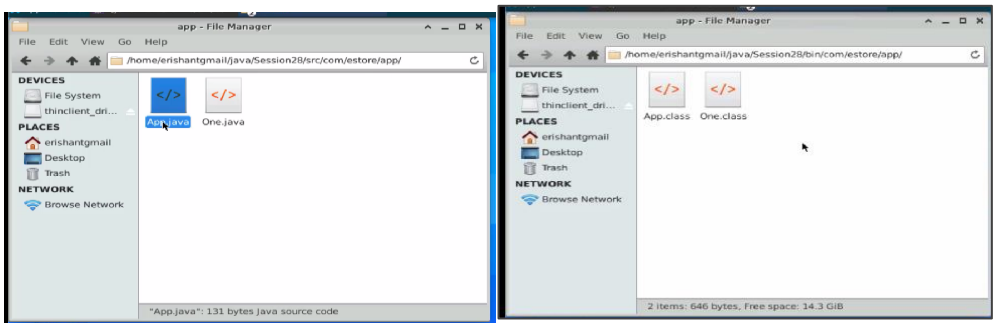
2.3 Thus there is a separate file called **One.java** in the same package.



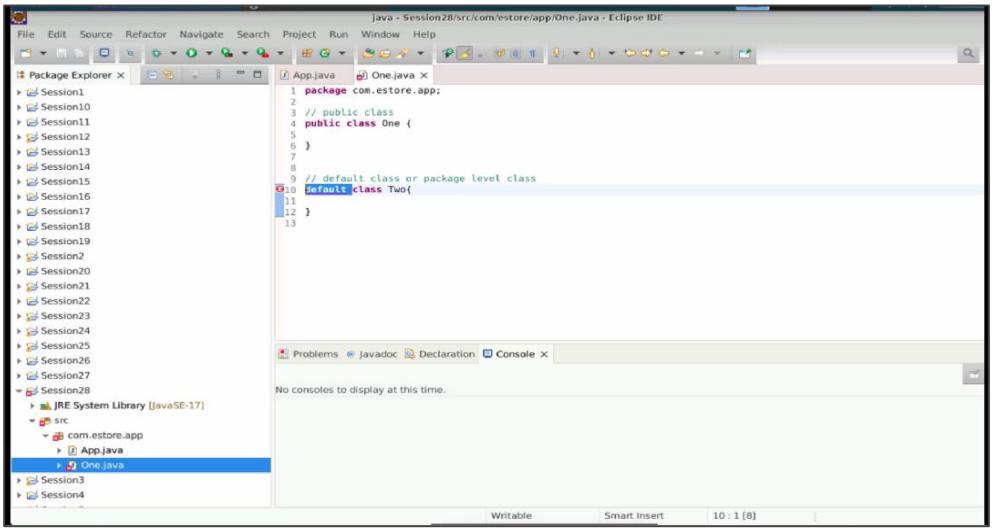
2.4 You can view the bytecodes by visiting the bin **com.estore.app.**

The bytecodes are in the other window, while the java file is in the first.

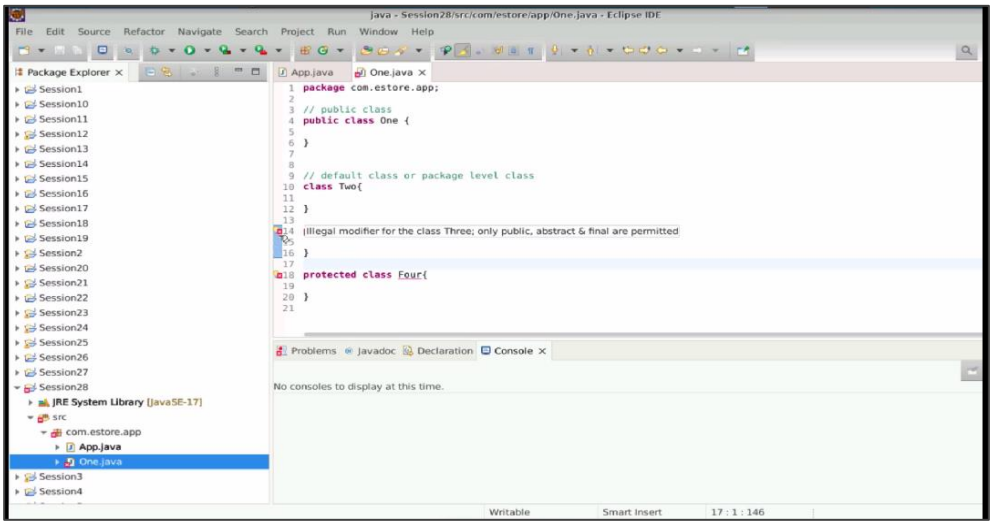
This is a straightforward description of what packages are and how to manage bytecodes.



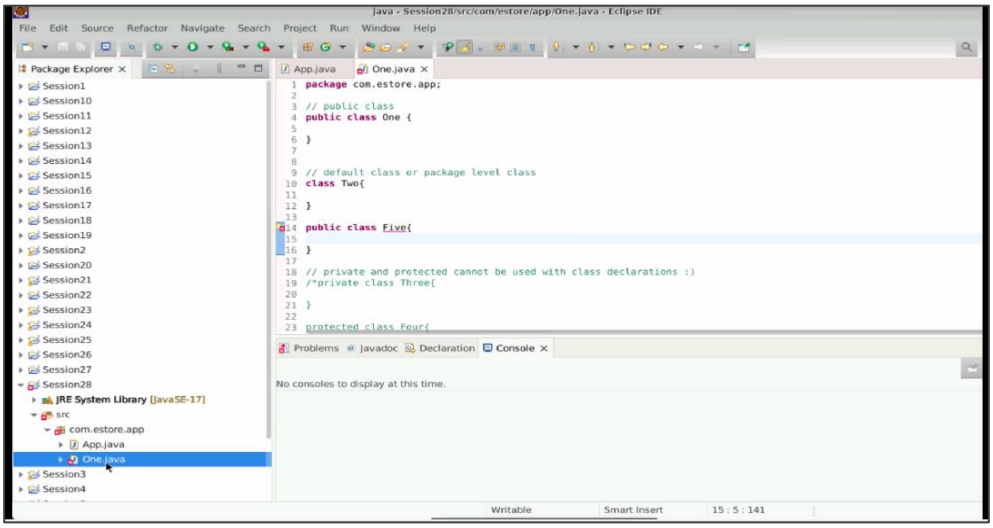
2.5 Class **One** is marked as public. Create another class called **Two**. This class will be a default class or a package level class and not a public class. If you try to put the default keyword in front of the class **Two** declaration, you will get an error because the **default keyword** is meant for interface methods.



2.6 If you try to make a class **public or protected** by writing a public or protected keyword in front of the class declaration, in both the cases an error will be raised. Hover over the error and it says that **illegal modifiers and only public, abstract and final are permitted.** This means that class can either be **default or public but not private or protected.**



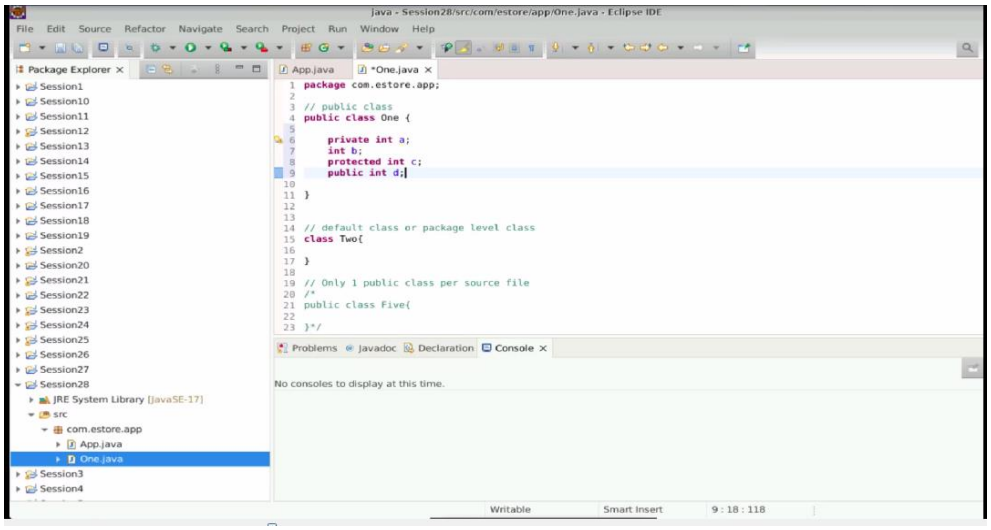
2.7 Create another public class as **public class five** in the same source file and an error will be raised which means **public class five** must be in its own file.



Note:- If a class has been created as public, then the name of the source file should be the same name as that of the class name. In a single source file, there could only be one public class. However, there can be multiple default classes.

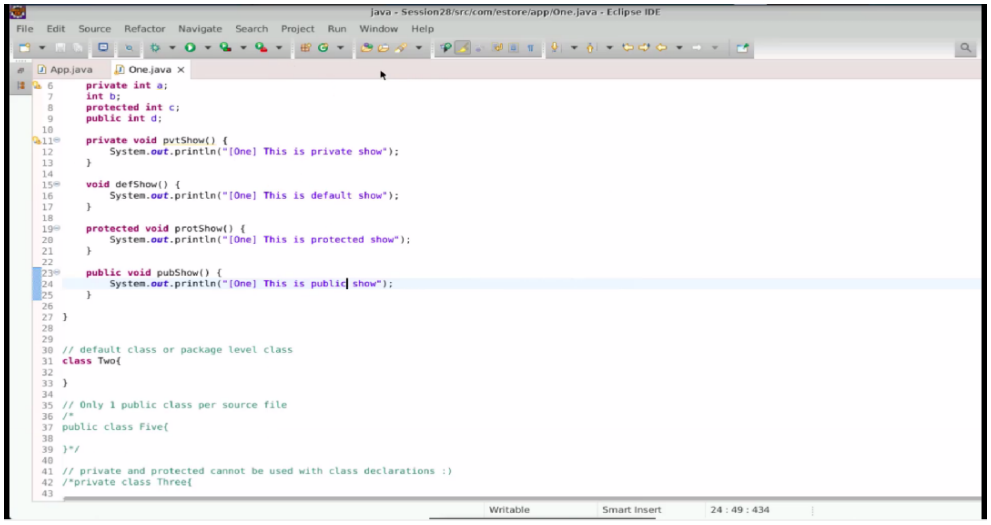
Step 3 Implement the use of access modifiers inside the class

3.1 Inside the class **One,** first define the attributes as **private int A, int B, protected int C and public int D.**



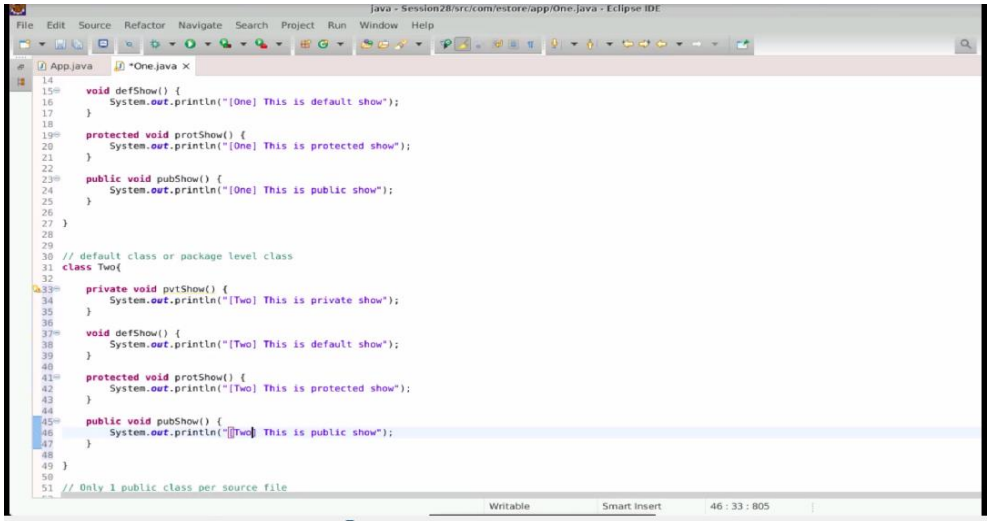
3.2 Now define the methods in class **One.** The methods are:

- **private void pvtShow() -** a method which prints **[One] this is a private method.**
- **void defShow() -** a method which prints **[One] this is a default method.**
- **protected void protShow() -** a method which prints **[One] this is a protected method.**
- **public void pubShow() -** a method which prints **[One] this is a public method.**



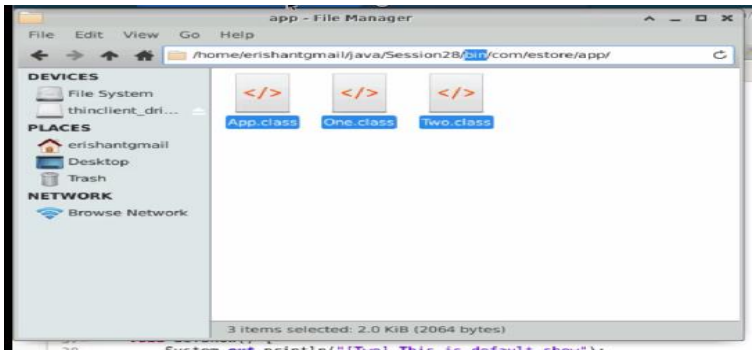
3.3 Similarly, define the methods in class **Two.** The methods are:

- **private void pvtShow() -** a method which prints **[Two] this is a private method.**
- **void defShow() -** a method which prints **[Two] this is a default method.**
- **protected void protShow() -** a method which prints **[Two] this is a protected method.**
- **public void pubShow() -** a method which prints **[Two] this is a public method.**

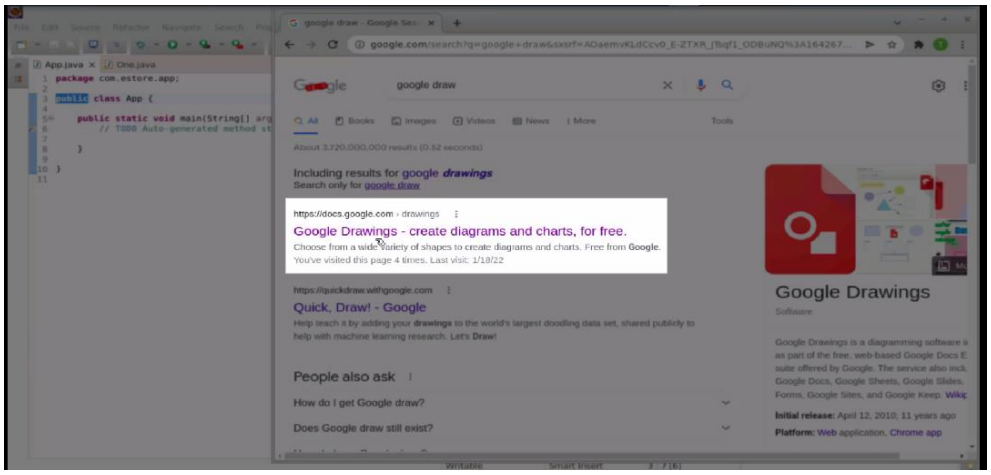


3.4 The package bin has three classes.

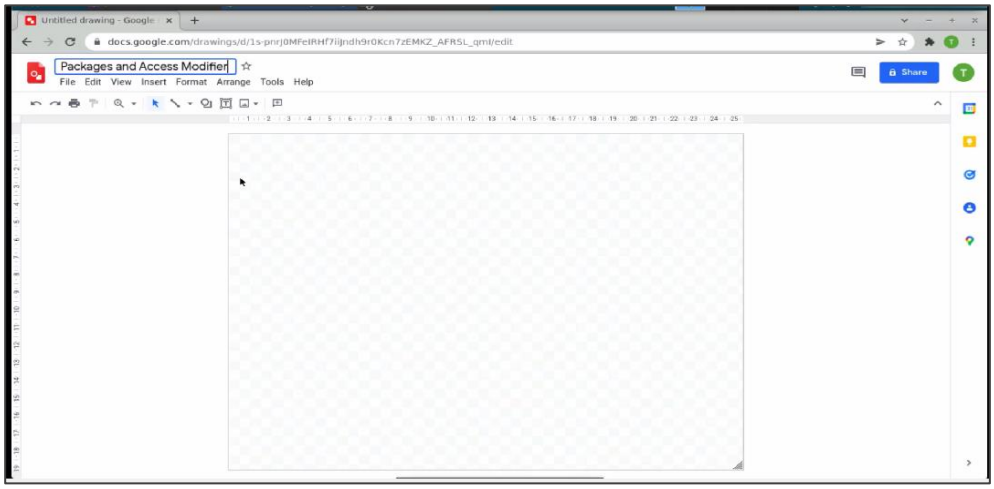
The **app** is **public**, class **one** is **public**, and class **Two** is **default** in the file manager. You can see the same thing in the eclipse IDE; class two is the default, class one is public, and since a file can only contain one public class, this App is also public in the other file under the name **App.Java.**



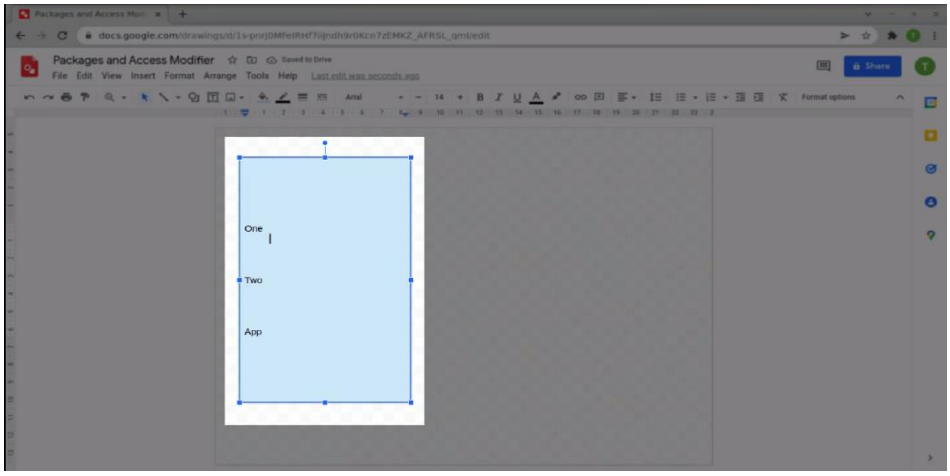
3.5 In your web browser, search **Google Draw** in the search bar and open it.



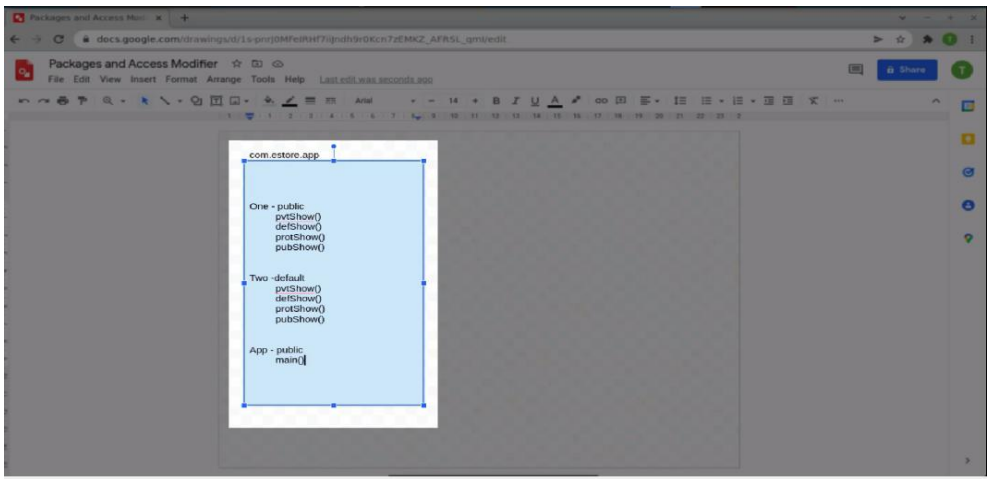
3.6 Name the file as **Packages and Access Modifiers**.



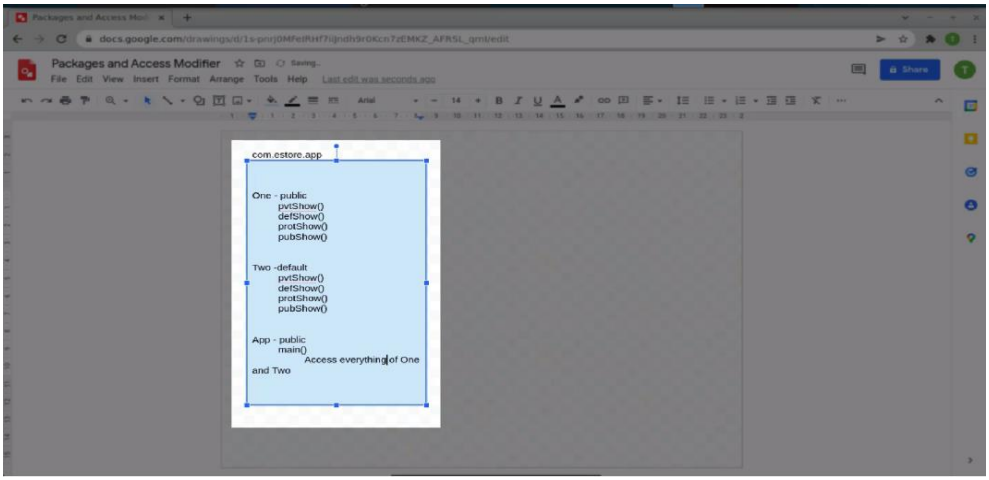
3.8 Draw a square box that will represent the package **com.estore.app**. Inside the package, we have class **One (public),** class **Two** (**default)** and class **App (public)**.



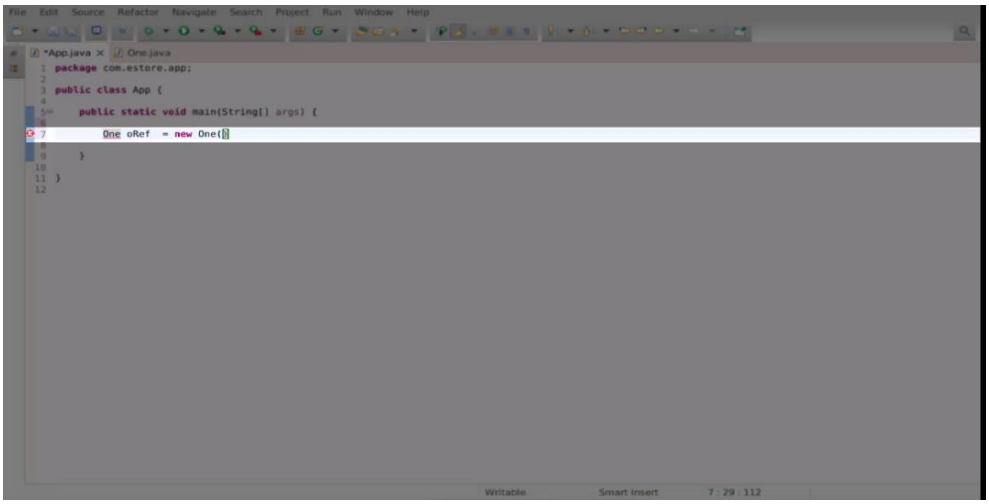
3.9 In class One, four methods are there : **pvtShow(), defShow(), protShow() and pubShow().** The same methods are there in class Two. Class App has a main method.



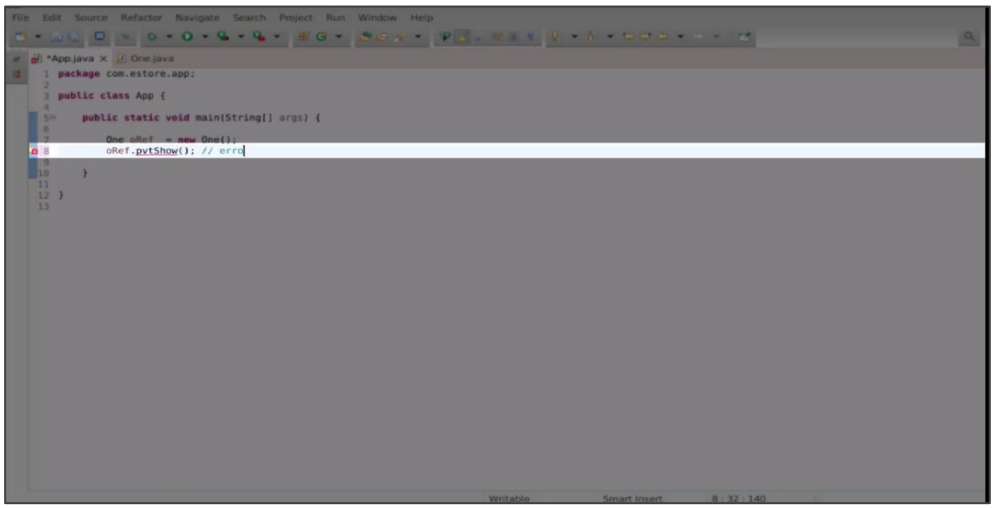
3.10 Inside the main(), everything of **One** and **Two** can be accessed other than **private attributes or methods.**

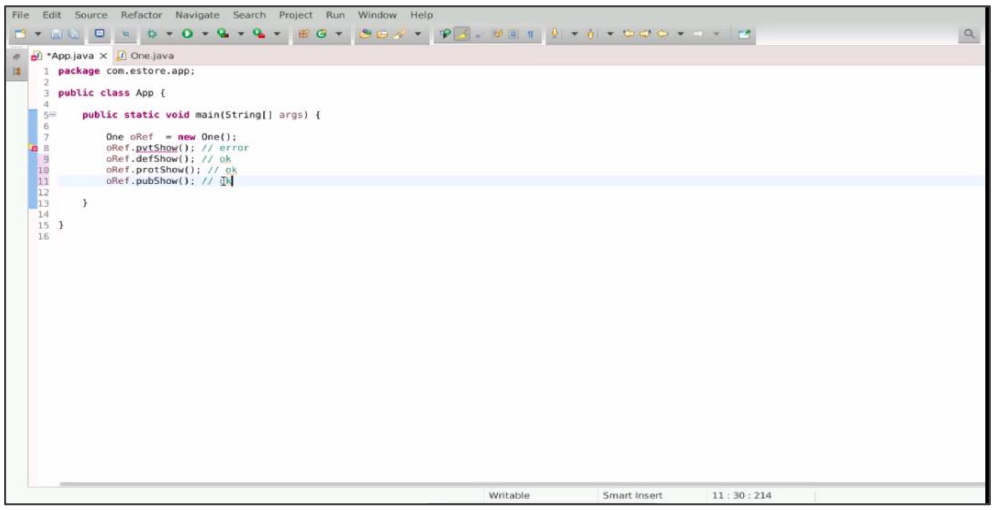


3.11 In the main() of **App.java,** create an object of **One** as **One oRef = new One**. This statement creates an object of class One.

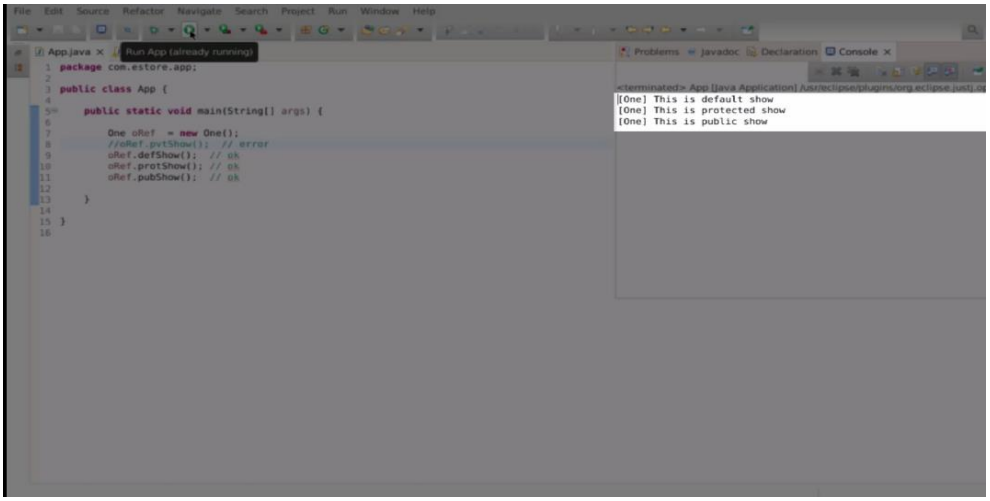


3.12 As soon as we write **oRef.pvtShow(),** an error is raised as we are trying to access a private method. The default show, protected show, and public show can all be accessed via the reference variable respectively. Therefore everything here is acceptable.

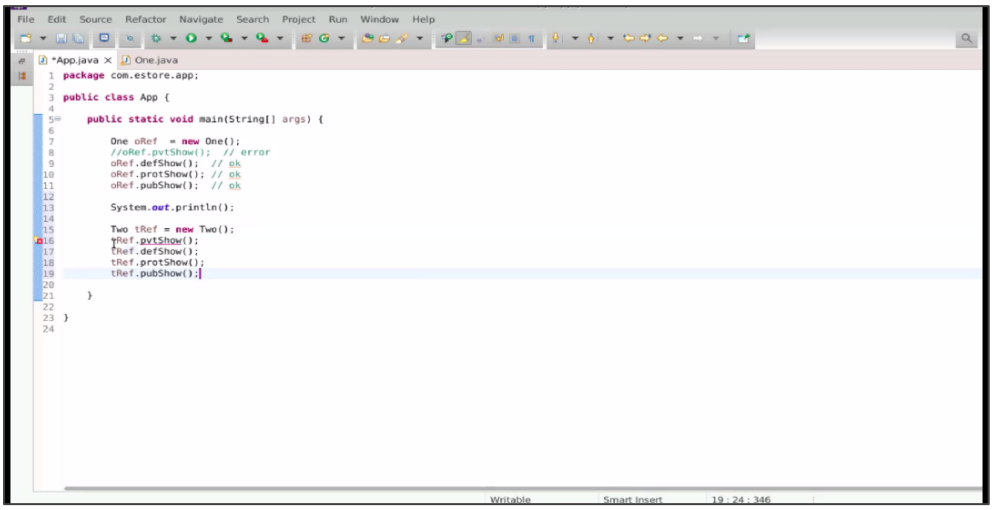




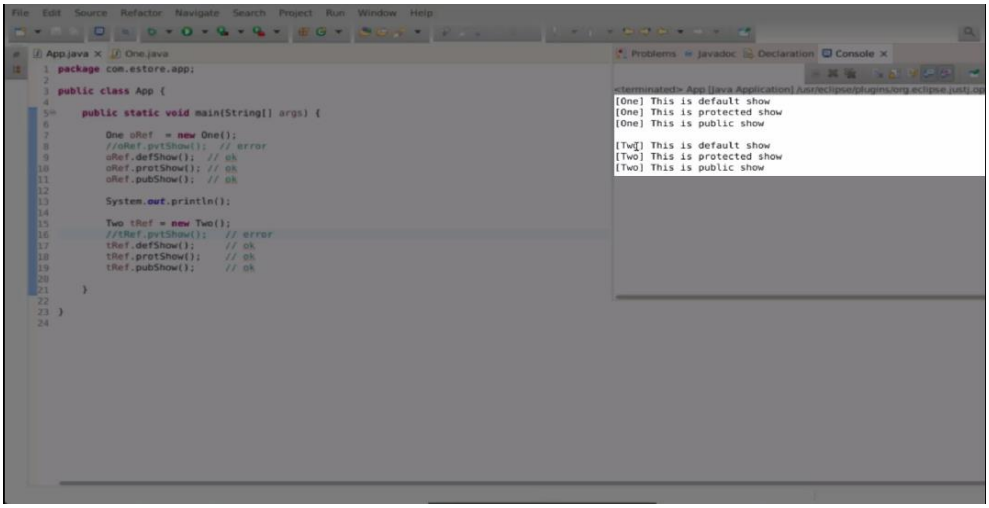
3.13 Comment down the private method and run the code. The following output will be obtained.



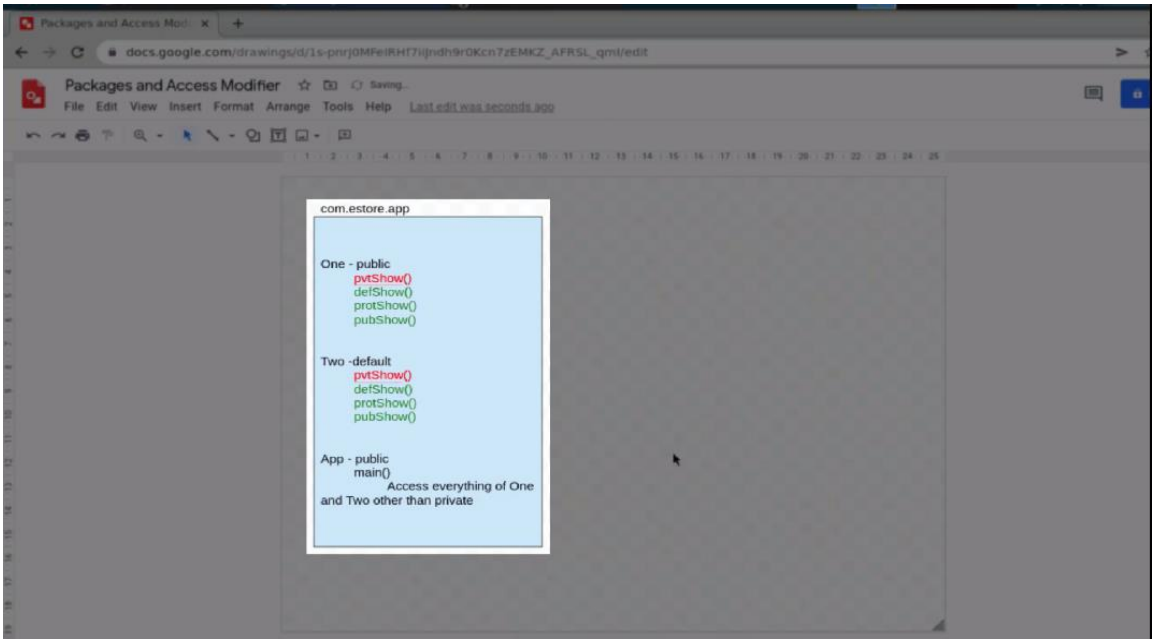
3.14 Create an object of class Two as **Two tRef = new Two()** and start accessing the methods. The default show, protected show, and public show can all be accessed via the reference variable but the private show will raise an error.

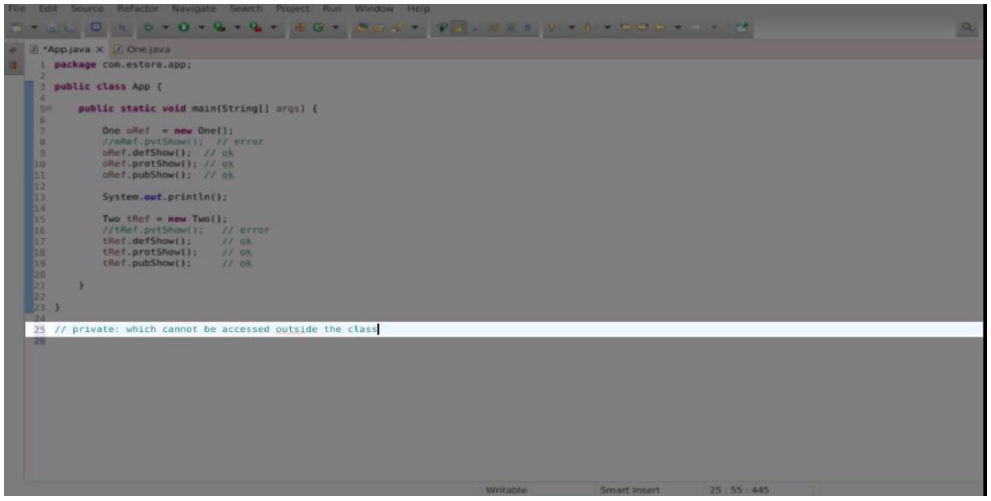


3.15 Comment down the private method and run the code. The following output will be obtained.



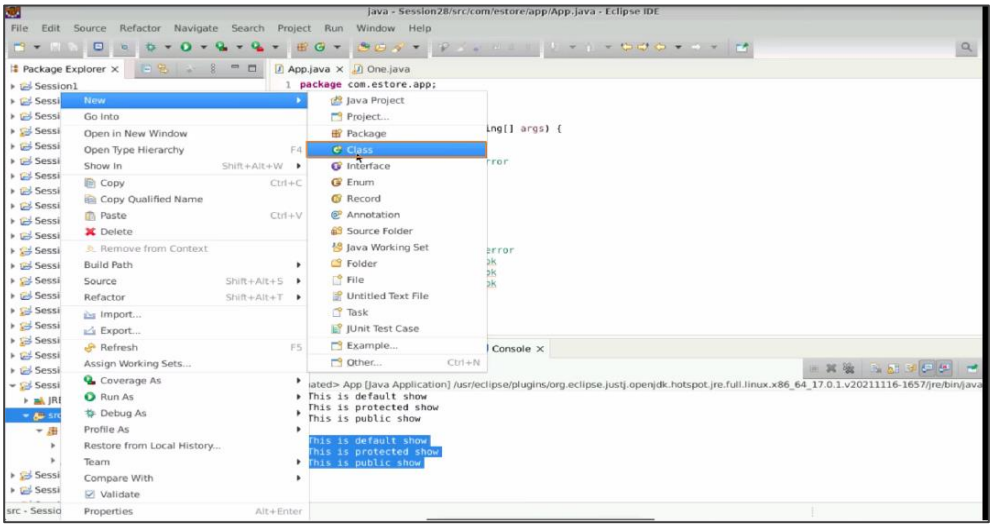
3.16 Private methods and attributes cannot be accessed outside their class. Apart from private, the other three access modifiers (public, protected, and default) allow access outside the class if they are in the same package. Marking an attribute as private restricts its access to within the class, thereby achieving encapsulation. You can then access these private members using methods, or through getters and setters.



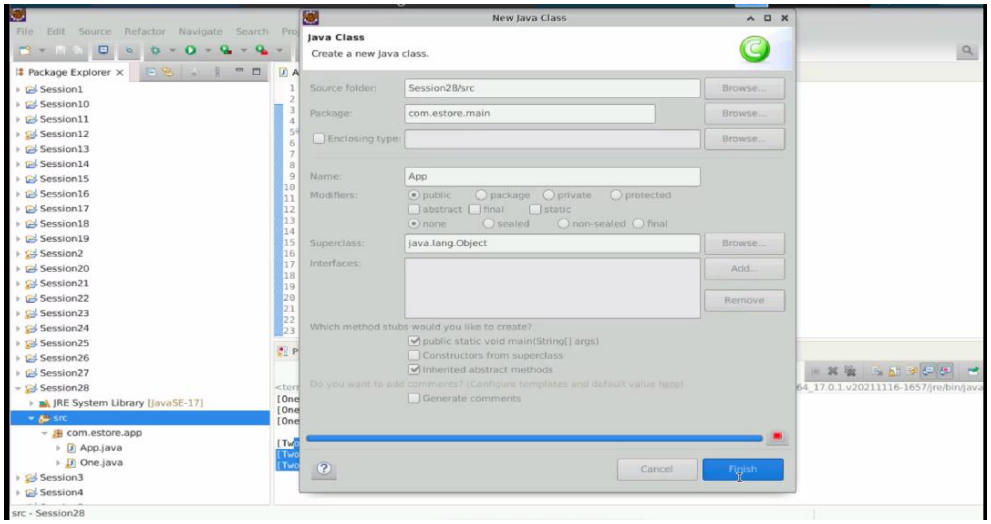


Step 4 : Implement the use of access modifiers outside the class

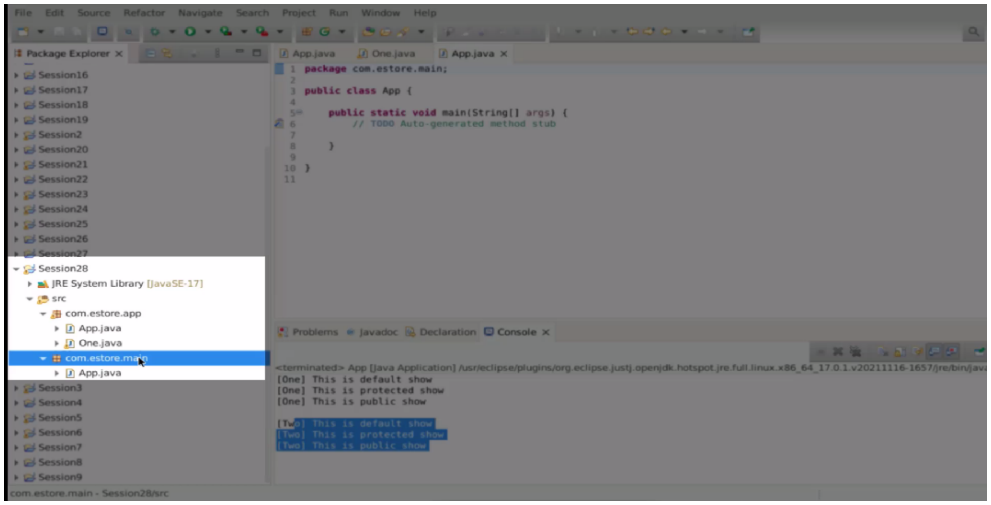
4.1 In the same package, right click, select **New** and then select **Class.**



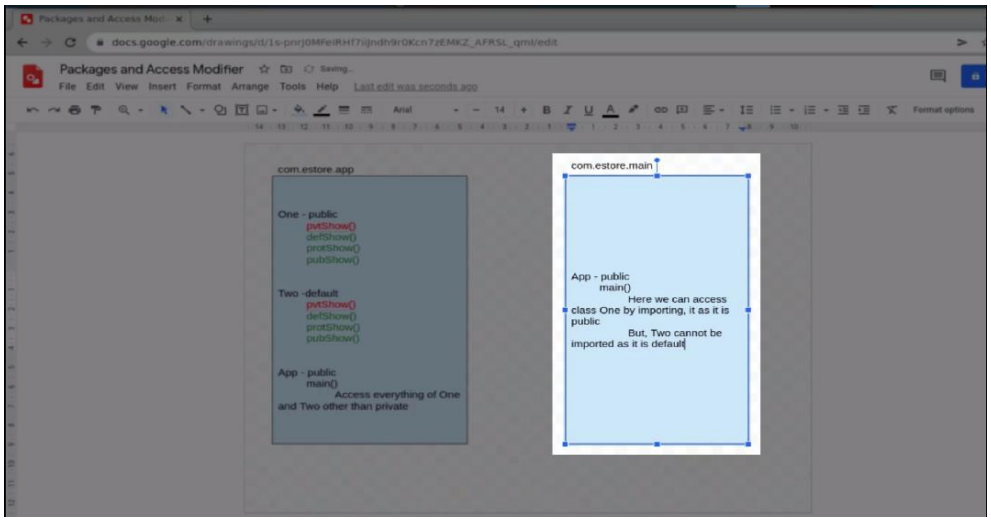
4.2 Name the class as **App** but change the package name to **com.estore.main**. Select **Finish.**



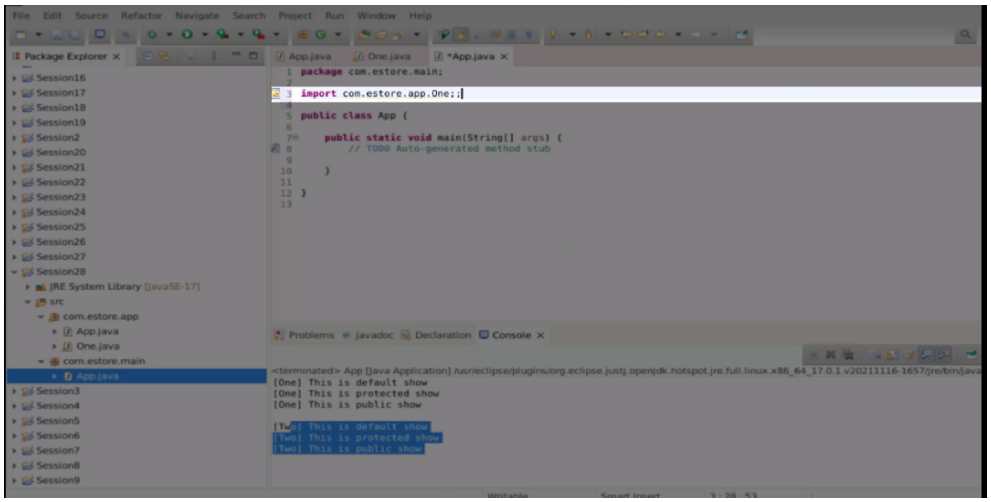
4.3 There can be different packages with the same class name.

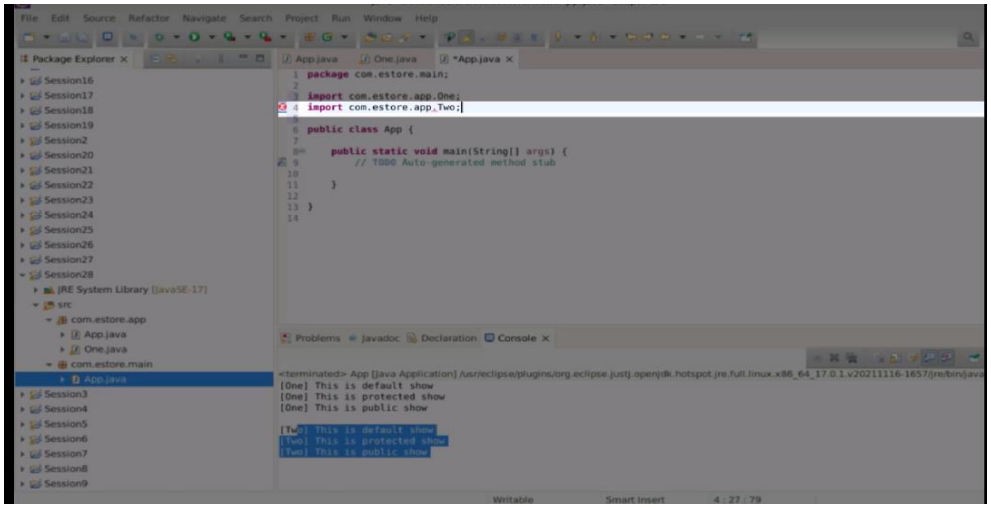


4.4 In the package **com.estore.main**, a class named **App (public)** has been created with a main() method. In the class **App** of this package, class **One** of the other package can be imported as it is public. But class **Two** cannot be imported as it is **default**.

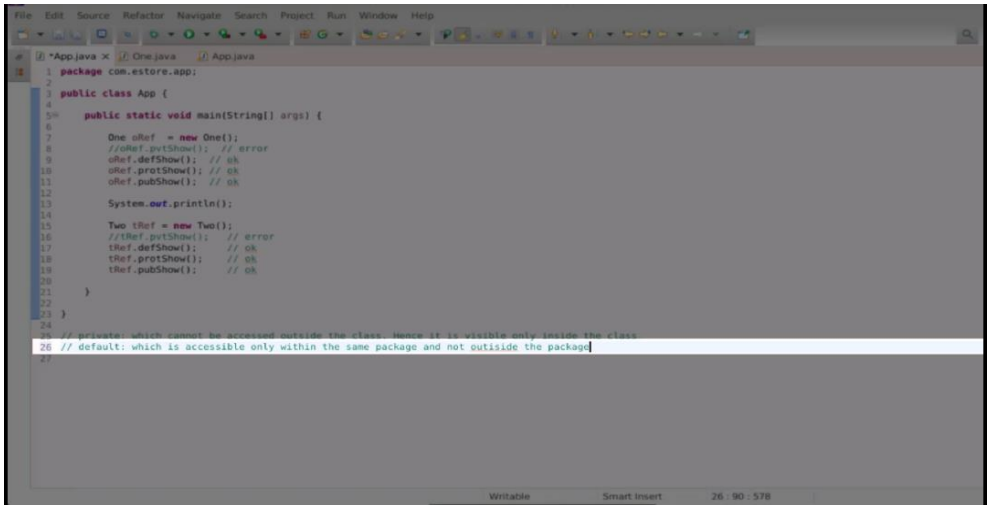


4.5 Inside the **App** class of **com.estore.main** package, import class One by writing **import com.estore.app.One.** This works fine and class One has been imported. But as you try to import class Two by writing **com.estore.app.Two,** an error is raised as default methods are accessible inside the same package.

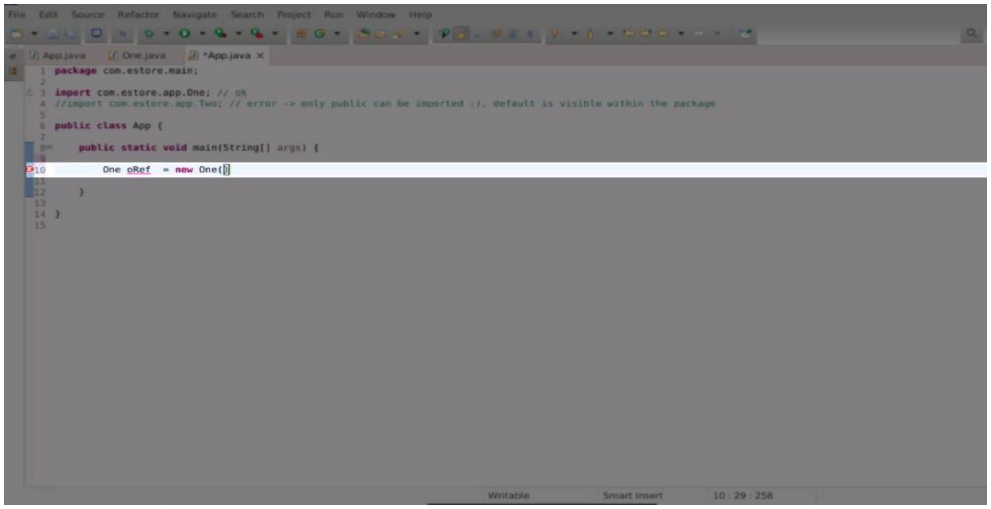




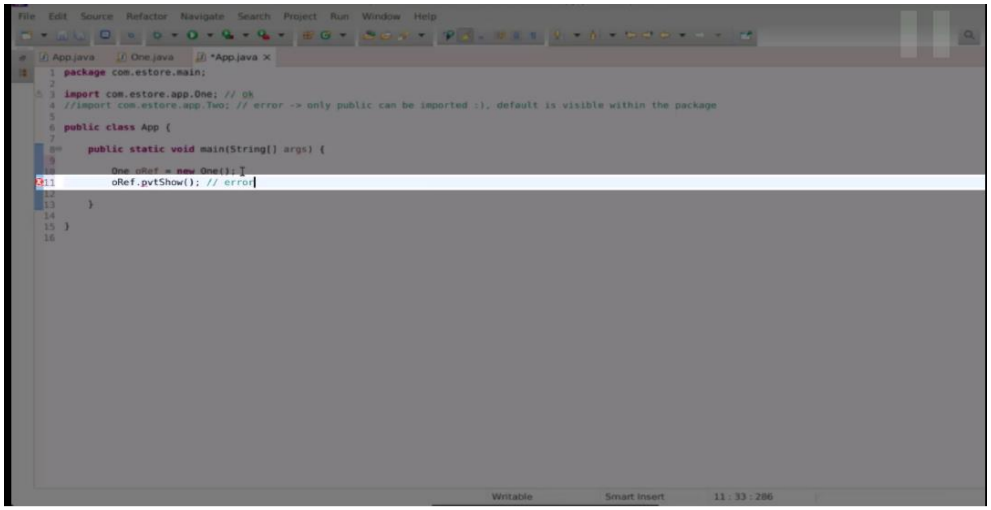
4.6 **default** is accessible only within the same package and not outside the package.



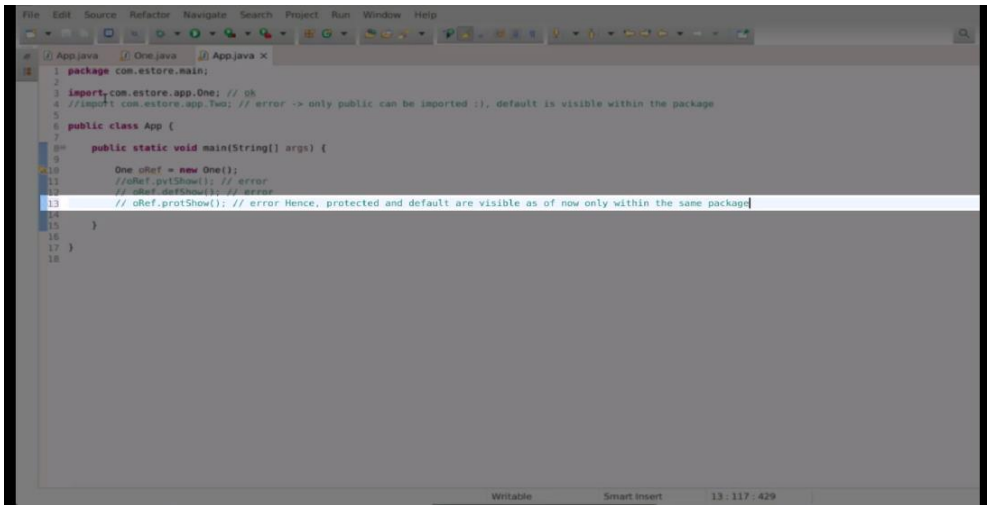
4.7 Inside the package **com.estore.main,** create an object of class One as **One oRef = new One().**



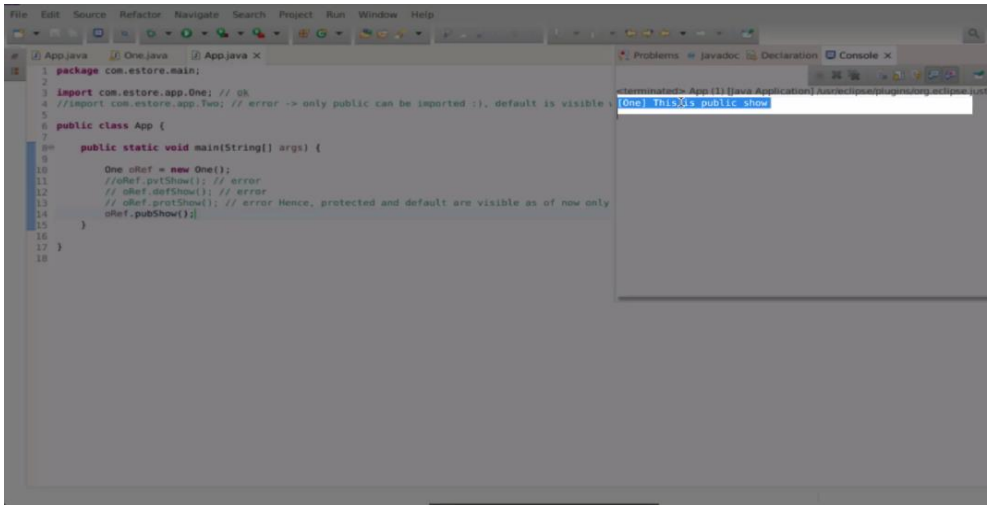
4.8 Write **oRef.pvtShow()** and try to access the private show, it raises an error. This is because private is accessible inside the same class.



4.9 Write **oRef.defShow() and oRef.protShow()** and try to access the default show and protected show, but it also raises an error. This is because default and protected are accessible inside the same package.

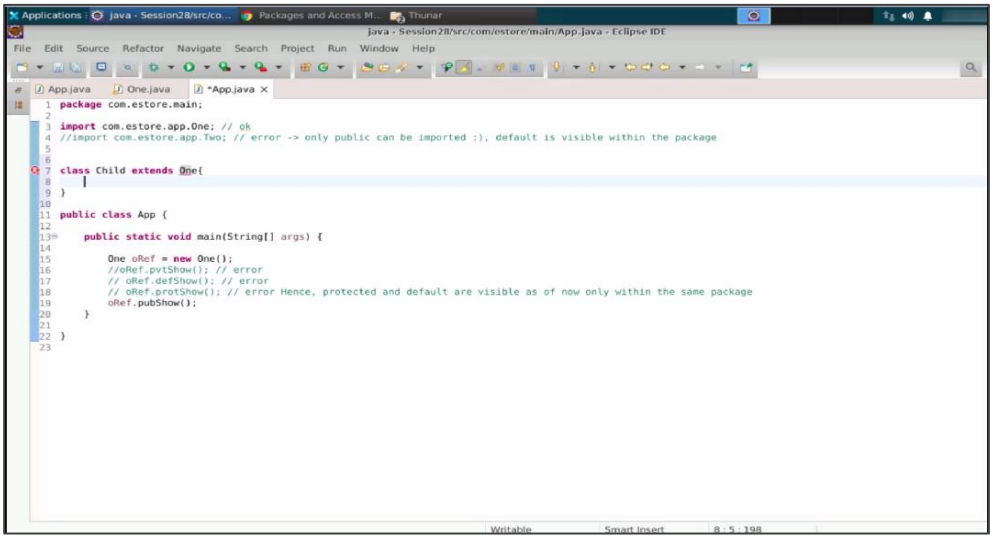


4.10 Write **oRef.pubShow()** and try to access the public show, it works fine. This is because public is accessible everywhere i.e. inside the class or outside the class.

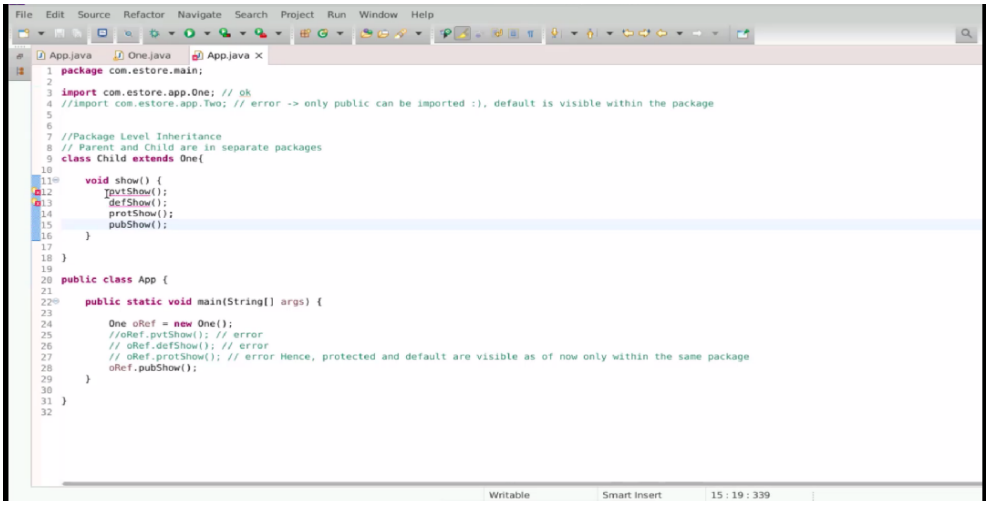


Step 5 : Implement the difference between protected and default access modifier

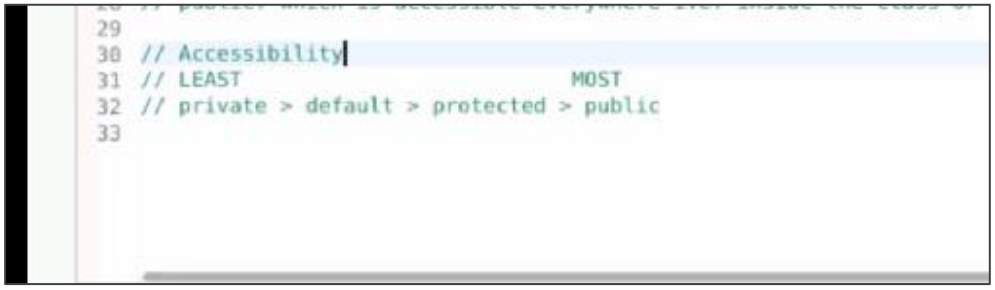
5.1 Inside the package, **com.estore.main**, create a child class of class **One** by writing **class Child extends One**, which means One is the parent class and Child is the child class. This is called **package level inheritance,** where parent and child are in separate packages.



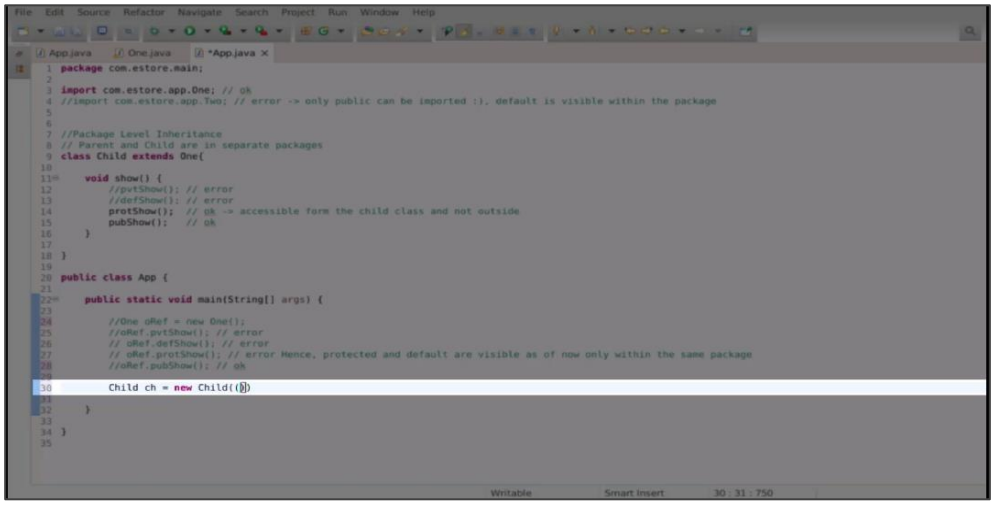
5.2 Inside the child class, create a method **void show(),** Inside this method, try to access the four methods of class **One. pvtShow() and defShow()** will raise an error as they cannot be accessed outside the class. But **protShow() and pubShow()** will work fine. This implies that **protected** is accessible from the child class outside the package.

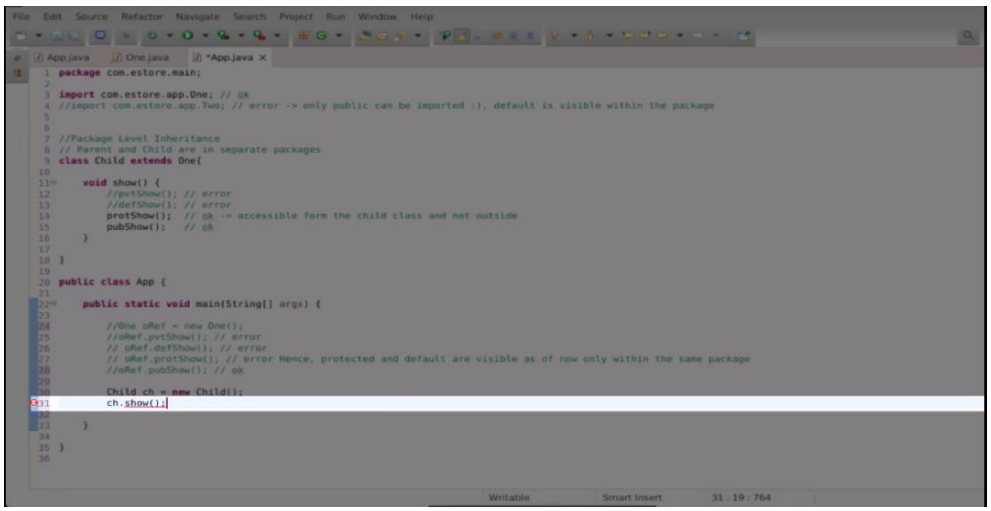


5.3 Accessibility from low to high is as follows:

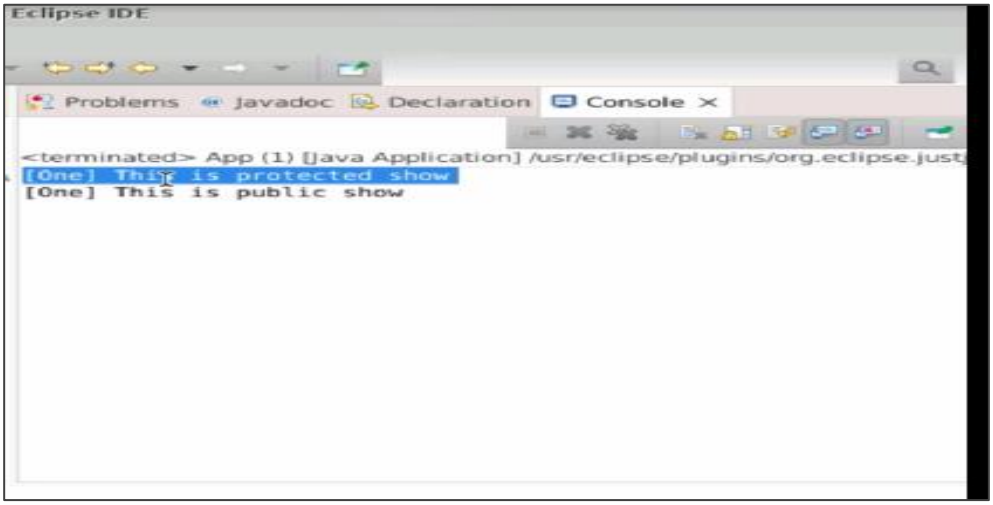


5.4 Create the object of the child class as **Child ch = new Child()** and call the show() method of the child class.





5.5 Run the code and the following output will be obtained:



By following the above steps, you have successfully implemented packages and access modifiers in Java, ensuring organized code structure and controlled accessibility.