IS 2104 - Rapid Application Development
Java Packages

M.V.P. Thilini Lakshika
University of Colombo School of Computing
tlv@ucsc.cmb.ac.lk

Lesson Outline

- What is Package in Java?
- Create a Package in Java
- Importing packages

What is Package in Java?

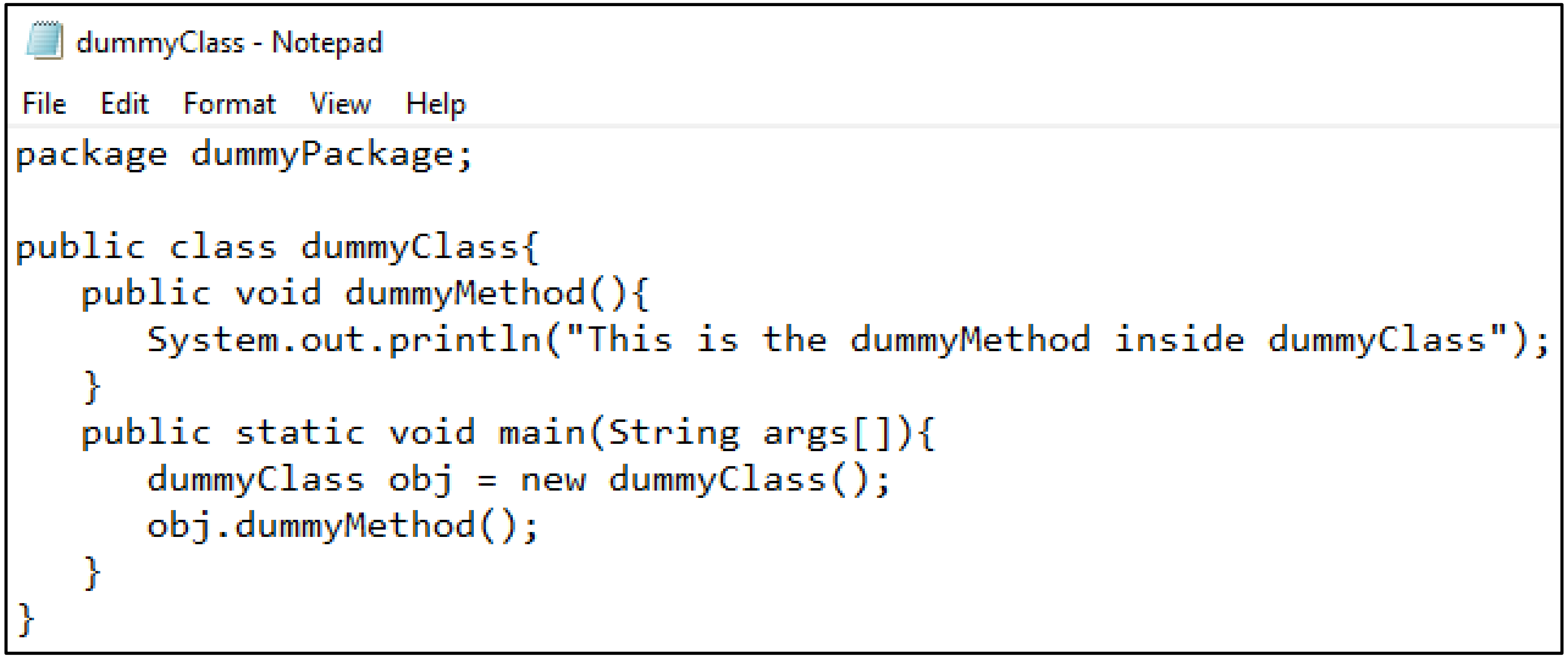
- A Package is a collection of related classes.
- Package organize classes into a folder structure and make it easy to locate and use them.
- More importantly, packages improve re-usability.
- Each package in Java has its unique name and organizes its classes and interfaces into a separate namespace, or name group.

  Syntax :- package nameOfPackage;

- Although interfaces and classes with the same name cannot appear in the same package, they can appear in different packages.

Create a Package in Java

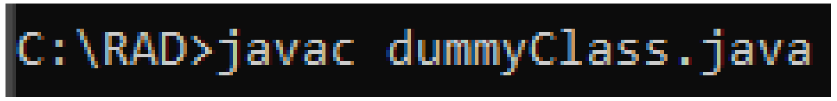
- Save this file as dummyClass.java



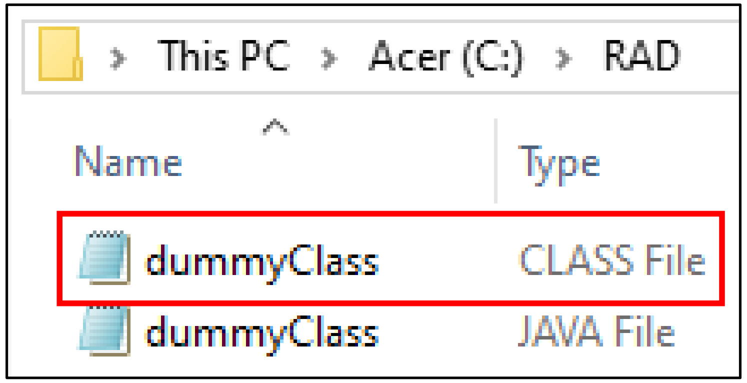
```
package dummyPackage;

public class dummyClass{
   public void dummyMethod(){
      System.out.println("This is the dummyMethod inside dummyClass");
   }
   public static void main(String args[]){
      dummyClass obj = new dummyClass();
      obj.dummyMethod();
   }
}
```

Create a Package in Java

- Next compile the demo.java file.



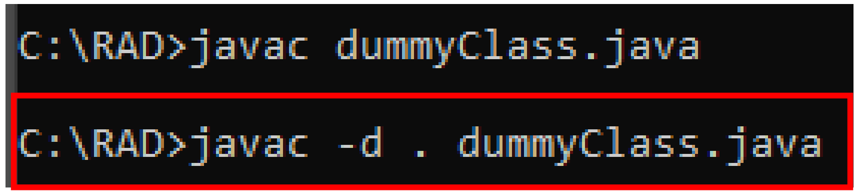
```
C:\RAD>javac dummyClass.java
```



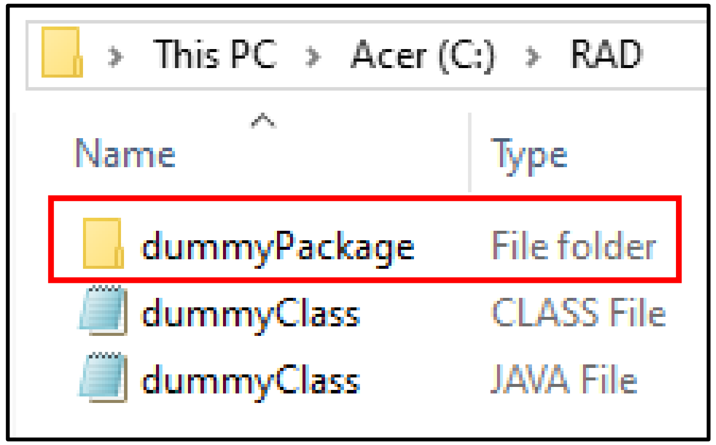
This PC › Acer (C:) › RAD

Name	Type
dummyClass	CLASS File
dummyClass	JAVA File

- The compilation is completed. The class file dummyClass is created. However, no package is created.
- Then create the package using below command. The "." operator represents the current working directory.



```
C:\RAD>javac dummyClass.java

C:\RAD>javac -d . dummyClass.java
```

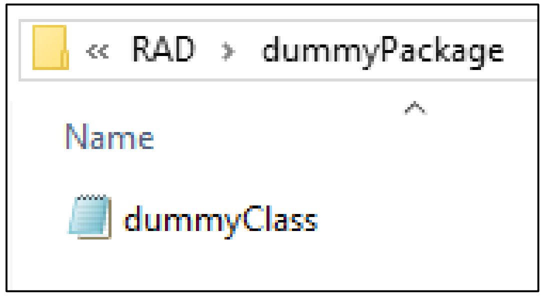


This PC › Acer (C:) › RAD

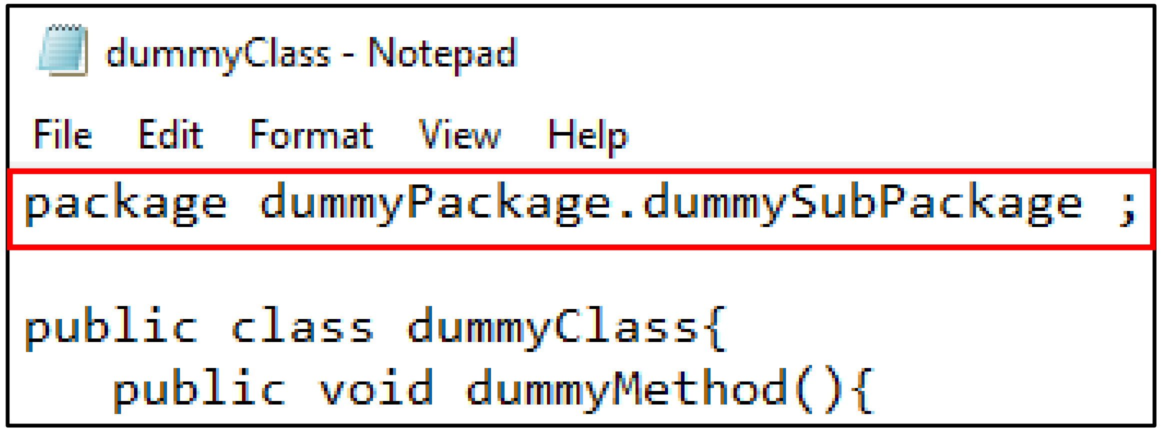
Name	Type
dummyPackage	File folder
dummyClass	CLASS File
dummyClass	JAVA File

Create a Package in Java

- After the execution of the code, it creates a package dummyPackage.
- Open the java package dummyPackage. Inside the package, you will see the dummyClass.class file.



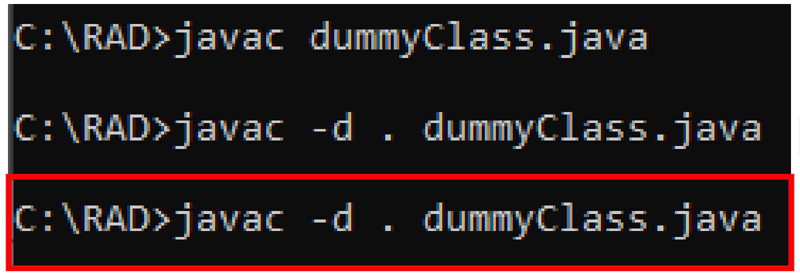
- Next create the sub package dummySubPackage within existing java package dummyPackage.



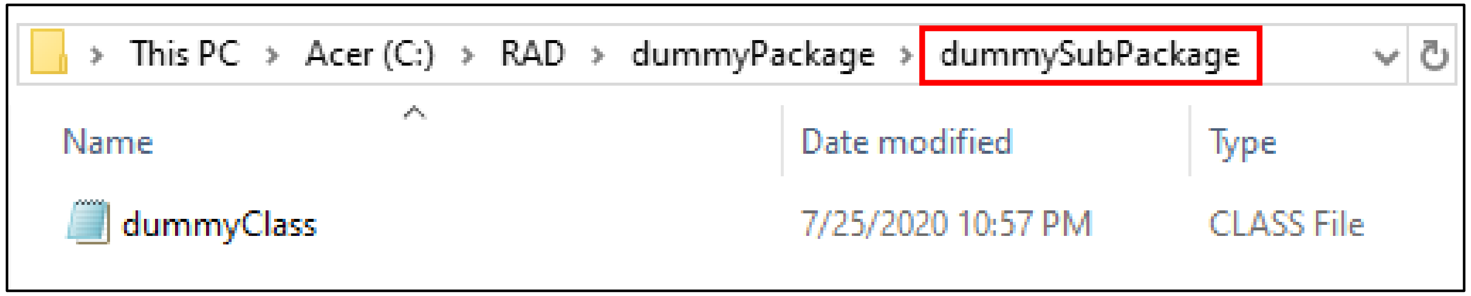
```
package dummyPackage.dummySubPackage ;

public class dummyClass{
   public void dummyMethod(){
```

Create a Package in Java

- Compile the demo.java file again.



```
C:\RAD>javac dummyClass.java

C:\RAD>javac -d . dummyClass.java

C:\RAD>javac -d . dummyClass.java
```

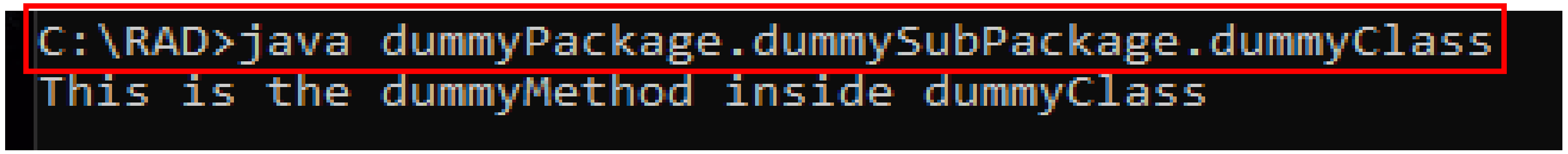


This PC › Acer (C:) › RAD › dummyPackage › dummySubPackage

Name	Date modified	Type
dummyClass	7/25/2020 10:57 PM	CLASS File

- It creates a sub package dummySubPackage having class dummyClass inside the package.
- How to execute the code with the fully qualified name of the class?

Ex: The package name followed by the sub package name followed by the class name.



```
C:\RAD>java dummyPackage.dummySubPackage.dummyClass
This is the dummyMethod inside dummyClass
```

Importing packages

- To create an object of a class (bundled in a package), in your code, you have to use its fully qualified name.

  Ex: java.awt.event.actionListner object = new java.awt.event.actionListner();
- But, it could become tedious to type the long dot-separated package path name for every class you want to use.
- Instead, it is recommended you to use the import statement.

  Syntax : **import** packageName;
- Once imported, you can use the class without mentioning its fully qualified name.
- There are two types of import statements: specific import and wildcard import.

Importing packages

- The specific import specifies a single class in the import statement.

  Ex: import javax.swing.JOptionPane; // imports JOptionPane from package javax.swing.

- The wildcard import imports all the classes in a package.

  Ex: import javax.swing.*; // imports all classes from package javax.swing.

- The information for the classes in an imported package is not read in at compile time or runtime unless the class is used in the program.
- The import statement simply tells the compiler where to locate the classes.
- There is no performance difference between a specific import and a wildcard import declaration.

Importing packages

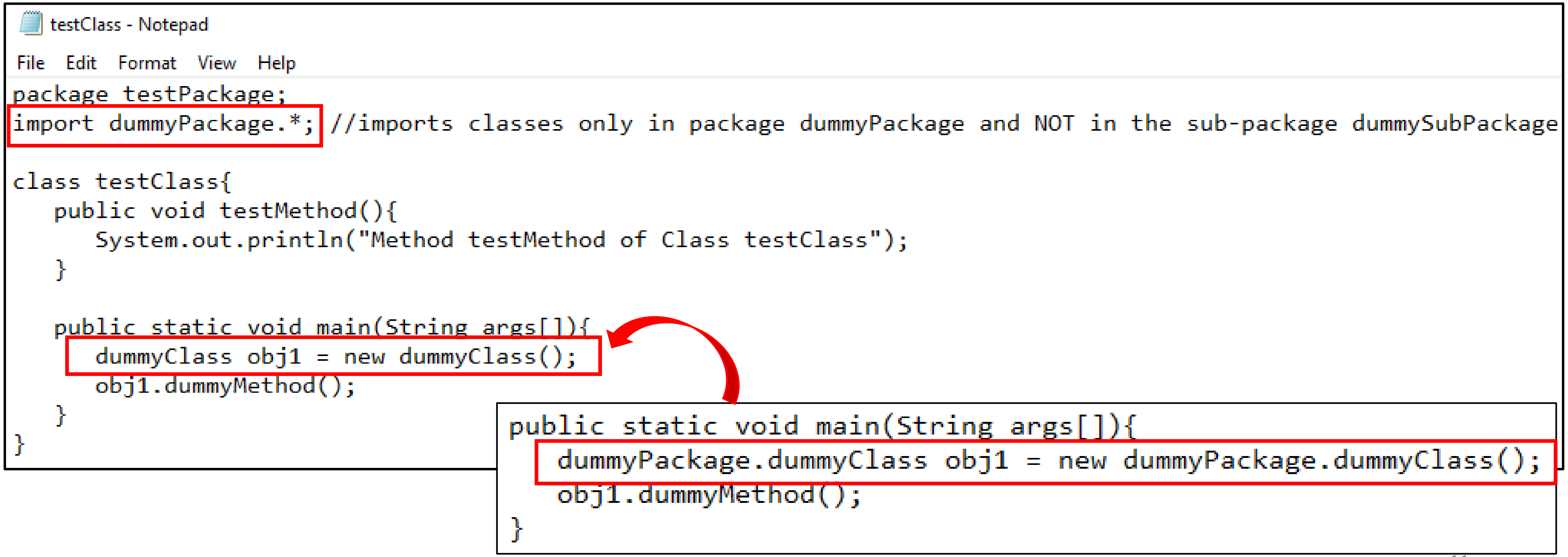
Ex:

```
import java.awt.event.*; // * signifies all classes in this package are imported

import javax.swing.Jframe; // Only the JFrame class is imported

JFrame f = new JFrame; // Use without fully qualified name.
```

- Consider the System class in the statement System.out.println(”Welcome to Java”);
- The System class is not imported because, it is in the java.lang package.
- All the classes in the java.lang package are implicitly imported in every Java program.

Importing packages

- Copy the code into an editor and save the file as testClass.java

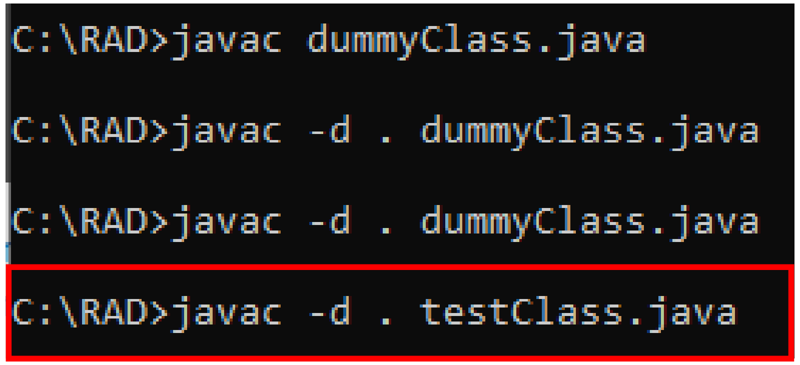


```
package testPackage;
import dummyPackage.*; //imports classes only in package dummyPackage and NOT in the sub-package dummySubPackage

class testClass{
   public void testMethod(){
      System.out.println("Method testMethod of Class testClass");
   }

   public static void main(String args[]){
      dummyClass obj1 = new dummyClass();
      obj1.dummyMethod();
   }
}
```

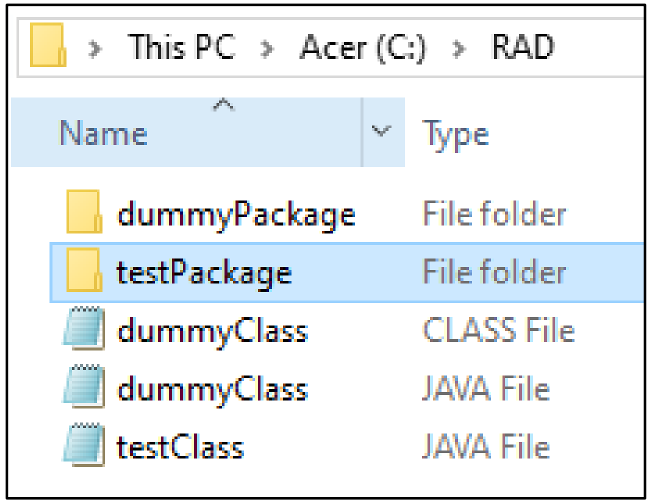
```
public static void main(String args[]){
   dummyPackage.dummyClass obj1 = new dummyPackage.dummyClass();
   obj1.dummyMethod();
}
```

Importing packages

- Compile the testClass.java using the below command.



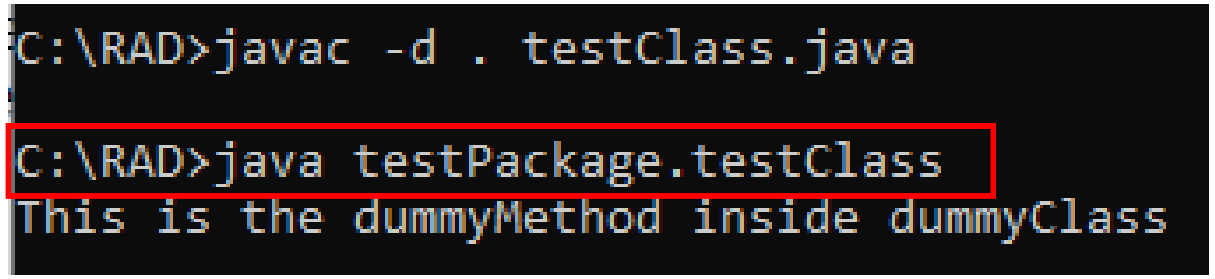
```
C:\RAD>javac dummyClass.java

C:\RAD>javac -d . dummyClass.java

C:\RAD>javac -d . dummyClass.java

C:\RAD>javac -d . testClass.java
```



This PC > Acer (C:) > RAD

Name	Type
dummyPackage	File folder
testPackage	File folder
dummyClass	CLASS File
dummyClass	JAVA File
testClass	JAVA File

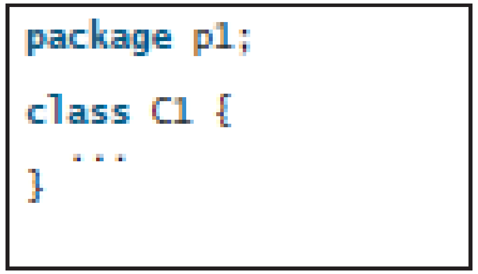
- Execute the code using the below command and check the results.



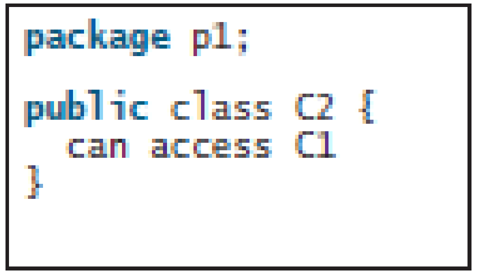
```
C:\RAD>javac -d . testClass.java

C:\RAD>java testPackage.testClass
This is the dummyMethod inside dummyClass
```

Importing packages

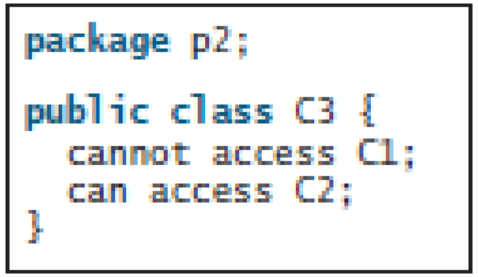
- The private modifier restricts access to its defining class, the default modifier restricts access to a package, and the public modifier enables unrestricted access.
- If a class is not defined public, it can be accessed only within the same package.
- As shown in below figure, **C1** can be accessed from **C2** but not from **C3**.



```
package p1;

class C1 {
  ...
}
```



```
package p1;

public class C2 {
  can access C1
}
```



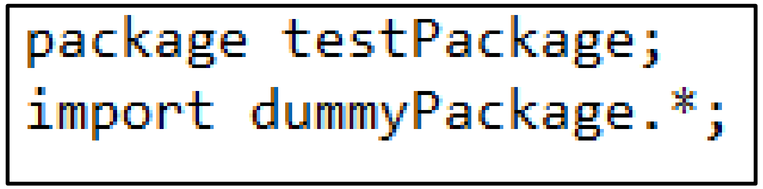
```
package p2;

public class C3 {
  cannot access C1;
  can access C2;
}
```

Packages in Java

- To avoid naming conflicts, packages are given names of the domain name of the company in reverse.

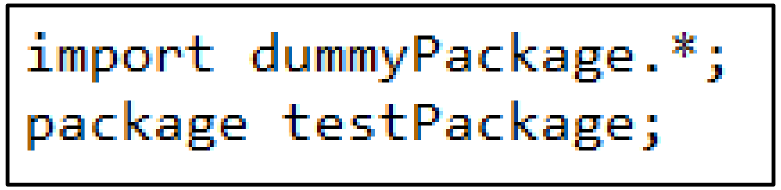
  Ex: com.tutorials, com.microsoft, com.infosys

- If a class is defined without the package statement, it is said to be placed in the default package.
- The statement for creating package must be written before any other import statements.



```
package testPackage;
import dummyPackage.*;
```

✓



```
import dummyPackage.*;
package testPackage;
```

X

END