Packages

Learning Objectives

- To learn package statement.
- To learn import packages.
- To learn set CLASSPATH from CMD prompt.
- To learn access specifiers.
- To learn interfaces.
- To apply interfaces
- . To understand extending interface.
- To study multiple inheritance issues.

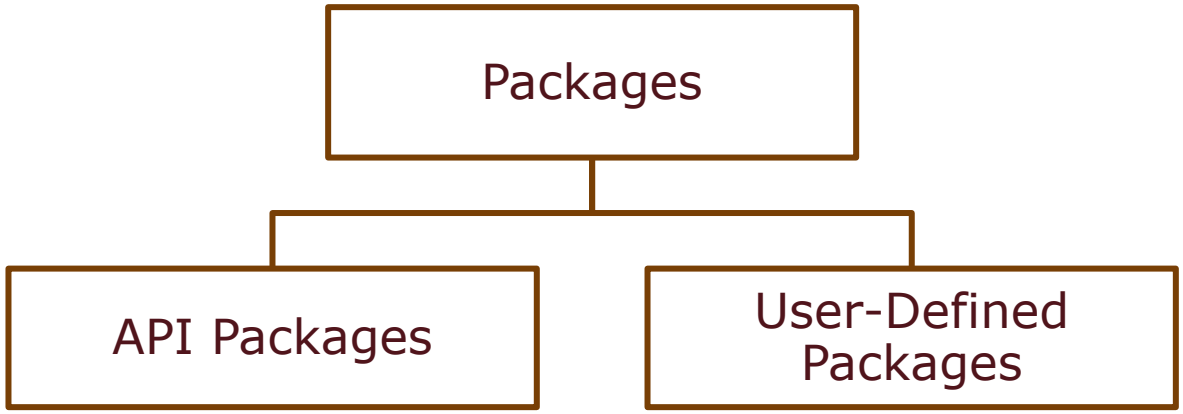
Software Packages

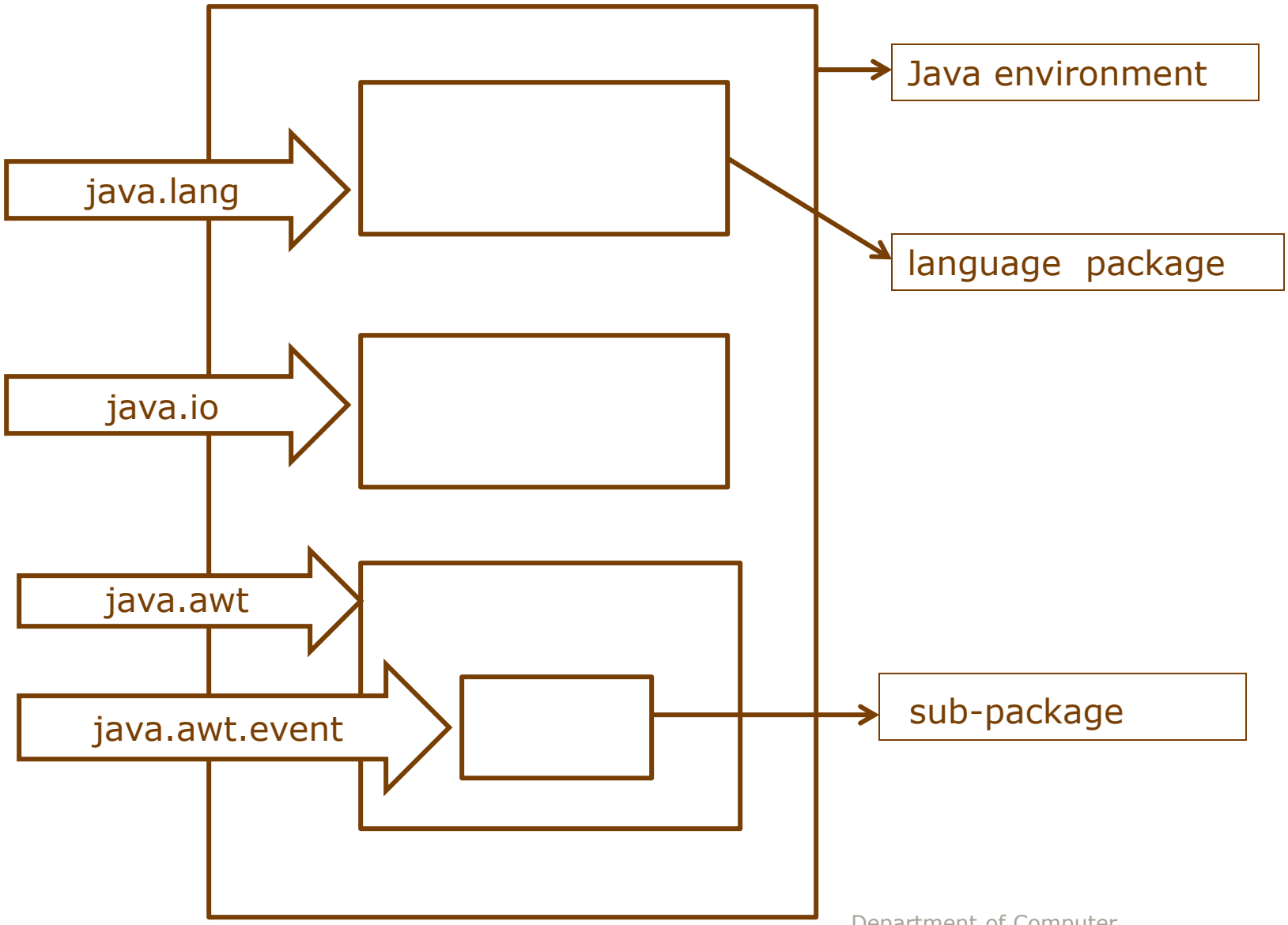
Packages enable you to organize the class files provided by Java.

Most software systems are large. It is common to group classes into packages to ease the management of the system.

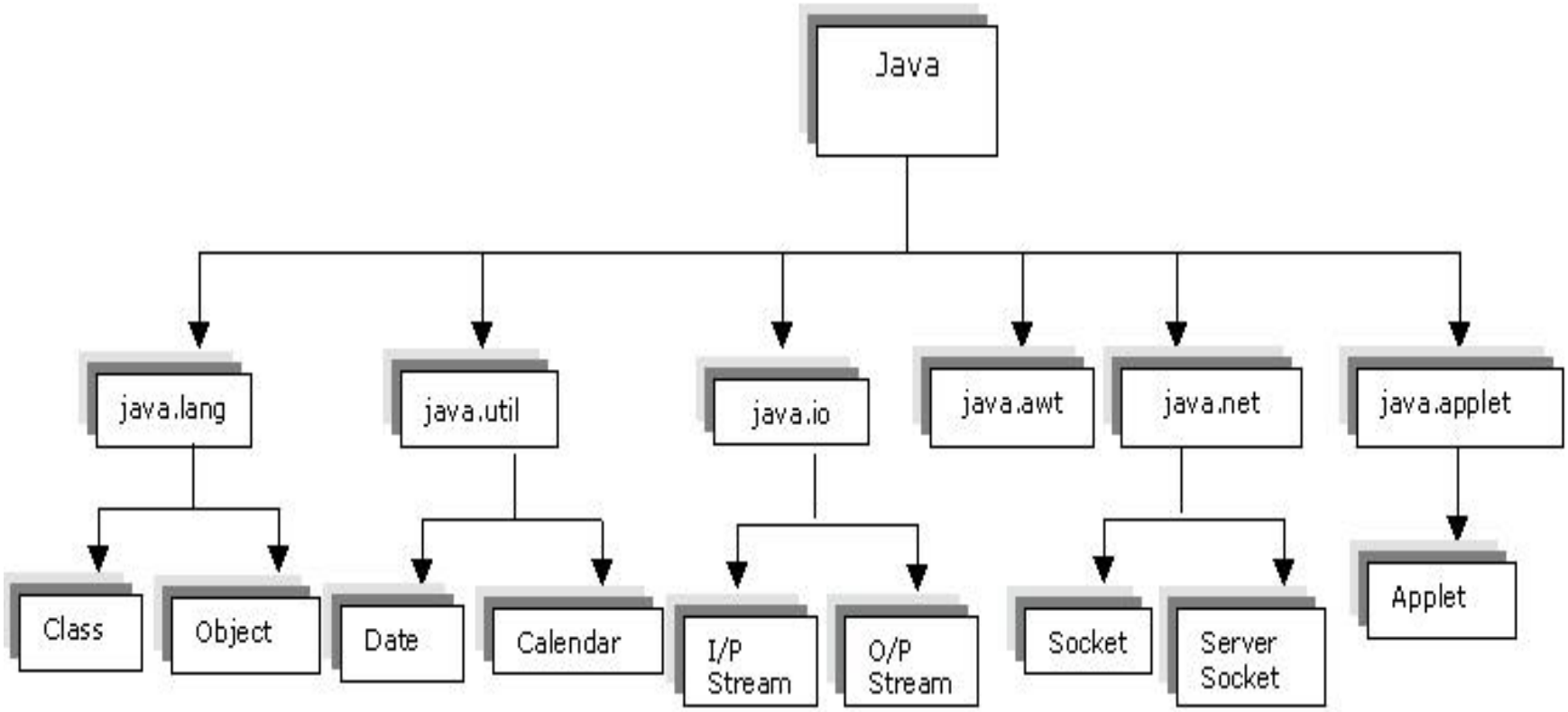
Packages can contain classes and sub-packages.

Categories




Java environment
java.lang
language package
java.io
java.awt
java.awt.event
sub-package

The hierarchy of the Java API packages:



The `package` Statement

The Java technology programming language provides the package statement as a way to group related classes.

Syntax

```
package <top_pkg_name> [.<sub_pkg_name>];
```

You can indicate that classes in a source file belong to a particular package by using the `package` statement.

for example

```
package  shipping.domain;
public  class  Vehicle {
        . . .
        . . .
}
```

Guidelines

- Specify the package declaration at the beginning of the source file.
- Only one package declaration per source file.
- If no package is declared, then the class is placed into the default (unnamed) package.
- Package names must be hierarchical and separated by dots.
- It is usual for the elements of the package name to be entirely lower case.
- If package declaration contains more than one class, then only one class can be public.

The `import` Statement

When you want to use packages, use the `import` statement to tell the compiler where to find the classes.

Syntax

```
import  <pkg_name>[.<sub_pkg_name>].<class_name>;
```

(or)

```
import  <pkg_name>[.<sub_pkg_name>].*;
```

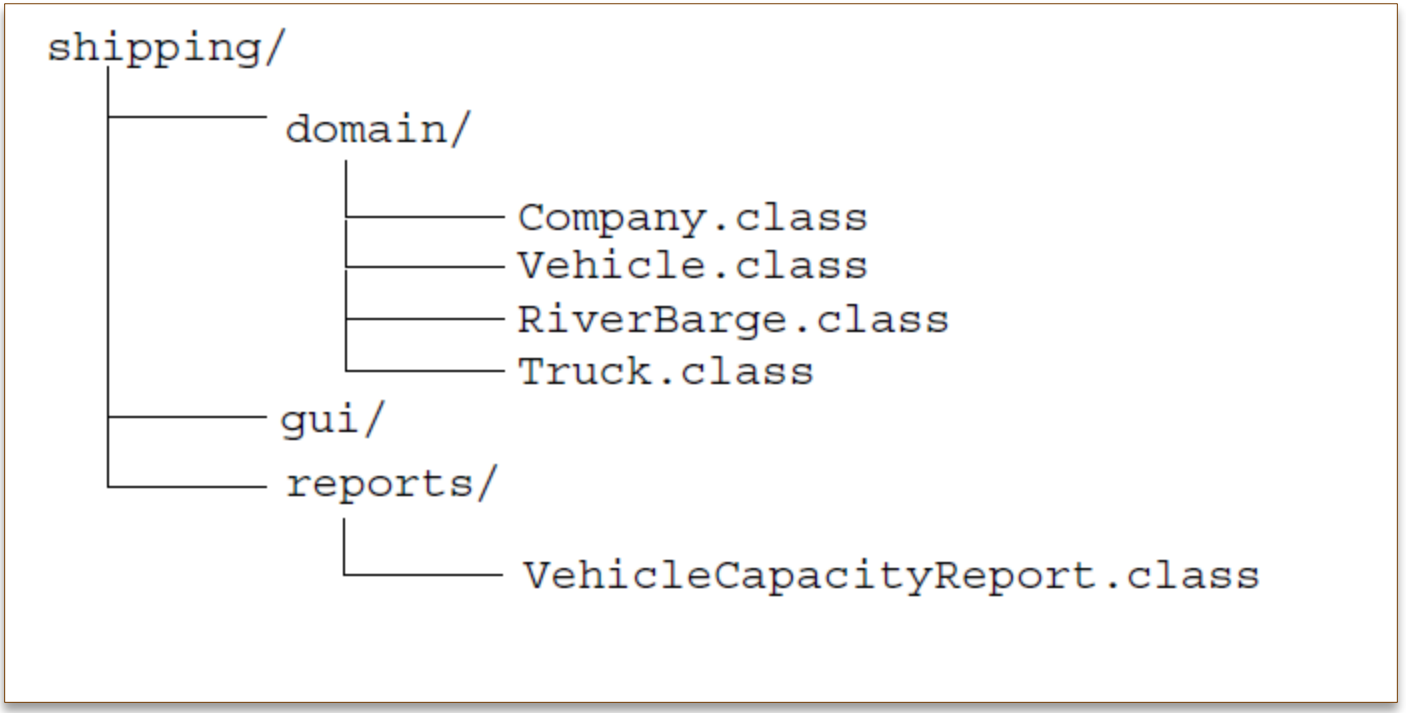
for example

```
import  shipping.domain.*;
import java.io.*;
import java.util.List;
public  class  VehicleReport {
        - - -
}
```

Directory Layout and Packages

Packages are stored in the directory tree containing the package name.

for example

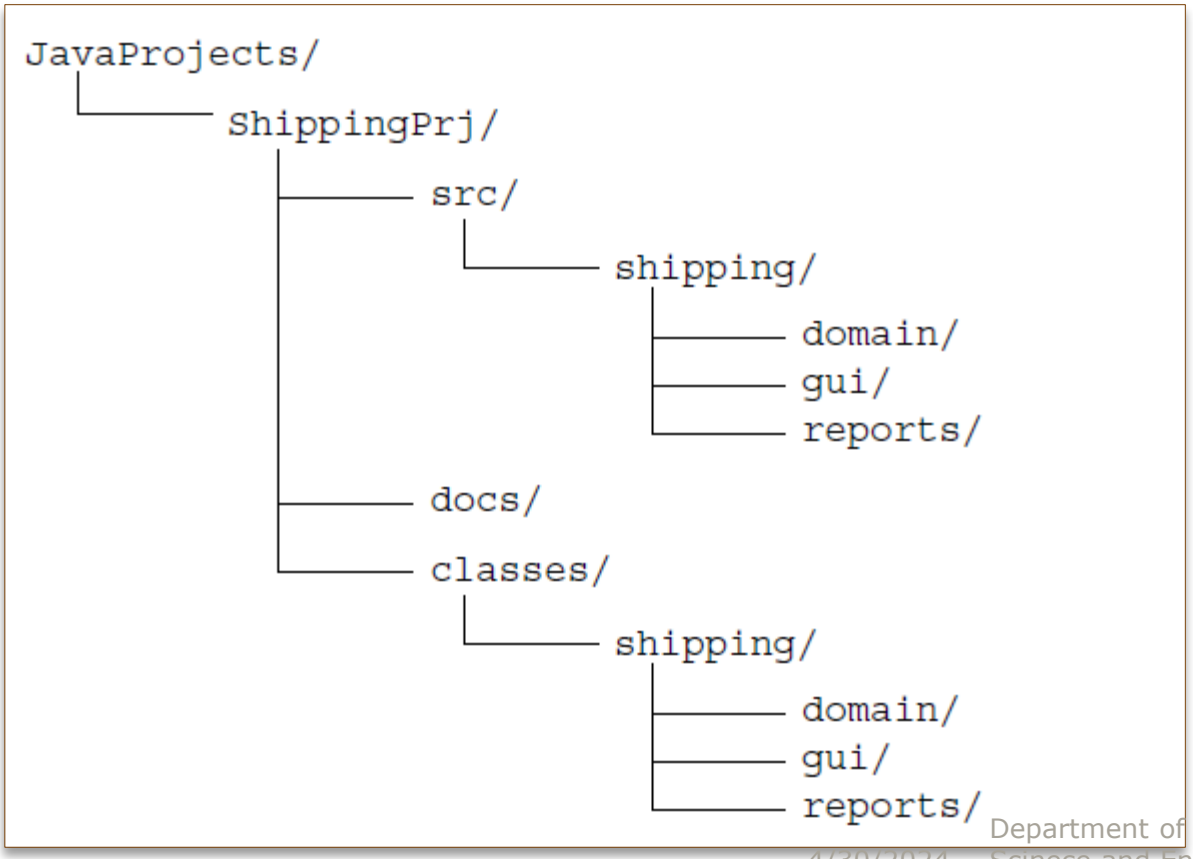


Development

It is common to be working on several development projects at once.

There are many ways to organize your development files.

for example



Compiling Using `-d` Option

Normally, the Java compiler places the class files in the same directory as the source files.

You can reroute the class files to another directory using the **`-d`** option of the `javac` command.

for example

To compile all the files within the `shipping.domain` package and have the compiled classes end up in their correct package directory under `shippingPrj/classes`

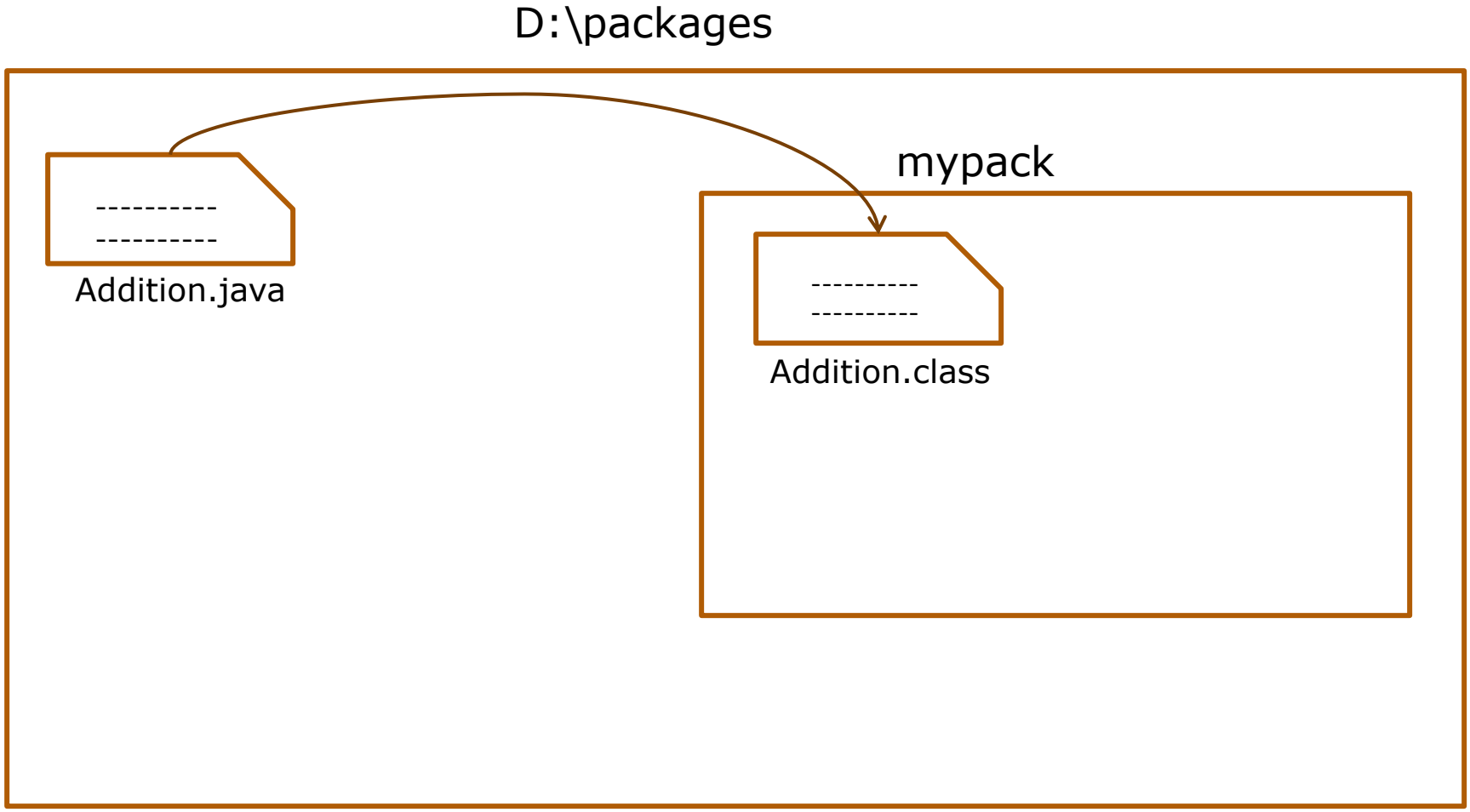
```
cd  JavaProjects/ShippingPrj/src
javac -d ../classes shipping/domain/*.java
```

Working with Packages

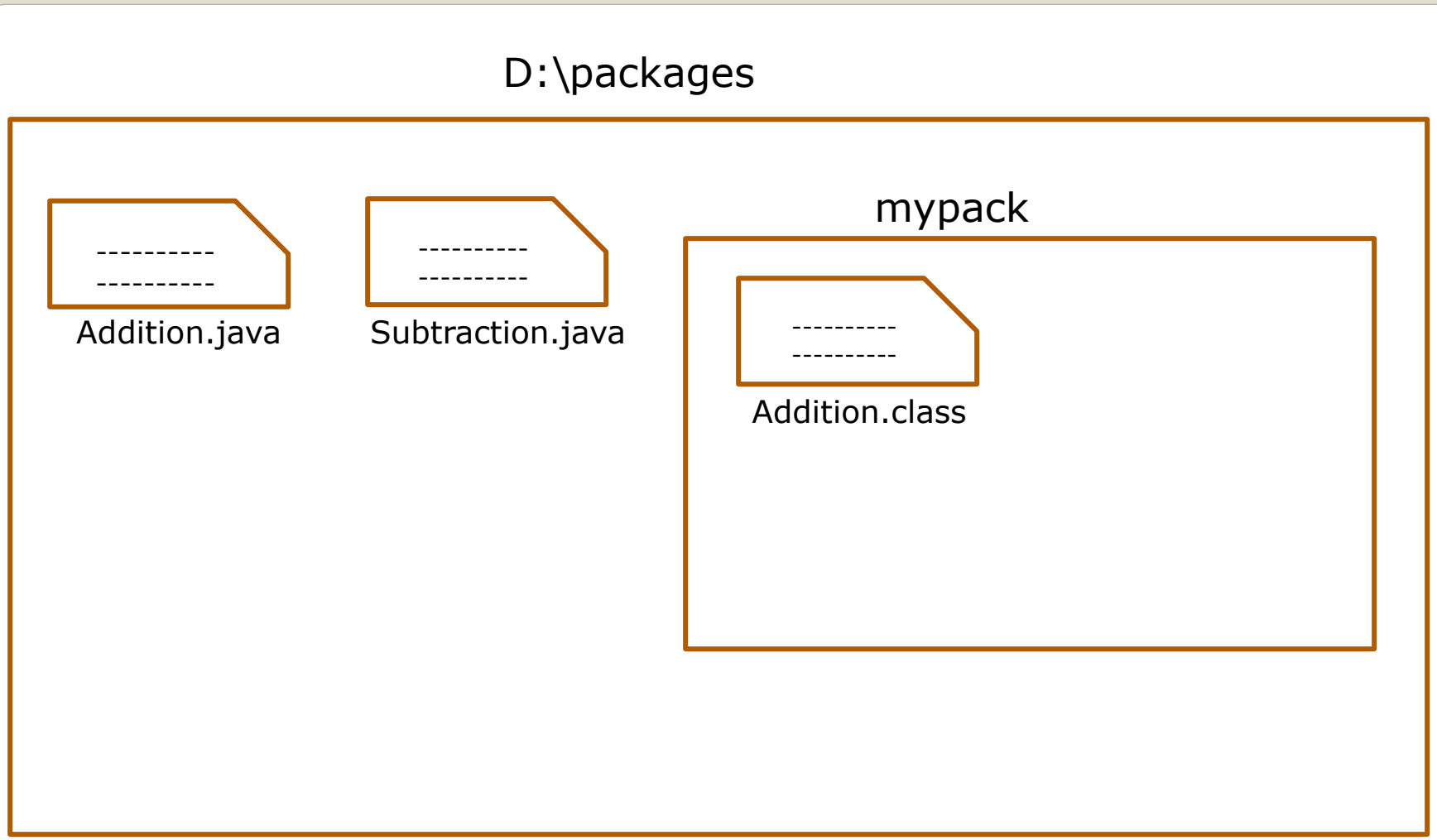
D:\packages

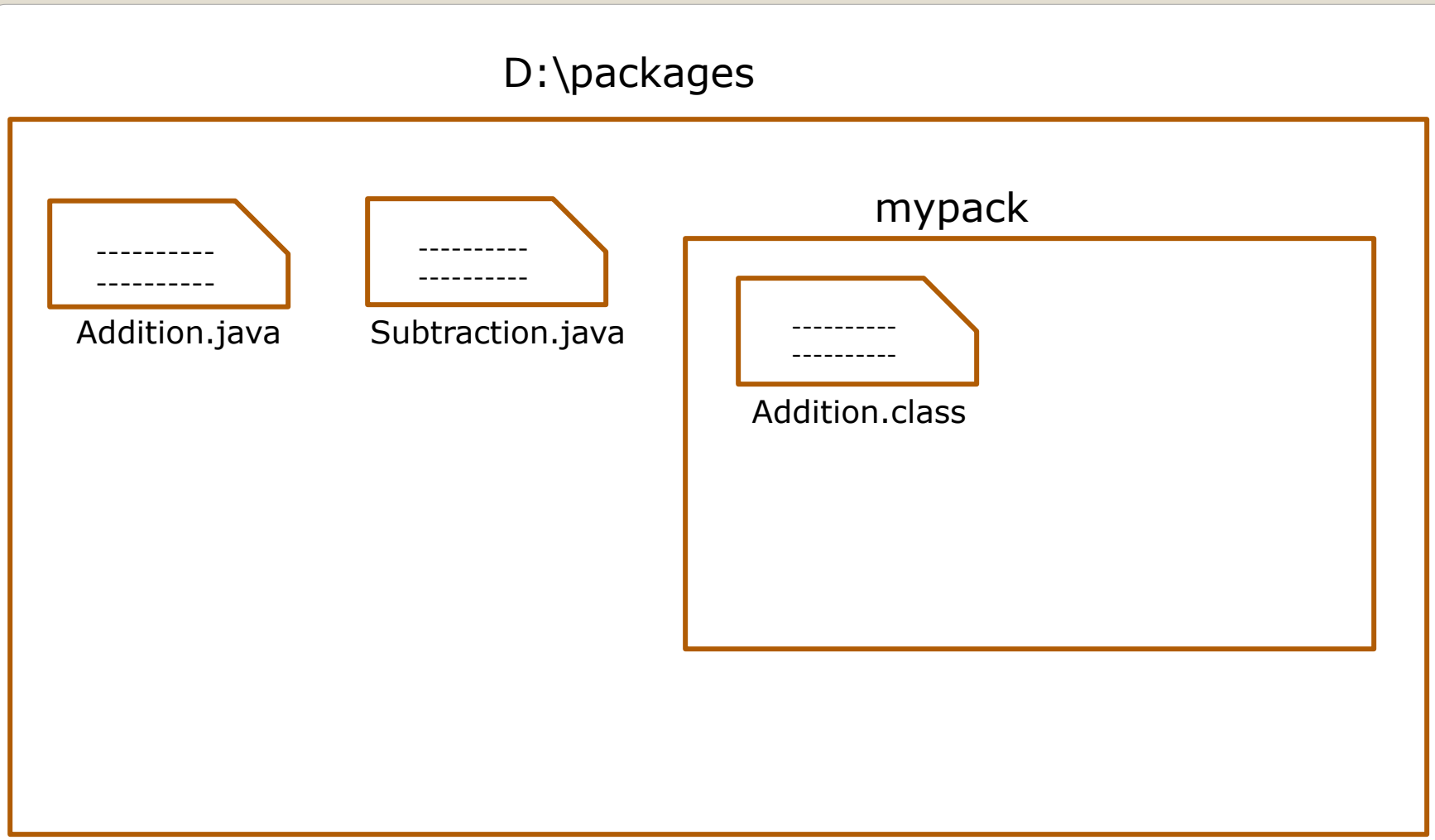
Addition.java

```
D:\packages>  javac –d . Addition.java
```

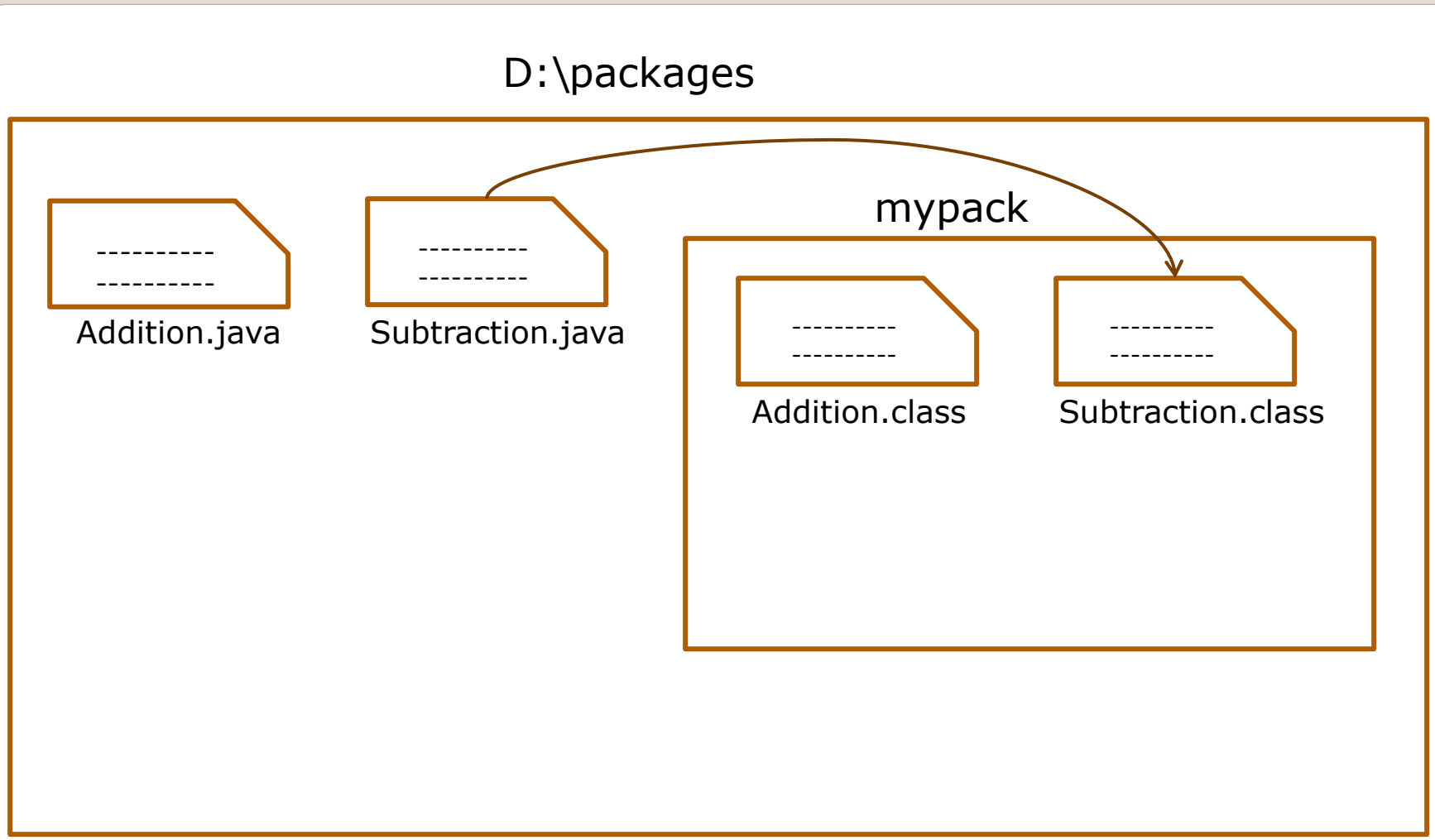



```
D:\packages>  javac –d . Addition.java
```


D:\packages
Addition.java
Subtraction.java
mypack
Addition.class



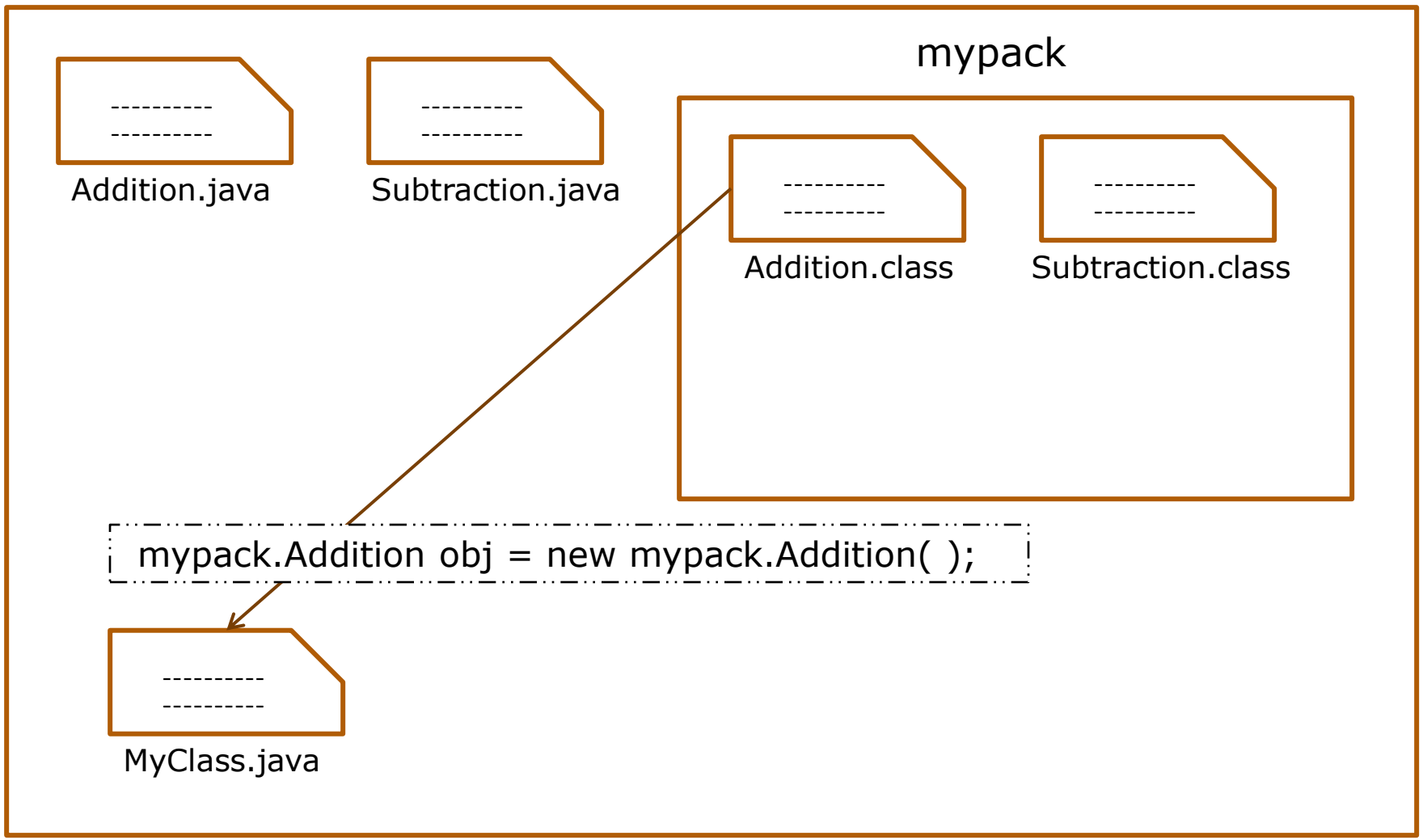

```
D:\packages>  javac –d . Subtraction.java
```

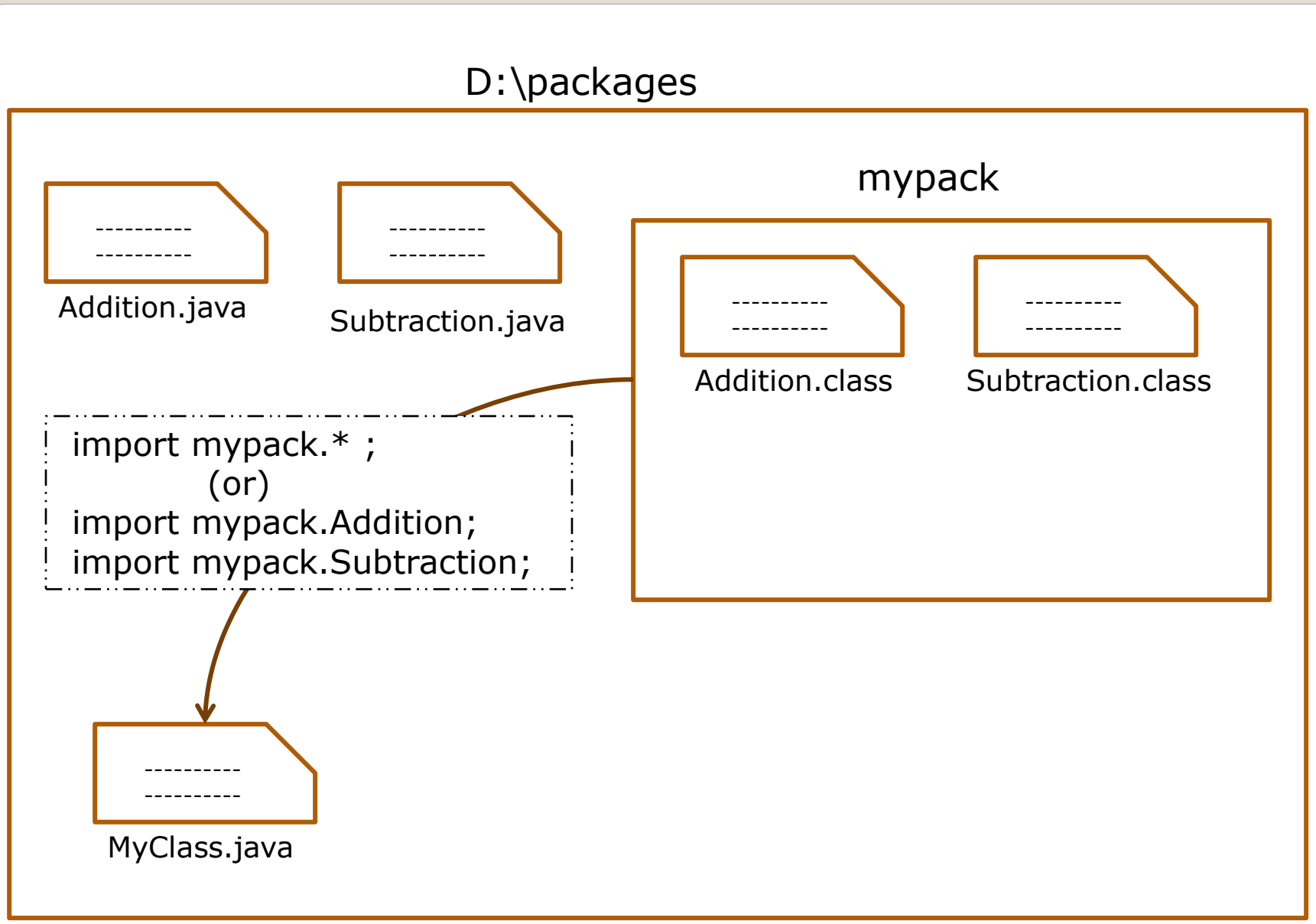


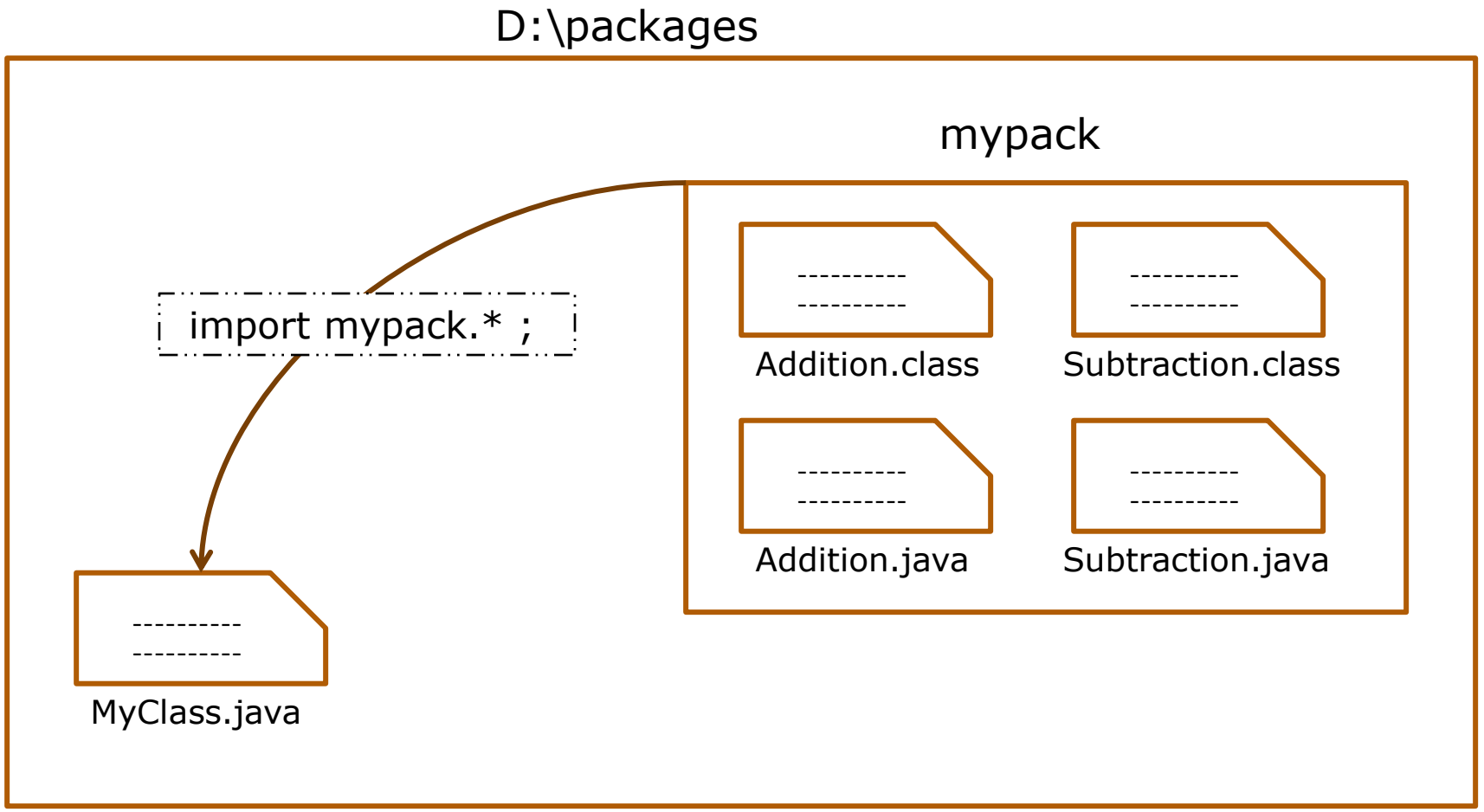

```
D:\packages>  javac –d . Subtraction.java
```

Ways to use the Classes of the package in your application

- Using fully qualified package name.
- By importing the package in your program
- By importing the package and as well as extending the class

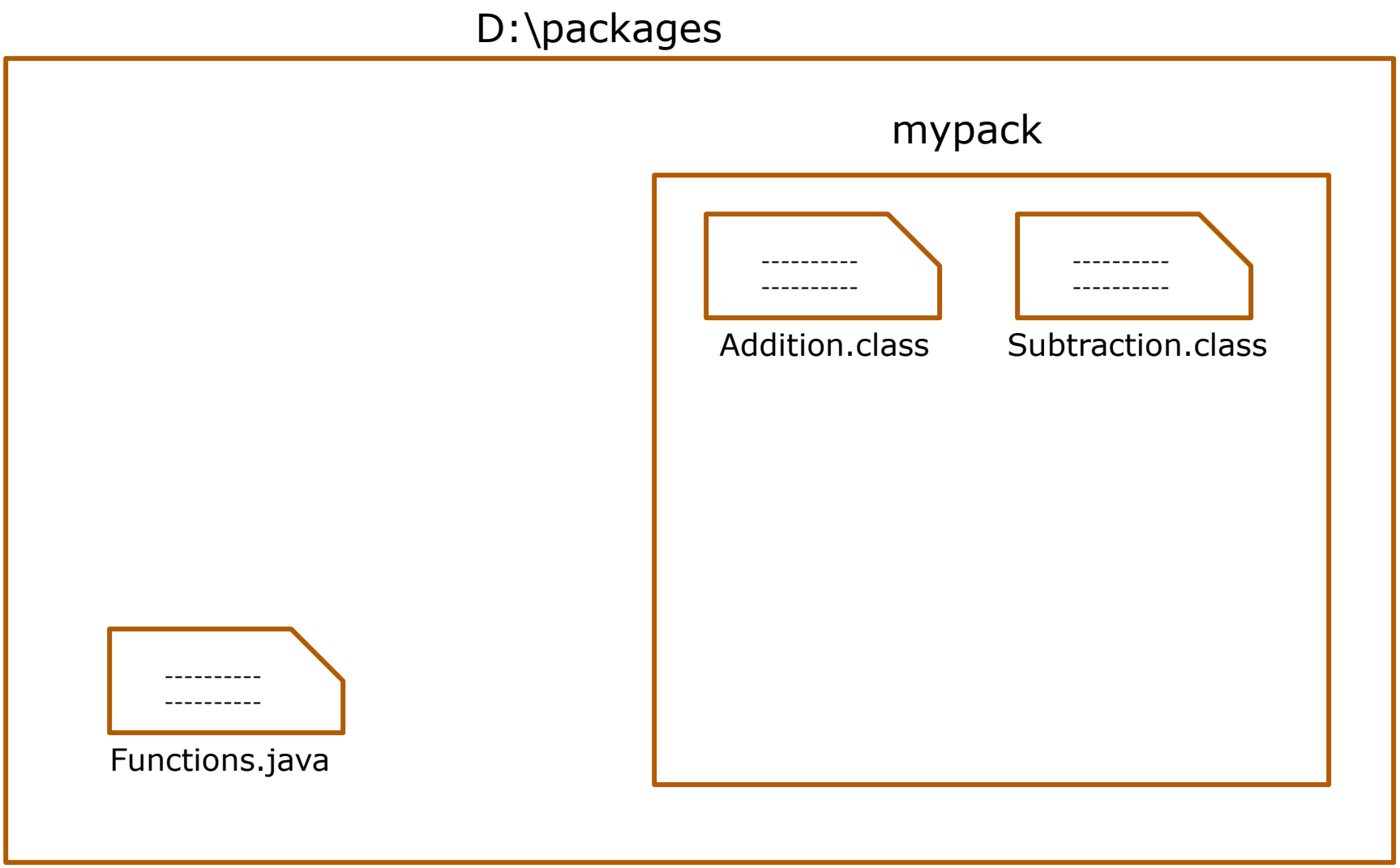

D:\packages
mypack
Addition.java
Subtraction.java
Addition.class
Subtraction.class
mypack.Addition obj = new mypack.Addition( );
MyClass.java


D:\packages
mypack
Addition.java
Subtraction.java
Addition.class
Subtraction.class
import mypack.* ;
(or)
import mypack.Addition;
import mypack.Subtraction;
MyClass.java

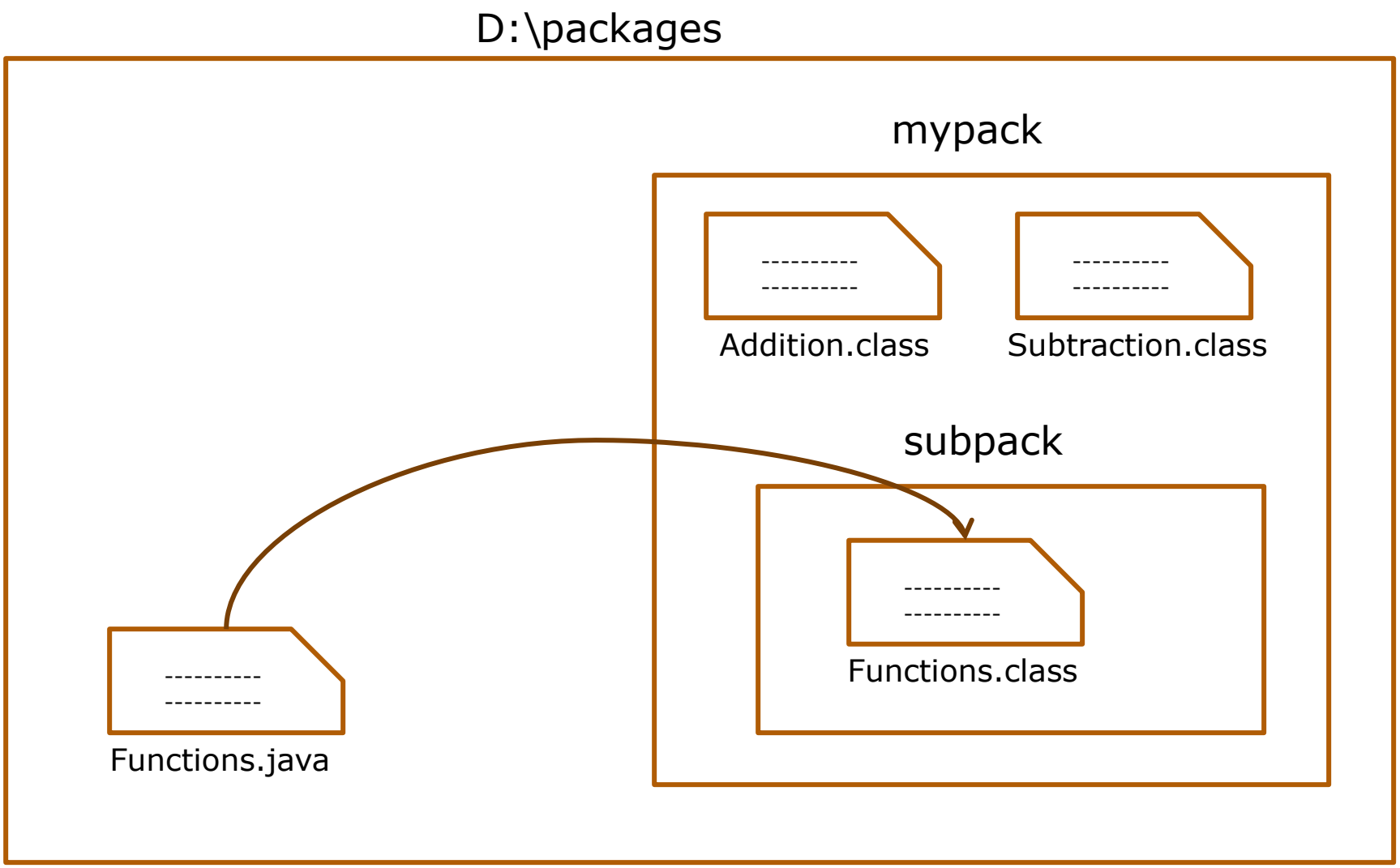


While using import mypack.* statement ensure that source files should not be in the current directory

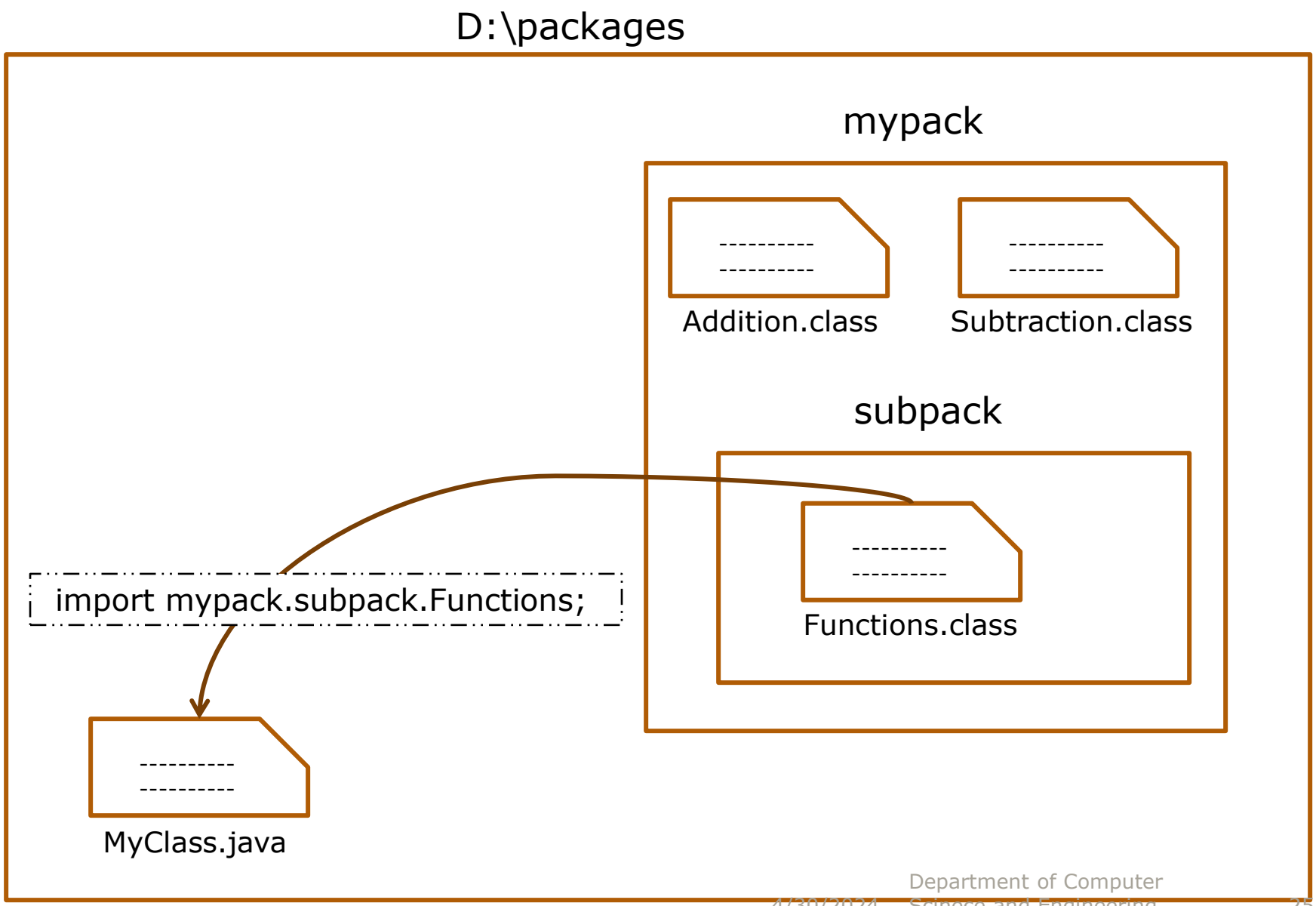
Sub-Packages

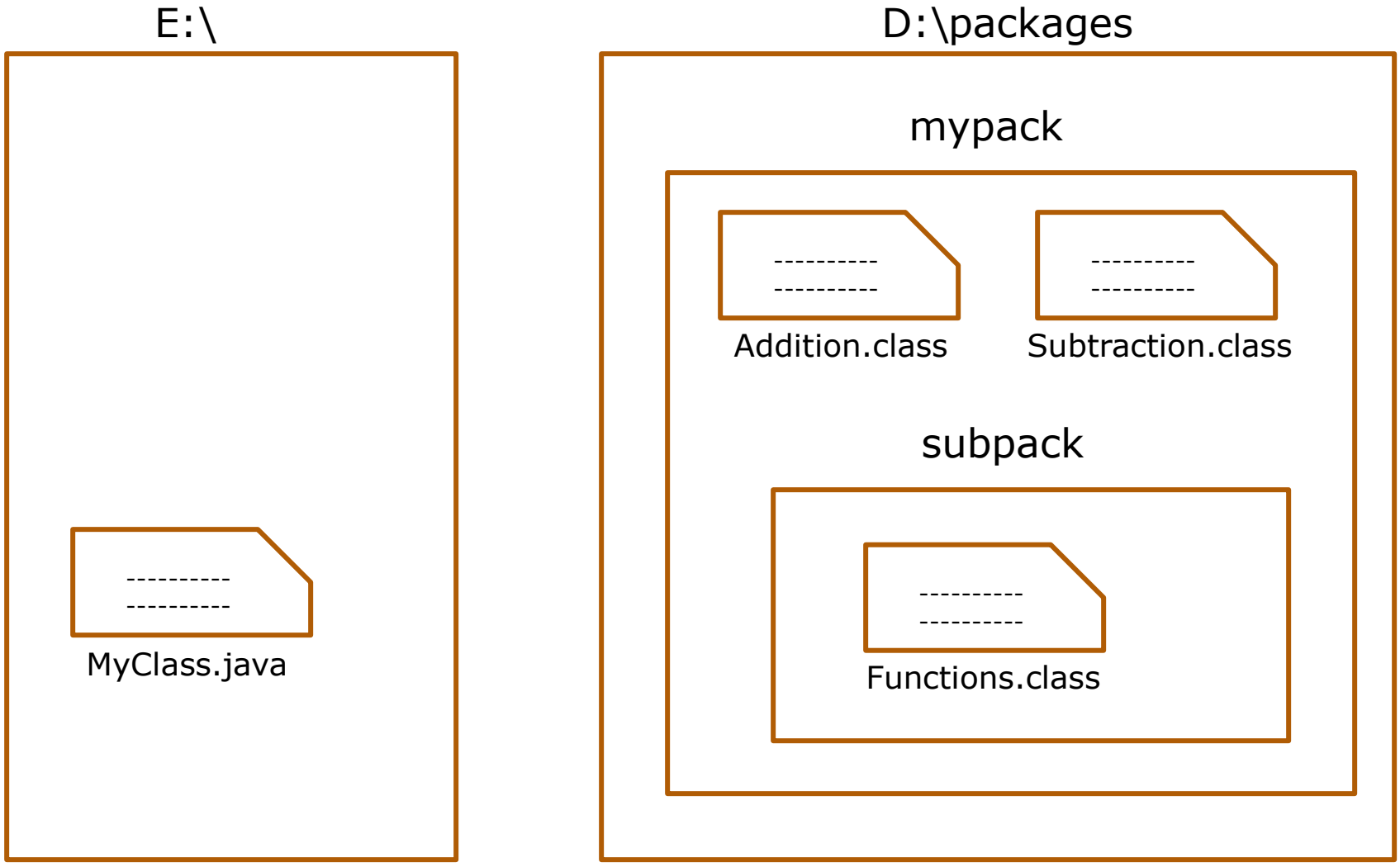


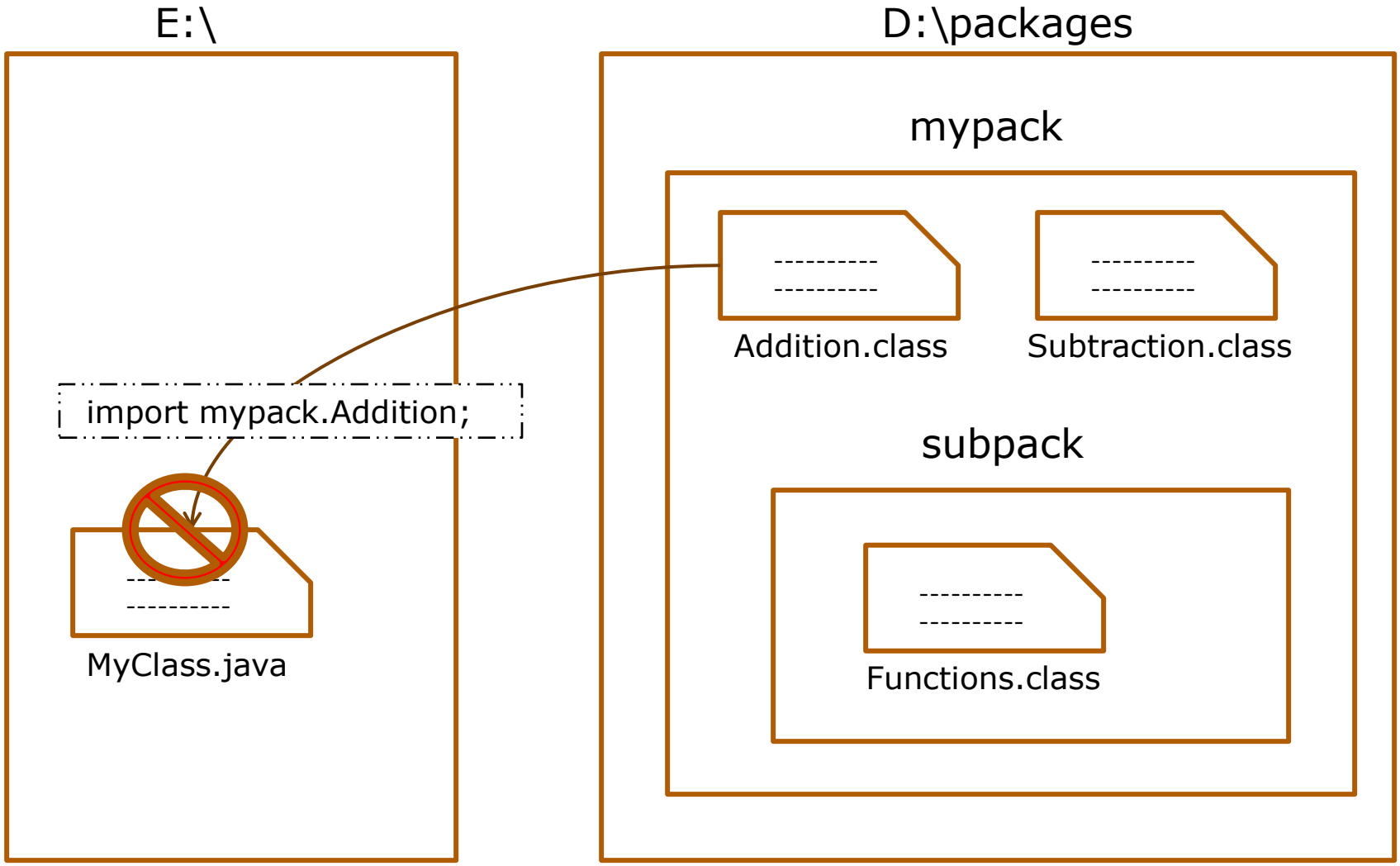

```
D:\packages>  javac –d . Functions.java
```



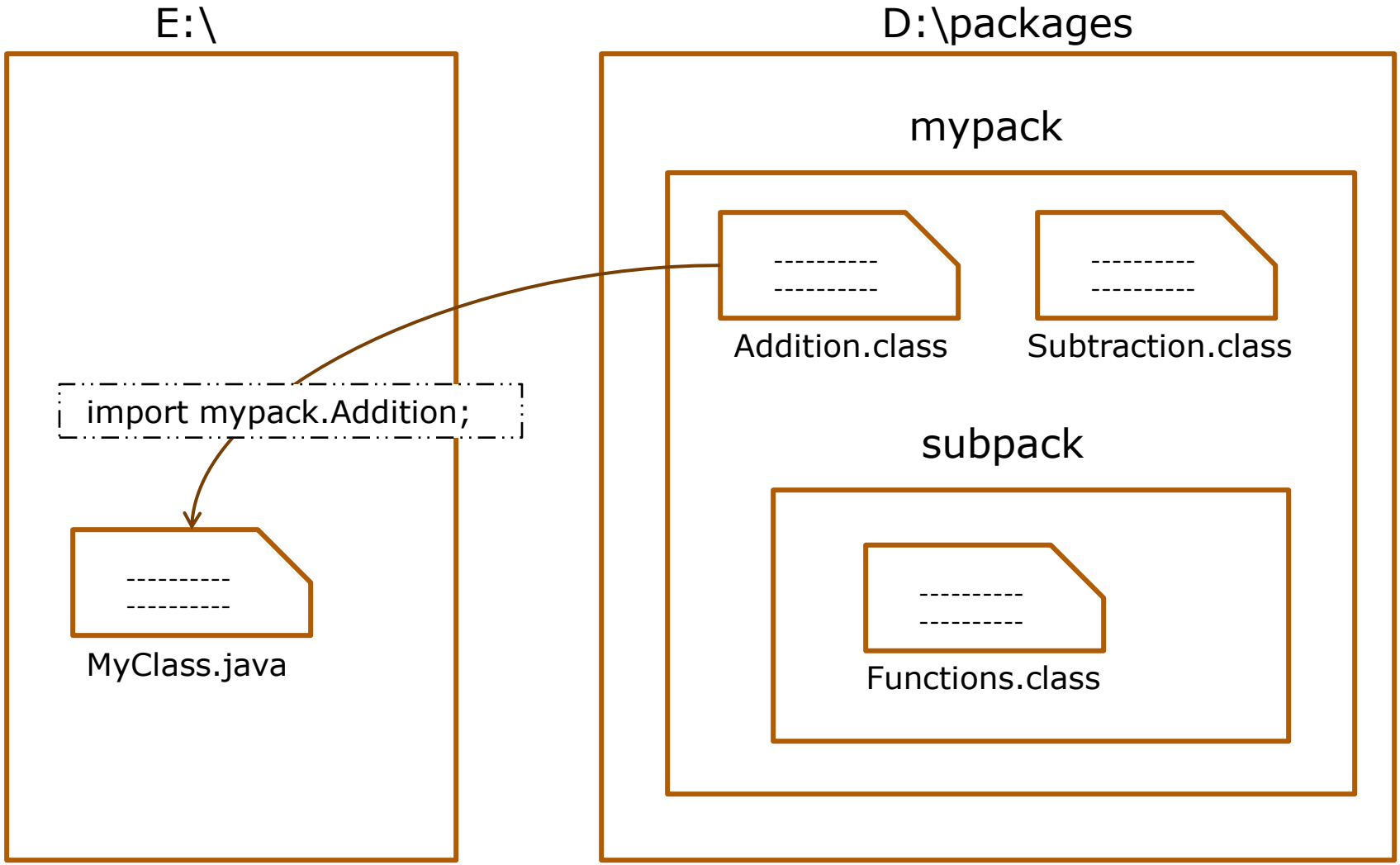

```
D:\packages>  javac –d . Functions.java
```


D:\packages
mypack
Addition.class
Subtraction.class
subpack
Functions.class
import mypack.subpack.Functions;
MyClass.java


E:\
MyClass.java
D:\packages
mypack
Addition.class
Subtraction.class
subpack
Functions.class



- If your program is in another drive, then you have to set the **CLASSPATH**
- The **CLASSPATH** is an environment variable that tells the Java compiler where to look for class files to import**.**

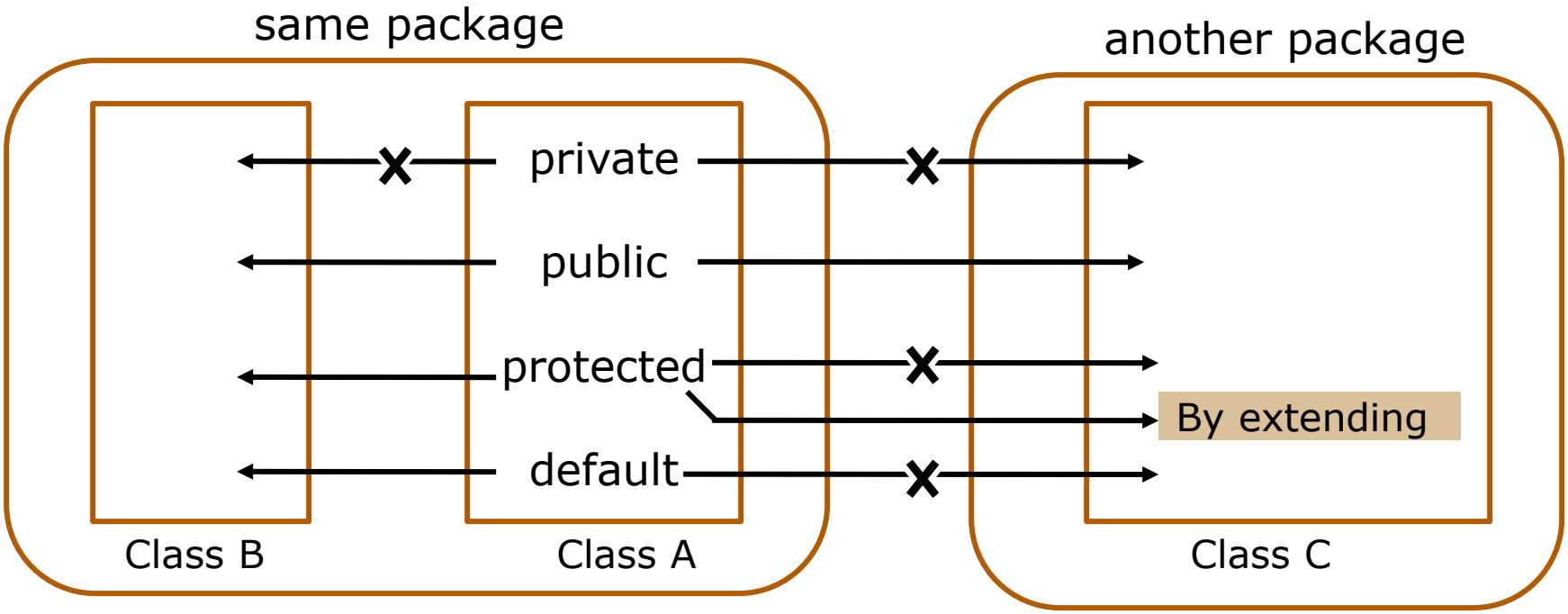


E :\> Set CLASSPATH = D:\Packages ; . ; %CLASSPATH% ;

Access Specifiers in Java:

- There are four access specifiers in java :
  private, public, protected and default.

- Let's see how these Access specifiers protects the data



	Private	Default	Protected	Public
Same class	✓	✓	✓	✓
Same package Sub – class	✗	✓	✓	✓
Same package Non – subclass	✗	✓	✓	✓
Different package Sub – class	✗	✗	✓	✓
Different package Non – subclass	✗	✗	✗	✓

Overview of Interface:

- An interface contains only abstract methods which are incomplete methods.
- It is not possible to create an object.
- We can create a separate class where we can implement all the methods.
- These classes are called implementation classes.