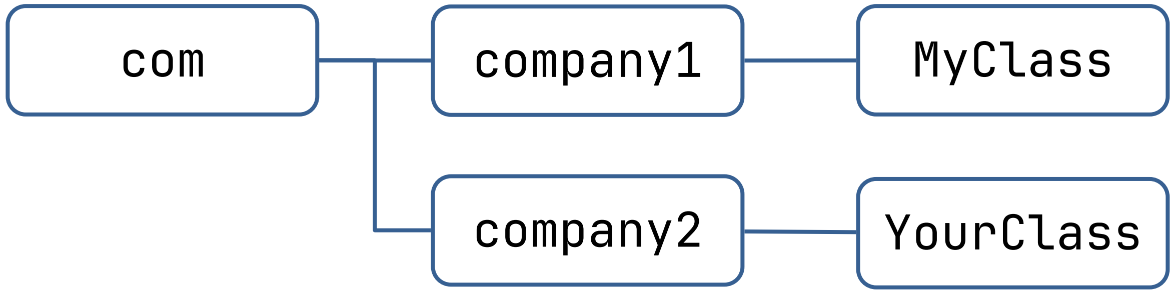
05_1 Packages and Access Modifiers

Object-Oriented Programming

Package

- Set of classes
- Physical form of a package: a folder in the file system.
- Identifier that makes the class unique
  - The classes with the same name in different packages are recognized as different classes
- Class name = parent_package . child_package . class_name

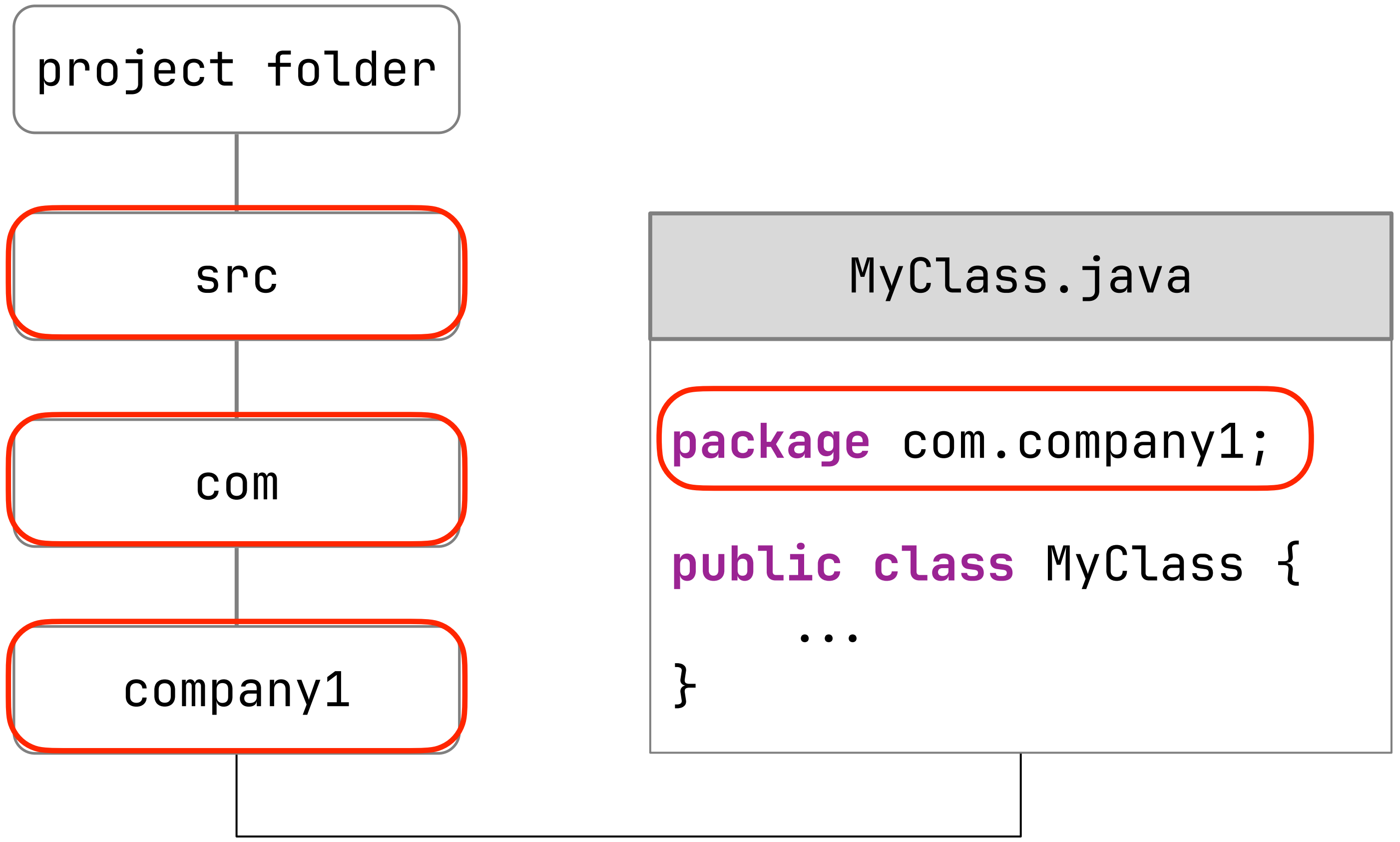
- Ex)



com.company1.MyClass

com.company2.YourClass

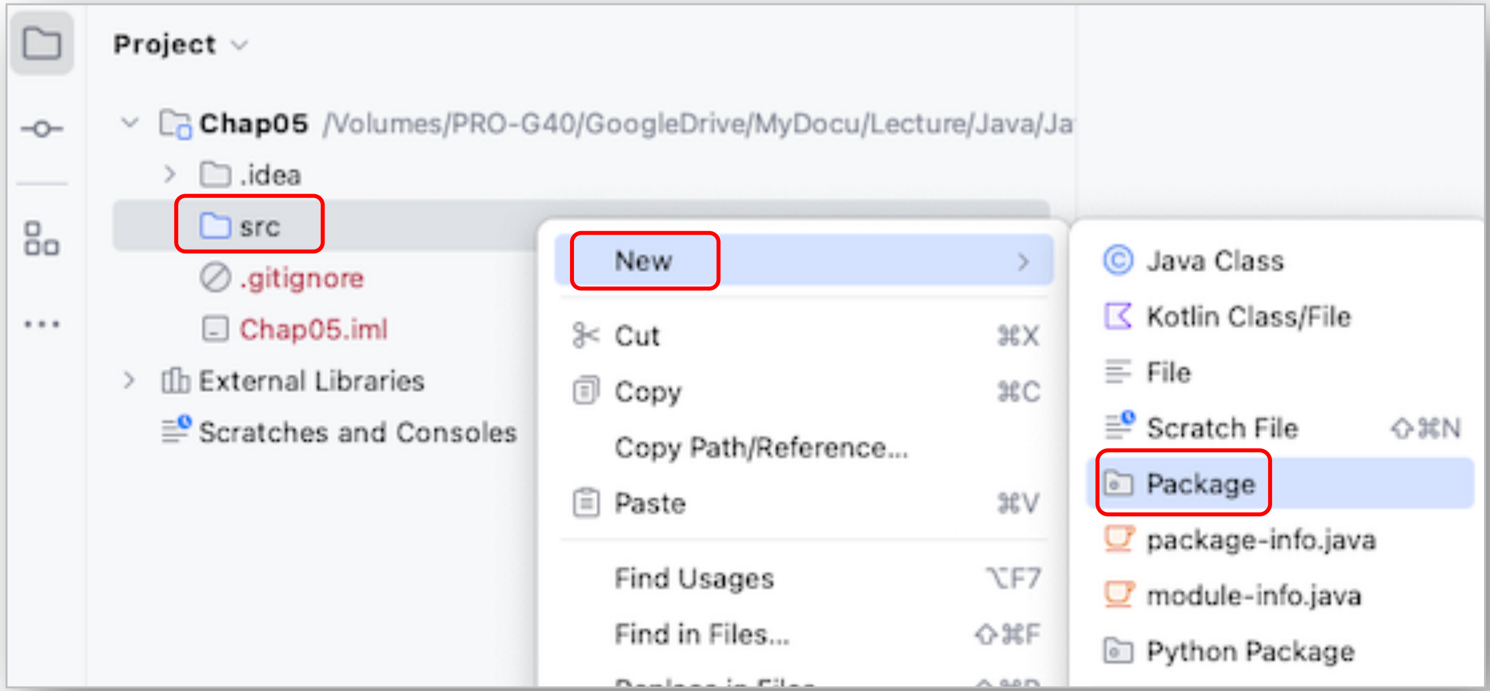
Java Class in a Package

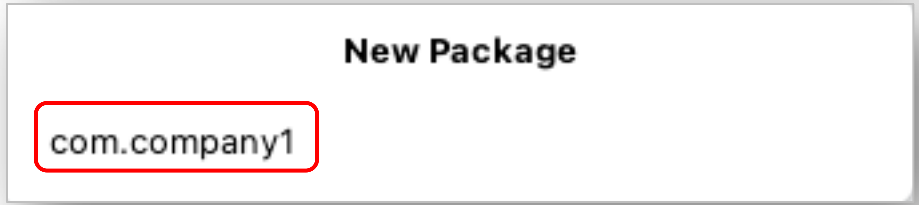


```
package com.company1;

public class MyClass {
    ...
}
```

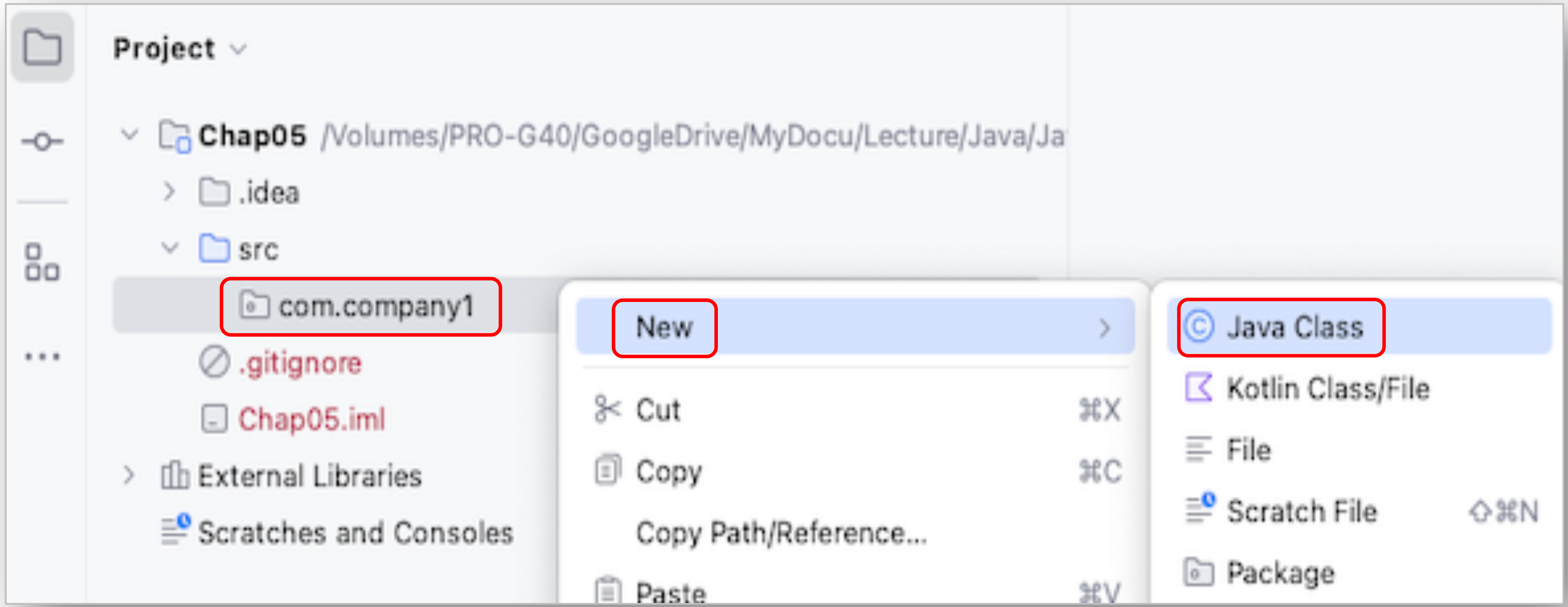
Creating Package in IntelliJ IDEA (1/6)



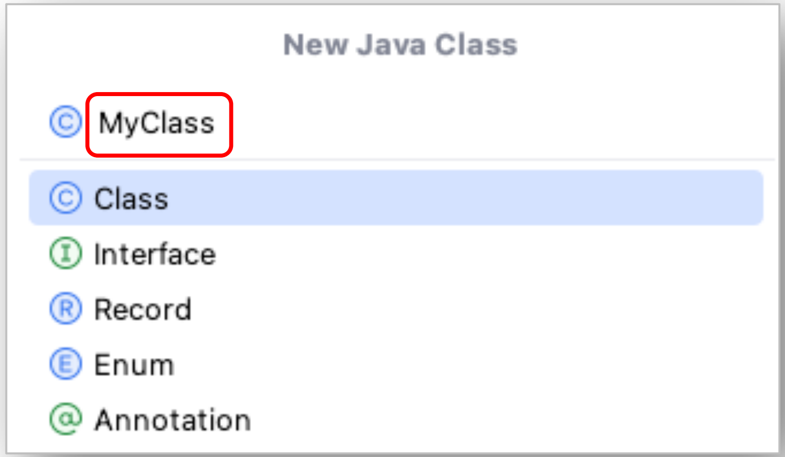


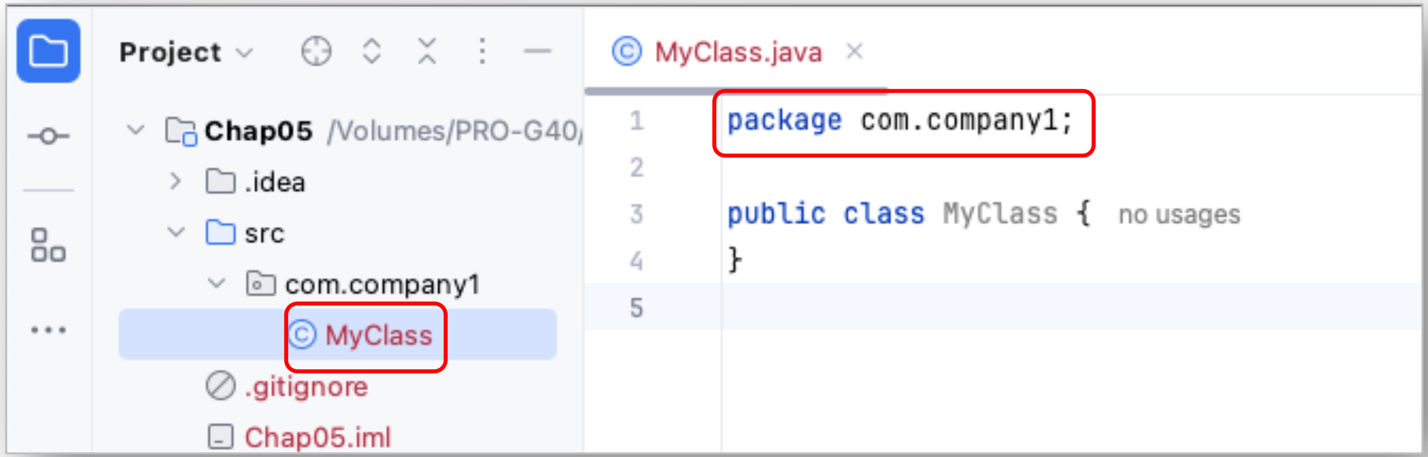
com.company1

Creating Package in IntelliJ IDEA (2/6)



Creating Package in IntelliJ IDEA (3/6)





```
package com.company1;

public class MyClass {  no usages
}

```

import Statement

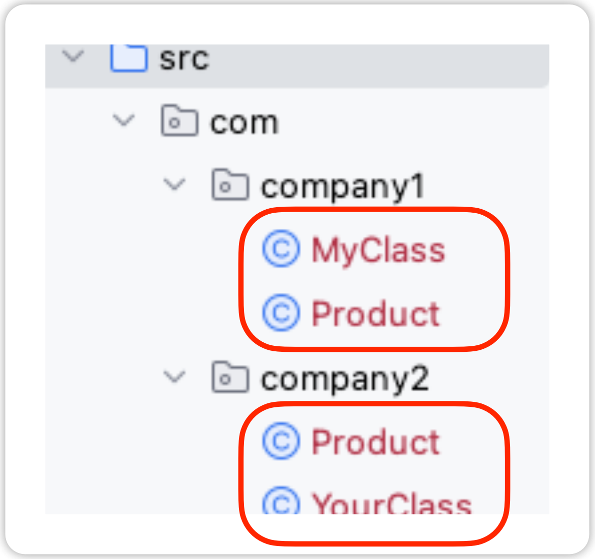
- If a class belongs to another package, specify it to be found
- ex)

```
// import all classes in package ‘com.company1’
import com.company1.*;

// import only ‘YourClass’ class in package ‘com.company2’
import com.company2.YourClass;
```

Example: PackageTest.java

- Assume we have the com.company1 and com.company2 packages:



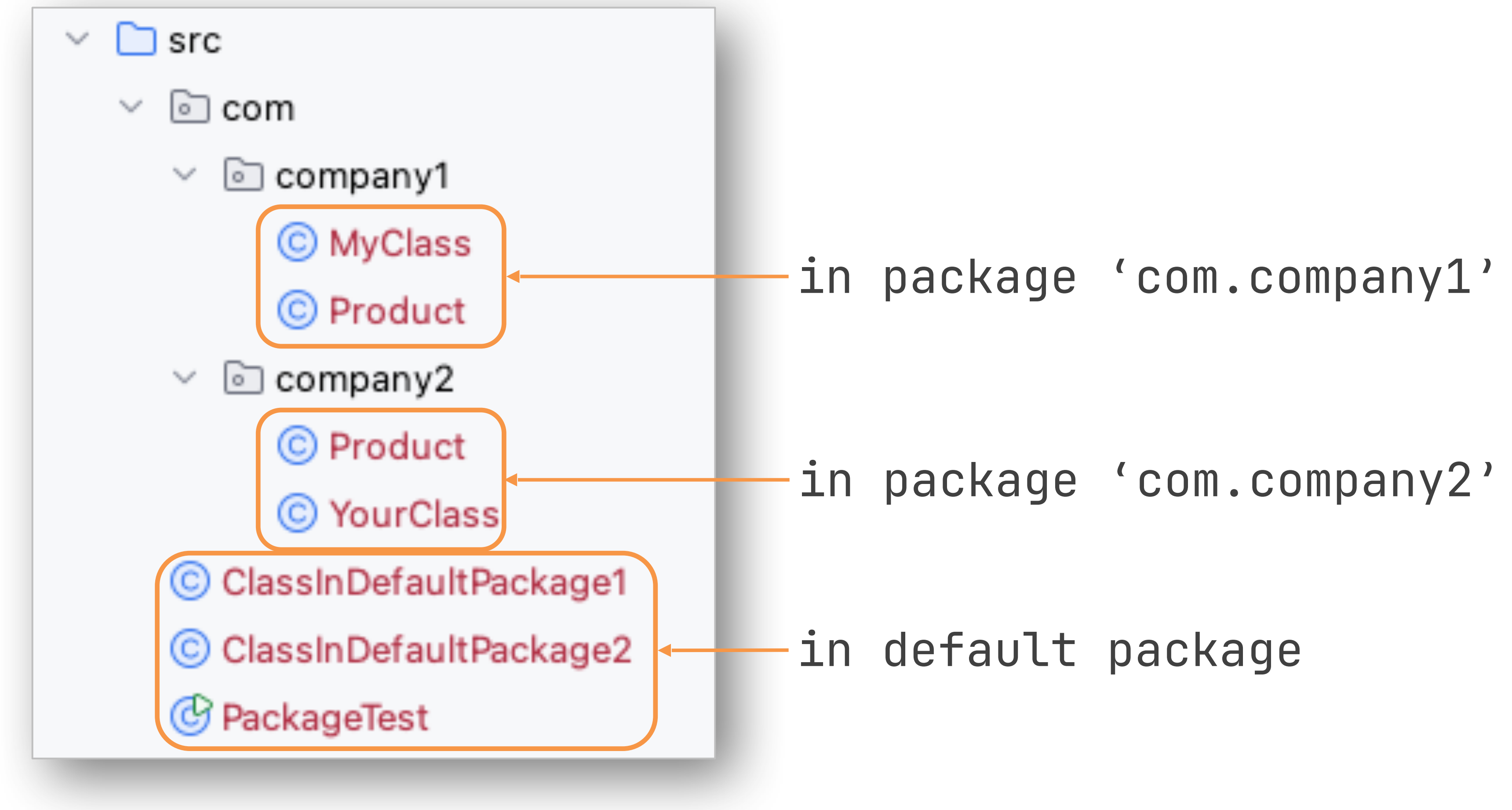
```java
import com.company1.*;
import com.company2.*;

public class PackageTest {
    public static void main(String[] args) {
        MyClass mClass = new MyClass();
        YourClass yClass = new YourClass();
        com.company1.Product p1 = new com.company1.Product();
        com.company2.Product p2 = new com.company2.Product();
    }
}
```

- class ‘Product’ exists in both packages, so, they should be distinguished by full name including the package name.

Default Package

- If no package is specified, the class belongs to the 'default package'.



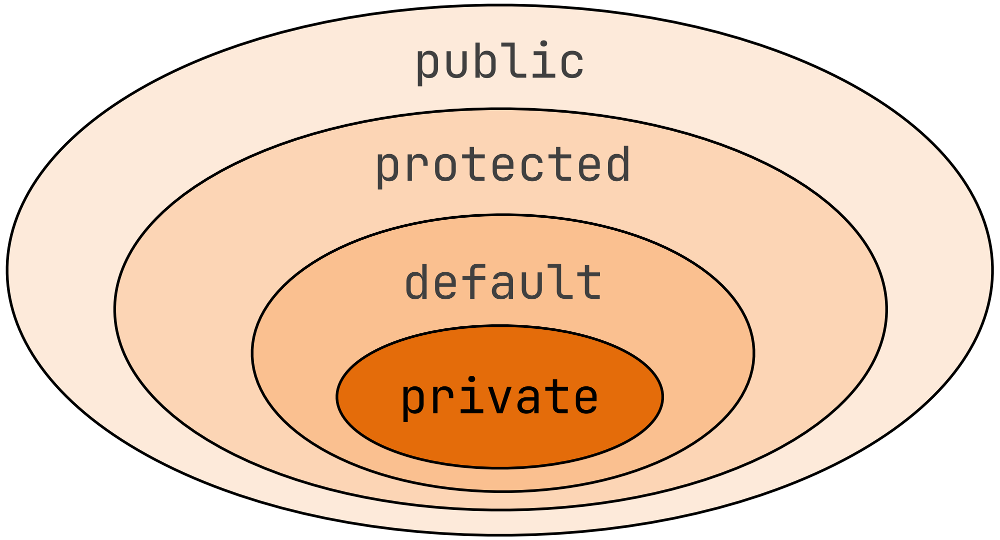
Built-in Packages

- java.lang
  - Containing the basic classes
  - Can be used without importing
  - ex) Object, String, Math, System, Thread, ...
- java.util
  - Containing data structures and utility classes
  - ex) Scanner, ArrayList, HashMap, HashSet, Date, Calendar, Collections, ...
- java.io
  - Provides input and output functionality
  - ex) File, InputStream, OutputStream, Reader, Writer, ...

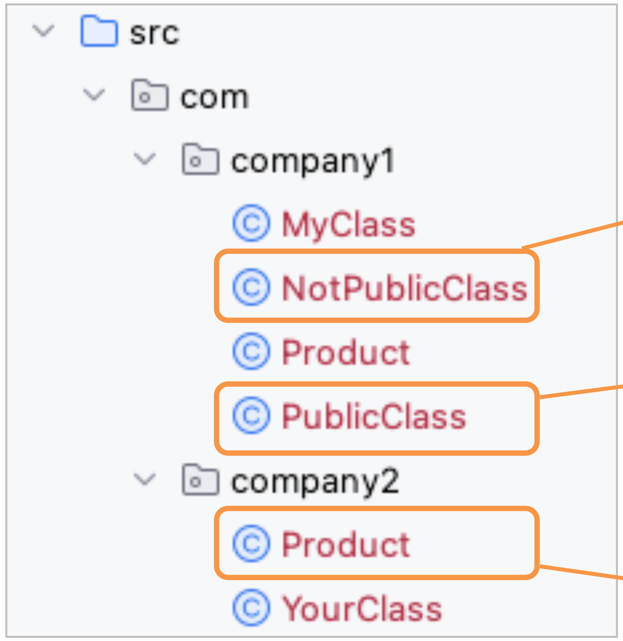
...

Access Modifier

- public
  - can be accessed from outside of the same package (anywhere)
- protected
  - can be accessed inside of the same package or from within the child class
- private
  - can be accessed only from inside of the same class
- default (package)
  - can be accessed inside of the same package



Example: Default (Package) Access



```java
package com.company1;

class NotPublicClass { }
```

```java
package com.company1;

public class PublicClass { }
```

```java
package com.company2;

import com.company1.*;

public class Product {
    public static void main(String[] args) {
        NotPublicClass nps = new NotPublicClass(); // compile error!
        PublicClass ps = new PublicClass();
    }
}
```

public and private Fields (1/2)

```java
public class AClass {
    public int x;
    private int y;
    int z;

    public AClass() {  x = 2;  y = 3;  z = 4; }
    public void publicMethod() {
        System.out.println("AClass:publicMethod " + (x + y + z));
    }
    private void privateMethod() {
        System.out.println("AClass:privateMethod");
    }
    void packageMethod() {
        System.out.println("AClass:packageMethod");
        publicMethod();
        privateMethod();
    }
}
```

public and private Fields (2/2)

```java
public class AClassTest {

    public static void main(String[] args) {

        AClass ac = new AClass();
        System.out.println("ac.x = " + ac.x);
        //System.out.println("ac.y = " + ac.y); // compile error!
        System.out.println("ac.z = " + ac.z);
        ac.publicMethod();
        //ac.privateMethod();  // compile error!
        ac.packageMethod();

    }
}
```

Recommendation – Information Hiding

- In terms of hiding information, it is recommended that all members within the class be private.
- To prevent misbehavior from outside the class, it is recommended to minimize the number of public access.

Accessor and Mutator

- Accessor (Getter)
  - A method to read the value of private variable from outside of the class
  - ex) public int getX();   public String getStr(); ...

- Mutator (Setter)
  - A method to write the value to the private variable from outside of the class
  - ex) void setX(int);   void setStr(String); ...
  - If no package is specified, the Java class will belong to the 'default package'.
  - Test for the conditions that a private instance variable should have (e.g. scope) before assigning it.

Example: Accessor and Mutator (1/3)

```java
public class BClass {
    private int x;
    private String str;

    public BClass(int x, String str) { // constructor
        this.x = x;
        this.str = new String(str);
    }

    public int getX() { // accessor
        return x;
    }

    public String getStr() { // accessor
        return str;
    }
```

Example: Accessor and Mutator (2/3)

```
    public void setX(int x) {  // mutator
        this.x = x;
    }

    public void setStr(String str) {  // mutator
        this.str = new String(str);
    }

}
```

Example: Accessor and Mutator (3/3)

```java
public class BClassTest {
    public static void main(String[] args) {

        BClass b = new BClass(3, "Korea");
        System.out.println("b.x=" + b.getX() + "  b.str=" + b.getStr());

        b.setX(5);
        b.setStr("Seoul");
        System.out.println("b.x=" + b.getX() + "  b.str=" + b.getStr());
    }
}
```

```
OUTPUT:
b.x=3  b.str=Korea
b.x=5  b.str=Seoul
```