05_1 Packages and Access Modifiers

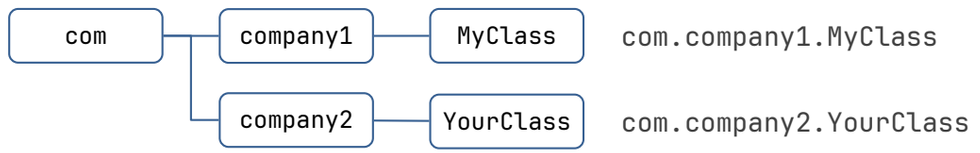
Object-Oriented Programming

Package와 Access Modifier에 대해 강의하겠습니다.

Package

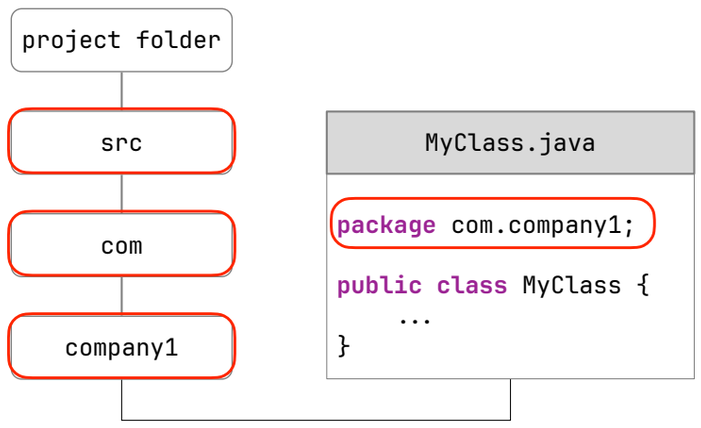
- Set of classes
- Physical form of a package: a folder in the file system.
- Identifier that makes the class unique
  - The classes with the same name in different packages are recognized as different classes
- Class name = parent_package . child_package . class_name

- Ex)



페이지 2
package는 class들이 모여있는 것으로
같은 package에 포함된 class들은
같은 폴더에 들어 있습니다.
한 package 안에는 같은 이름의 class가 존재할 수 없습니다.
따라서 package들로 묶어 놓는 경우
혹시 같은 이름을 가지는 class들이라도
다른 package에 속해 있다면
package의 이름으로 구별될 수 있는 것입니다.
package의 이름까지 고려한 class의 이름은
parent package의 이름, dot, child package의 이름, dot, class 이름으로
확장되기 때문에 구별이 가능해 집니다.
이 예에서 MyClass는 com, dot, company1 package에 포함되어 있고,
YourClass는 com, dot, company2 package에 포함되어 있습니다.

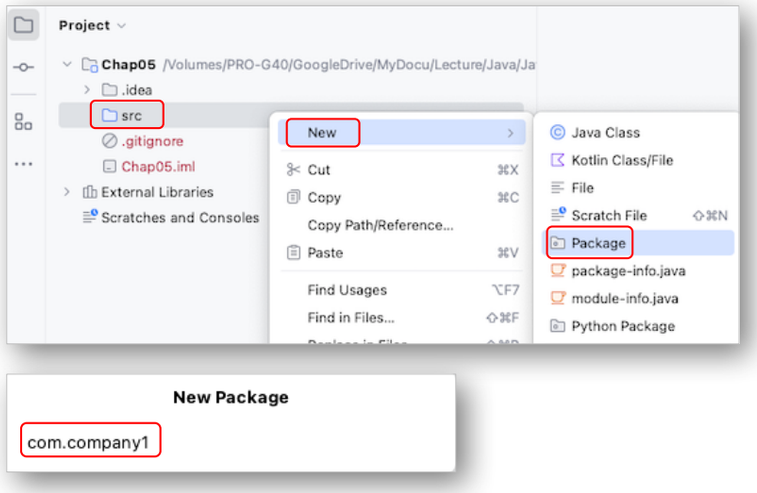
Java Class in a Package



페이지 3

Java class가 어떤 특정한 package에 포함되도록 하려면
프로젝트 폴더 아래 src 폴더 아래
parent package 폴더 아래
child package 폴더 아래
Java source file, 예를 들면, MyClass dot java가 있어야 합니다.
Java source file의 맨 윗줄에는
'package' 라는 keyword와
package 이름, 예를 들면, com dot company를 적어야 합니다.
즉, 같은 package에 속한 모든 Java source file들의
맨 윗 줄은 이와같이 자신이 속한 package 이름이 적혀있어야 합니다.

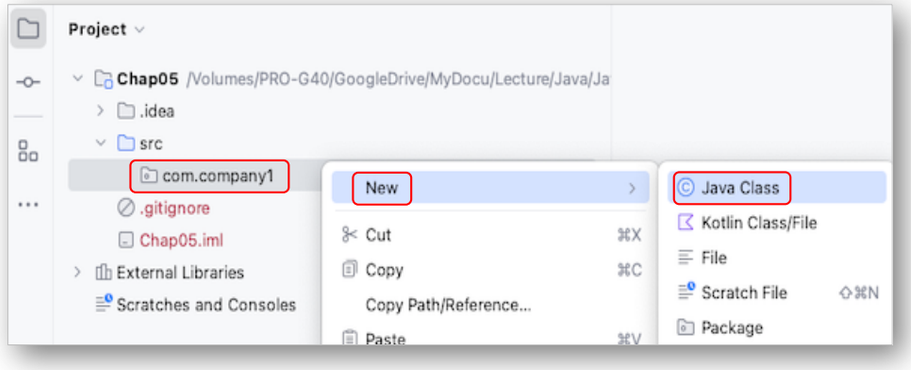
Creating Package in IntelliJ IDEA (1/6)



페이지 4

그럼 IntelliJ IDEA에서 새로운 package를 만들고
그 package에 새로운 class source file을 만드는 방법을
살펴 보도록 하겠습니다.
파일 네비게이터에서 project 이름 아래 src 폴더에서
right mouse click 하여 메뉴를 popup하고
new, Package를 차례로 선택하고
나오는 popup에 package 이름을 적습니다.
여기에서는 com dot company1 을 Package 이름으로 적었습니다.

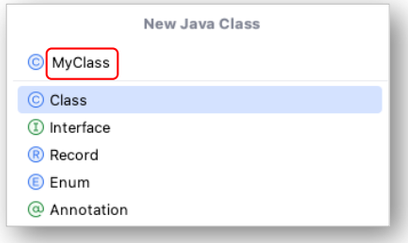
Creating Package in IntelliJ IDEA (2/6)

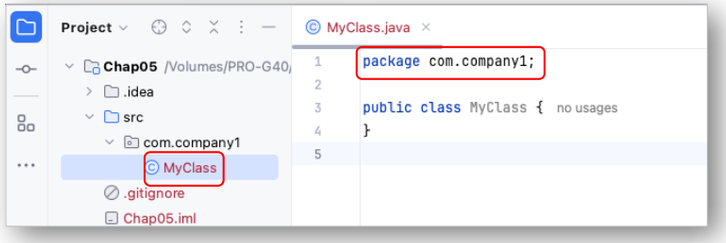


Page 5

새로 생긴 패키지 이름을 right mouse click하여 메뉴를 열고
New > Java Class 를 선택하여
새로운 class를 만듧니다.

Creating Package in IntelliJ IDEA (3/6)





페이지 6

여기에서는 com dot company1 package 아래에
MyClass라는 class를 만들었습니다.

import Statement

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

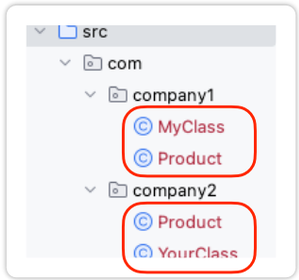
```
// import all classes in package 'com.company1'
import com.company1.*;

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


페이지 7

같은 package에 속하지 않은 class를 사용하려 할 때에는
import statement를 사용합니다.
만약 com dot company1 에 속하는 모든 class를
import 하려 할 때에는
import com dot company1 dot * 를 하면되고
특정한 class 하나만을 import 하려고 하면
예를 들어 import com dot company2 dot YourClass 와 같이
하나의 class만을 import 합니다.

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();
    }
}
```

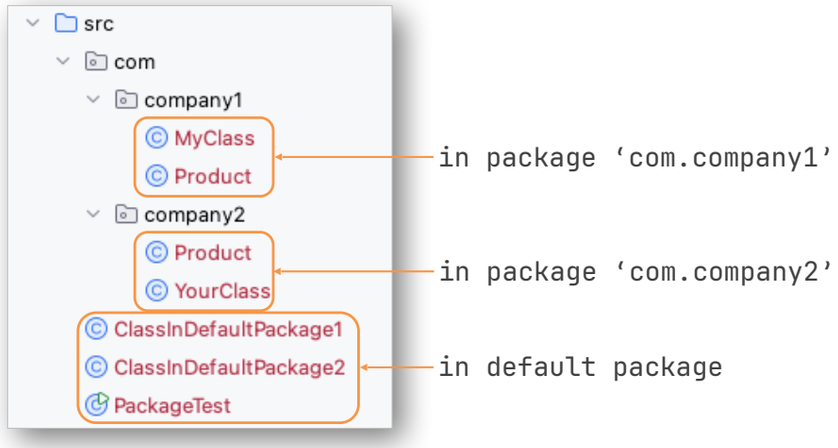
- class 'Product' exists in both packages, so, they should be distinguished by full name including the package name.

페이지 8

PackageTest.java 프로그램을 보도록 하겠습니다.
com dot company1 package 안에는 MyClass와 Product class가 있고
com dot company2 package안에는 YourClass와 Product class가 있습니다.
여기에서 양쪽 package들 안에 Product라는 class가
모두 들어있는 것을 확인할 수 있습니다.
처음에 com dot company1 과 com dot company2 의
모든 class들을 import 하였습니다.
PackageTest class의 main method 안에서
먼저 MyClass의 object인 mClass와
YourClass의 object인 yClass를 하나씩 create 하였습니다.
이제 각기 다른 package 안의 Product class의 object를
하나씩 create 하려 하는데
Product class의 이름이 같기 때문에
package의 full path를 모두 써 주어야 합니다.

Default Package

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



페이지 9

그런데 지금까지 우리는 class를 정의할 때
그 class가 속하는 package를 명시하지 않고
그냥 define하는 경우가 대부분이었습니다.
이렇게 class가 속하는 package를 명시하지 않으면
이 class는 어떤 package에 속하게 되는 것일까요?
답은 default package입니다.
default package는 다른 sub folder 아래가 아닌
src 폴더를 말합니다.
이 class 구성의 예를 보면
com dot company1 package 아래 MyClass, Product가 있고
com dot company2 package 아래 YourClass, Product가 있으며
com 아래가 아닌 src 폴더에
ClassInDefaultPackage1과 ClassInDefaultPackage2,
PackageTest class들이 있는데
이 세개의 class들은 default package 소속입니다.

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, ...

...

페이지 10

Java가 제공하는 built-in package들 중에
많이 쓰이는 package들에 대해 알아 보겠습니다.
먼저 java.lang은 가장 기본적인 class들을 포함하고 있습니다.
java.lang package는 import 하지 않아도 기본적으로 import가 됩니다.
대표적인 class로는 Object, String,
Math, System, Thread 등이 있습니다.

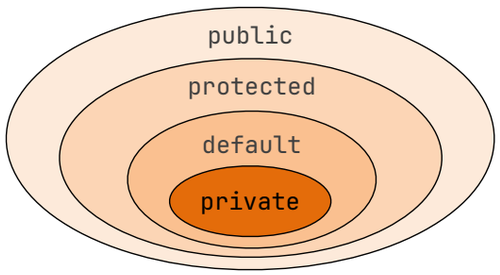
java.util package는 data 구조나 utility class들을 가지고 있습니다.
Scanner, ArrayList, HashMap, Date,
Calendar, Collections 등의 class들이 있습니다.

java.io package는 input과 output 기능을 제공하는
class들로 구성되어 있습니다.
File, InputStream, Reader, Writer 등의 class들로 구성되어 있습니다.

그외에도 Java에서는 많은 package들이 제공되고 있습니다.

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



페이지 11

이제 access modifier에 대해 알아보겠습니다.
public modifier가 붙어있는
instance variable, method, class 등은
같은 package가 아닌 다른 곳에서도
모두 access가 가능합니다.
protected 가 붙어있으면
같은 package 내에서 또는 child class에서 access가 가능합니다.
child class에 대해서는 inheritance (상속) 챕터에서
자세히 살펴볼 것입니다.
private가 붙어있으면
같은 class 내에서만 access가 가능합니다.
default 또는 package 가 붙어 있으면
같은 package (폴더) 이내에서는
access가 가능합니다.
벤 다이어그램은
access modifier들의 포함관계를 보여주고 있습니다.
private access가 가장 좁은 access 범위를 가지고 있고
default, protected, public access의 차례로
access 범위가 넓어집니다.

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();
    }
}
```


페이지 12

access modifier의 사용예로
default, 즉, package access인 class를 사용하는
example을 보도록 하겠습니다.
먼저 com dot company1 package에는
MyClass, NotPublicClass, Product, PublicClass 의
4개 class들이 속해 있습니다.
이 중 NotPublicClass는 access modifier가 붙어있지 않으므로
default, 즉, package access입니다.
PublicClass 앞에는 public access modifier가 붙어있고
따라서 public class가 되겠습니다.

이제 com dot company2 package의 Product class에서
com dot company1 의 모든 class들을 import 하였습니다.
그리고 NotPublicClass와 PublicClass의 object를
생성하려고 했습니다.
이 때 NotPublicClass는 public class가 아니라서
자신의 package 밖에서는 access가 불가능 합니다.
따라서 compile error가 발생하게 됩니다.
그러나 PublicClass의 경우는 public class이기 때문에
다른 package에서도 자유롭게 접근이 가능합니다.

public and private Fields (1/2)

```
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();
    }
}
```


페이지 13

AClass의 instance variable들 중
x는 public, y는 private, z는 package access입니다.
default constructor에서 x, y, z의 값을 각각
2, 3, 4로 assign 했습니다.
그 아래에는 public access권한의 publicMethod() 가 있고
private권한인 privateMethod(),
package권한인 packageMethod() 의
총 세개의 method들이 있습니다.

public and private Fields (2/2)

```
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();

    }
}
```


페이지 14

이제 AClassTest의 main method에서
AClass의 object인 ac를 하나 생성합니다.
이때 default constructor가 실행되게 되니까
ac의 x, y, z는 각각 2, 3, 4의 초기값을 가지게 됩니다.
ac.x는 정상적으로 print되는데
ac.y를 print하려는 순간 compile error가 나게 됩니다.
이것은 ac.y가 AClass에서 private instance variable이기 때문입니다.
즉 y는 AClass 내에서만 사용할 수 있는 것입니다.
ac.z 는 package 권한이라서 같은 폴더 (패키지) 내에 있는
AClassTest class에서는 자유롭게 access가 가능합니다.
또 public method인 ac.publicMethod()와
package method인 ac.packageMethod()
자유롭게 call 하는 것이 가능합니다.
그러나 private method인 ac.privateMethod() 의 경우
같은 class 이내에서만 사용이 가능하기 때문에
외부에서 call했을 경우 compile error가 나게 됩니다.

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.

페이지 15

information hiding의 개념의 측면에서
class 안의 모든 member를
private으로 선언하는 것을 권고하고 있습니다.
이것은 class외부에서 member를 잘못 사용하는 것을
막아주는 역할을 합니다.
대신에 private member의 값을 읽어주고
또는 값을 바꾸어 줄 수 있는
public method들을 따로 만들어 두어야 합니다.

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.

페이지 16

Accessor는 Getter라고도 불리는데
어떤 class의 외부로부터
그 class의 private variable의 값을 읽는데 사용되는
public method를 말합니다.
예를 들면, private int x; 를 위한 accessor는
public int getX()가 되고
private String str; 를 위한 accessor는
public String getStr() 이 됩니다.
이 accessor method가 하는 일은 정말 간단해서
private variable의 값을 그대로 return 하는 것입니다.

한편 Mutator는 Setter라고도 불리는데
private variable에 값을 써주는 public method입니다.
예를 들면 void setX(int); void setStr(String); 등이 가능합니다.
Mutator의 역할도 매우 간단해서 주어진 parameter 값을
private instance variable에 assign해 주는 것입니다.
그런데 parameter의 값이 유효하지 않은 data 인지를
먼저 test해 보고 private variable에 assign하도록
할 수 있습니다.
예를 들면 날짜의 '월' 은 1부터 12사이의 정수가 아니면
유효하지 않기 때문에 테스트에서 걸러질 수 있습니다.

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;
    }
```


페이지 17

이제 accessor와 mutator의 실제 사용 예를
보도록 하겠습니다.
Bclass에 있는 int x와 String str이
모두 private instance variable 입니다.
Constructor에서 x와 str의 initial value들을
assign 해 주고 있습니다.
public int getX()와 public String getStr()은
accessor method들입니다.
accessor들이 하는 일은
instance variable 값을 대신 읽어서 return해 주는 것입니다.

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);
    }

}
```


페이지 18

Mutator인 setX와 setStr은
x와 str에 새로운 값을 assign해 주는
public method들입니다.
특히 setStr에서는 주어진 parameter String을
그대로 assign하지 않고
새로운 String object를 만들어서
assign 해 줍니다.
이렇게 하면 privacy leak을 방지할 수 있습니다.
privacy leadk에 대해서는
05_3 Copy Constructor에서 자세히 공부할 것입니다.

Example: Accessor and Mutator (3/3)

```
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
```


페이지 19

처음에 default constructor에서
x와 str을 3과 "Korea"로 initialize했고
accessor로 읽어서 print했습니다.
mutator로 x와 str 값을 5와 "Seoul" 로 바꿔 주었고
다시 accessor로 읽어서 print했습니다.