Access Modifiers in Java

Session 박제영

NL_Algorithm

0. Overview

- public
- private
- protected
- default (no keyword)

1. Default

키워드를 명시적으로 사용하지 않을 때,(즉, 생략 되있을 경우)

Java는 특정 Class, Method 또는 Property에 대한

기본 접근 권한을 설정합니다.

이 말은, 모든 구성원이 같은 패키지 안에서 볼 수 있다는 것 입니다.

하지만, 다른 패키지에서는 접근할 수 없습니다.

1. Default//Example

A good example...

```
package com.NL_GOOD.accessmodifiers;

public class SuperPublic {
    static void defaultMethod() {
        ...
    }
}

package com.NL_GOOD.accessmodifiers;

public class Public {
    public Public() {
        SuperPublic.defaultMethod(); // 같은 Package 내에서 사용가능.
    }
}
```

A bad example...

```
//이는 같은 패키지가 아닐때의 예시를 보여줌
package com.NL_algorithm

public class Public{
                public Public(){
                                SuperPublic.deafaultMethod(); // NO.
                }
}
```

2. Public

Class, Method 또는 Property에 공용 키워드(Public)을 추가하면,

모든 코드파일(World of Java)에서 사용가능 합니다. (Literally, Public)

(즉, 모든 패키지의 다른 모든 클래스가 이를 사용할 수 있습니다)

다음 예시는 제한적이지 않은 접근제어자 입니다.

2. Public//Example

```
package com.NL_GOOD.accessmodifiers;

public class SuperPublic {
    public static void publicMethod() {
        ...
    }
}
```

```
package com.NL_GOOD.accessmodifiers.another;

import com.NL_GOOD.accessmodifiers.SuperPublic;

public class AnotherPublic {
    public AnotherPublic() {
        SuperPublic.publicMethod();
        }
}
```

3. Private

모든 Method, Property 또는 Constructor(생성부분)에

private 키워드를 사용하면 동일한 클래스에서만 액세스할 수 있습니다.

이는 가장 제한적인 접근제어자 입니다,

캡슐화(encapsulation) 개념의 핵심입니다.

모든 데이터는 외부 코드로부터 보호, 제한 가능합니다.

3. Private//Example

```
package com.NL_GOOD.accessmodifiers;

public class SuperPublic {
    static private void privateMethod() {
        ...
    }

     private void anotherPrivateMethod() {
         privateMethod(); // available in the same class only.
    }
}
```

4. Protected

Public 과 Private 사이에 있는 Class로써,

이는 말그대로 보호하는 접근 제어자 입니다.

만약 Method, Property 또는 Constructor를 Protected 와 함께 선언을 해주면,

같은 패키지(package-private 수준에서)내의 서브클래스에서 접근 가능합니다.

다음 예시는 다른 Package에 놓여있는 예시 입니다.

4. Protected//Example

```
package com.NL_GOOD.accessmodifiers;

public class SuperPublic {
    static protected void protectedMethod()
{
        ...
    }
}
```

```
package com.NL_GOOD.accessmodifiers.another;

import com.NL_GOOD.accessmodifiers.SuperPublic;

public class AnotherSubClass extends SuperPublic {
    public AnotherSubClass() {
        SuperPublic.protectedMethod();
    }
}
```

5. Conclusion

다음은 각 제어자에 대한 접근들을

표로 재구성 해보았습니다.

표를 통해 다음과 같이 제어자들의

공개 범위가 좁혀 집니다.

Public > Protected > Default > Private

제어자	Class	Package	Subclass	World
Public	Y	Y	Y	Y
Protected	Y	Y	Y	N
Default	Y	Y	N	N
Private	Y	N	N	N

6. Access My house ?

6. Access My house ?

```java
package com.Myhome;

class MyHouse {
    public String garden = "정원";
    private String bedroom = "내 방";
    protected String kitchen = "부엌";
    String livingRoom = "거실";

    public void showInfo() {
        System.out.println("집 정보:");
        System.out.println("거실: " + livingRoom);
        System.out.println("내 방: " + bedroom);
        System.out.println("부엌: " + kitchen);
        System.out.println("정원: " + garden);
    }
}
```

```java
class Neighbor extends MyHouse {
    void visit() {
        System.out.println("이웃이 방문합니다.");
        System.out.println("거실: " + livingRoom);
        // System.out.println("내 방: " + bedroom); //private 주의!
        System.out.println("부엌: " + kitchen);
        System.out.println("정원: " + garden);
    }
}


public class justdoit {
    public static void main(String[] args) {
        MyHouse myHouse = new MyHouse();
        myHouse.showInfo();

        Neighbor neighbor = new Neighbor();
        neighbor.visit();
    }
}
```