Access Control Specifiers

Information Hiding (정보은닉)

- 클래스 안의 멤버들을 보이지 않도록 감싸는 것이다.
- 커피 자동판매기 class
  - 커피, 프림, 컵
  - 자동판매 : 설탕커피, 프림커피, 블랙커피..

- 사람이 자판기를 열어서 커피, 프림, 컵을 직접 꺼내 혼합해서 마시지는 않는다.
  (자판기의 재료 (변수)에 직접 접근하는 것은 의미가 없고, 안의 재료들과 기능들을 직접 만지게 되면 잘못될 수도 있다.)

- 자판기 안의 일어나는 일들과 재료들이 보이지 않게 뚜껑으로 닫고(정보은닉) 사용자는 버튼만 누르면 된다.

Access Control Specifier - Private

```
class A {
	public String y(){
		Return "public void y()";
	}

	private String z(){
		Return "public void z()";
	}

	public String x(){
		Return z();
		// x()는 z()와 같은 Class의 소속이므로 private인 z() 메소드를 사용할 수 있다.
	}
}

public class AccessDemo1 {
	public static void main(String[] args) {
		A a = new A();

		System.out.println(a.y());

		// System.out.println(a.z());, 외부의 다른 클래스에서 사용하려고 하니 ERROR

		System.out.println(a.x());
	}
}
```

- A class가 가지고 있는 멤버가 public이면 누구든지 A class의 멤버를 사용할 수 있다.
- 하지만 멤버가 private이면 A class 내부적으로는 사용할 수 있지만, 외부의 다른 class는 A의 멤버를 사용할 수 없다.

Access Control Specifier의 사용 이유?

PUBLIC PRIVATE

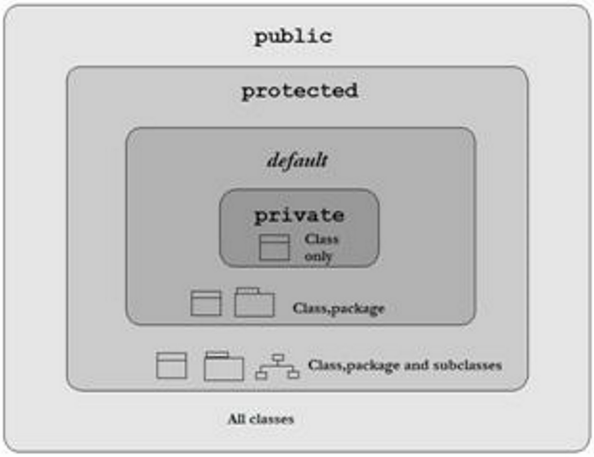
에플리케이션이 커진다는 것은 다른 말로 망가질 확률이 커진다는 의미와 같다.
특히 로직이 망가지는 첫번째 용의자는 사용자다.
객체를 사용하는 입장에서 객체 내부적으로 사용하는 변수나 메소드에 접근함으로서
개발자가 의도하지 못한 오동작을 일으키게 되는 것이다.

객체의 로직을 보호하기 위해서는 맴버에 따라서 외부의 접근을 허용하거나 차단해야 할 필요가 생긴다.
마치 은행이 누구나 접근 할 수 있는 창구와 관계자외에는 출입이 엄격하게 통제되는 금고를 구분하고 있는 이유와 같다.

CalculatorDemo (Use Access Control Specifier)

```
class Calculator {
     private int left, right;

     public void setOprands(int left, int right) {
          this.left = left;
          this.right = right;
     }

     사용자(CalculatorDemo.class)에게는 sum()이라는 메소드를 노출시킬 필요가 없고,
     더한 결과에 데코레이션만 해주는 기능만 제공해주면 된다는 취지의 프로그램
     사용자의 입장에서는 public 멤버들만 신경쓰면 된다는 장점이 있다!

     private int _sum() {  return this.left + this.right; }

     public void sumDecoPlus() { System.out.println("++++"+_sum()+"++++"); }

     public void sumDecoMinus() { System.out.println("----"+_sum()+"----"); }

}

public class CalculatorDemo {
     public static void main(String[] args) {
          Calculator c1 = new Calculator();
          c1.setOprands(10, 20);
          c1.sumDecoPlus();
          c1.sumDecoMinus();

          // c1.left = 100; 인스턴스 변수를 private 하지않으면 이런 문제가 발생
     }
}
```

Access Control Specifier - 세밀한 제어



- 같은 패키지 안에서는 인스턴스를 생성하던, 상속을 받던 private를 제외한 모든 접근제어 지시자에 접근 가능
- 다른 패키지에 있는 클래스에는 public만 접근가능
- **상속을 받으면 패키지에 관계없이 Protected 접근가능**
- **같은 패키지 내에서만 default 접근가능**

같은 클래스(클래스 자신)	public	protected	default	private
같은 패키지	public	protected	default	
다른 패키지	public			
다른 패키지 + 상속	public	protected		

Access Control Specifier - Class

Public인 Class는 다른 Package의 Class에서도 사용할 수 있고, default인 경우는 같은 Package에서만 사용 가능하다.

한가지 중요한 제약 사항이 있다.
public class가 포함된 Source Code는 public Class의 이름과 Source Code의 파일이름이 같아야 한다.
public PublicClass { } → PublicClass.java
하나의 소스코드 안에는 하나의 public class만 가질 수 있다!

```
package org.opentutorials.javatutorials.accessmodifier.inner;
public class PublicClass {}

package org.opentutorials.javatutorials.accessmodifier.inner;
class DefaultClass {}

package org.opentutorials.javatutorials.accessmodifier.outer;
public class PublicClass {}
```

```
package org.opentutorials.javatutorials.accessmodifier.inner;

public class ClassAccessModifierInnerPackage {
     PublicClass publicClass = new PublicClass();
     DefaultClass defaultClass = new DefaultClass();
}
```

```
package org.opentutorials.javatutorials.accessmodifier.outter;
import org.opentutorials.javatutorials.accessmodifier.inner.*;

public class ClassAccessModifierOuterPackage {

PublicClass publicClass = new PublicClass();
//  DefaultClass defaultClass = new DefaultClass();

}
```
