은닉성

#01. 은닉성(캡슐화)

객체지향 언어적 요소를 활용하여 객체에 대한 구체적인 정보를 노출시키지 않도록 하는 기법

쉽게 말하면 멤버변수나 메서드가 객체에 노출되지 않고 은닉되도록 설정함

객체를 사용하는 측의 실수로 인한 기능의 오작동을 방지하기 위해, 클래스의 일부를 숨기는 처리

1) 캡슐화를 적용하는 이유

객체에 잘못된 데이터가 저장되는 것을 방지하기 위함

> Ex01_멤버변수_접근의_한계.java

2) 적용방법

멤버변수나 메서드와 생성자 이름 앞에 **접근 한정자**를 명시한다.

접근 한정자의 종류

접근한정자	설명
public	모든 곳에서 접근 가능
private	같은 클래스 내에서만 접근 가능. 객체를 통해 접근할 수 없다.
protected	같은 패키지 안에 존재하는 클래스와 현재 클래스를 상속받는 하위 클래스에서 접근 가능 (잘 사용하지 않는다.)
(default)	접근한정자를 명시하지 않은 경우. 동일한 소스파일 내에서만 접근 가능하다. (잘 사용하지 않는다.)

> 패키지, 상속에 대한 내용은 추후에 다룰 예정

접근 한정자를 적용한 예시

```
class PrivateTest {
    public int a;
    private int b; // 은닉 되어 있으므로 객체를 통한 접근 불가
}

public class SimplePrivate {
    public static void main(String[] args) {
        PrivateTest pTest = new PrivateTest();
        pTest.a = 200;  // --> public이므로 문제 없음.
        pTest.b = 200;  // --> private이므로 객체를 통해서는 접근할 수 없다.(에러)
```

```
    }
}
```

접근 한정자를 적용한 예시의 에러 메시지

```
SimplePrivate.java:10: error: b has private access in PrivateTest
        pTest.b = 200;  // --> private이므로 객체를 통해서는 접근할 수 없다.(에러)
        ^
1 error
```

정리

극단적인 획일화가 올바른 방법은 아니지만 객체지향을 모두 이해할 때 까지 아래의 규칙을 사용한다면 큰 문제는 없을 것이다.

1. **모든 멤버변수에는 무조건 private**를 적용한다.
    - 객체지향 언어는 데이터가 직접 적으로 노출되는 것이 소스코드 보안에 안좋다고 본다.
2. **모든 메서드에는 무조건 public**을 적용한다.
3. default는 사용하지 않는다.
    - 멤버변수나 메서드 이름 앞을 비워두지 않고 접근한정자 명시
4. protected는 public으로 대체한다.

#02. getter, setter

멤버변수가 은닉된 형태로 선언된 경우 프로그램의 가장 근본적인 목적인 데이터에 접근하는 방법이 사라지므로 **메서드를 통하여 간접적으로 접근하는 방법이 마련**되어야 한다.

getter, setter는 은닉된 멤버변수에 간접적으로 접근하기 위하여 정의된 메서드들을 의미하는 용어이다.

용어	설명
Getter	은닉된 멤버변수의 값을 **리턴** 하기 위한 메서드
Setter	**파라미터**로 전달된 값을 멤버변수에 **복사**하기 위한 메서드

1) 메서드 작성 규칙

- “get”, “set” 접두사 뒤에 변수이름을 첫 글자가 대문자인 형태로 명시
- Getter는 연결된 멤버변수의 데이터 타입을 리턴형으로 명시
- Setter는 연결된 멤버변수의 데이터 타입을 파라미터 형으로 명시

getter, setter는 자바 언어의 중요한 **코딩 규약**이기 때문에 반드시 준수해야 함

준수하지 않더라도 당장 에러는 없지만, 이후 진행할 수 많은 상위 기능들과의 연동이 불가능해짐

멤버변수	getter	setter
String userName	String getUserName()	void setUserName(String userName)

멤버변수	getter	setter
int age	int getAge()	void setAge(int age)

> Ex02_멤버변수_은닉.java

#03. 클래스에 대한 접근 한정자.

1) 클래스에 접근 한정자 지정하기

접근한정자	설명
public	객체 생성 가능. 서로 다른 소스코드에 정의된 클래스끼리도 객체 생성이 가능하다.
private, protected	클래스에 적용할 수 없다.
(default)	동일한 소스코드에 정의된 클래스끼리만 객체 생성이 가능하다. (지금까지의 예제 형태)

2) 클래스의 분리

어떤 프로그램이 단 하나의 소스코드로 모든 기능을 구현하고 있다면 하나의 파일이 지나치게 길어지게 되므로 유지보수에 비효율적이게 된다.

그렇게 때문에 프로그램을 개발할 때는 적절한 기능 단위로 소스코드를 분리해야 한다.

소스코드가 분리된 클래스끼리는 public이 명시되어야만 서로 객체 생성이 가능하다.

일반적으로 **클래스 정의에는 public 접근 한정자만 사용**한다.

하나의 소스코드에는 하나의 public 클래스만 존재할 수 있다.

다른 소스코드를 통해 객체 생성하기

> Member.java, Ex03_MemberJoinTest.java

#04. 자바에서의 데이터 표현 단위

1) 자바빈즈(JavaBeans)

자바 언어에서 사용하는 **복합적 데이터 표현의 최소 단위**.

재사용 가능한 컴포넌트(=객체)를 생성할 수 있다.

자바 빈즈 생성 규칙

자바빈즈 클래스로서 작동하기 위해서, 클래스는 명명법, 생성법 그리고 행동에 관련된 **일련의 관례를 따라야만 한다.**

이러한 관례는 빌더 형식의 개발 도구에서 자바빈즈와의 연결을 통해 객체의 생성과 호출, 그리고 클래스의 재배치를 자동으로 처리하도록 한다.

1. 클래스는 생성자를 가지고 있어야 한다. (별도의 생성자를 정의하지 않고 기본 생성자로 대체할 수 있다.)
2. 클래스의 속성들은 get, set 혹은 표준 명명법을 따르는 메서드들을 사용해 접근할 수 있어야 한다.

💡 앞 예제에서 작성한 Member 클래스가 JavaBeans 이다.

JavaBeans에서 각 구성요소의 용도

구분	설명
멤버변수	표현하고자 하는 데이터를 저장한다.
생성자	객체가 생성될 때 데이터를 입력한다. (기본 생성자를 사용하는 경우 기능 없음)
setter	객체가 저장하고 있는 데이터를 수정한다. 생성자가 없을 경우 데이터를 입력하는 기능을 수행하기도 한다.
getter	객체가 저장하고 있는 데이터를 반환하여 다른 프로그램으로 하여금 그 값을 활용할 수 있도록 한다.

💡 기본 생성자를 사용한다는 것은 소스코드 상에 별도의 생성자를 정의하지 않았다는 것을 의미한다.

뉴스 기사를 표현하는 클래스 예시

기사 번호, 제목, 내용, 작성자, 작성 일시 등을 표현하는 JavaBeans 클래스

> Article.java, Ex04_ArticleTest.java

2) JavaBeans를 의미하는 다른 용어들

POJO 클래스

특정 기술규약과 환경에 종속되지 않으면서 객체지향적인 원리에 충실한 클래스.

환경과 기술에 종속되지 않고 필요에 따라 재활용될 수 있는 방식으로 설계된 오브젝트를 말한다.

일반적으로 기본 Java 코드만으로 작성되어진 JavaBeans를 의미한다.

VO (Value Object)

값(value)만을 저장하고 있는 객체를 의미한다.

JavaBeans에 의해 생성된 객체를 의미한다.

보통 클래스 이름 뒤에 "VO"를 붙인다. —> **ArticleVO**

DTO (Data Transfer Object)

데이터를 전송하는데 사용되는 객체.

JavaBeans에 의해 생성된 객체를 파라미터나 리턴값으로 사용할 경우 DTO라고 부른다.

일반적으로 데이터베이스 시스템 연동시에 많이 사용한다.

보통 클래스 이름 뒤에 "DTO"를 붙인다. —> **ArticleDTO**