프로그래밍 언어 활용 강의안

1. 정적 멤버와 static

정적(static) 멤버란?

클래스에 고정된 필드와 메소드 - 정적 필드, 정적 메소드

정적 멤버는 클래스에 소속된 멤버

객체 내부에 존재하지 않고, 메소드 영역에 존재

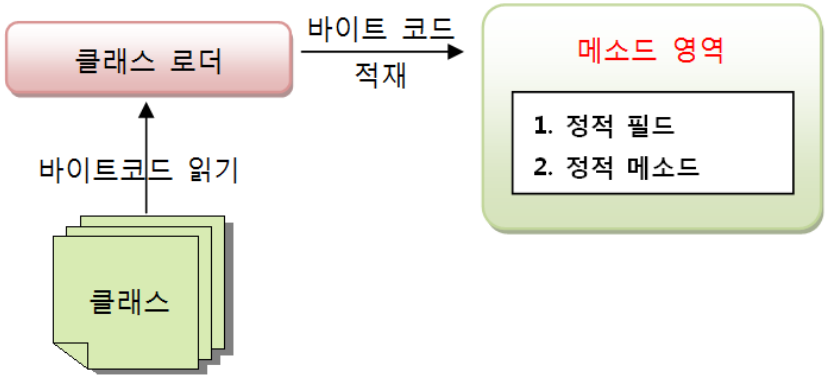
정적 멤버는 객체를 생성하지 않고 클래스로 바로 접근해 사용

정적 멤버 선언

필드 또는 메소드 선언할 때 static 키워드 붙임

```
public class 클래스 {
  //정적 필드
  static 타입 필드 [= 초기값];

  //정적 메소드
  static 리턴타입 메소드( 매개변수선언, ... ) { ... }
}
```



1. 정적 멤버와 static

정적 멤버 사용

클래스 이름과 함께 도트(.) 연산자로 접근

```
클래스.필드;
클래스.메소드( 매개값, ... );
```

```
public class Calculator {
   static double pi = 3.14159;
   static int plus(int x, int y) { ... }
   static int minus(int x, int y) { ... }
}
```

정적 ... 로 접근

[바람직한 사용]

```
double result1 = 10 * 10 * Calculator.pi;
int result2 = Calculator.plus(10, 5);
int result3 = Calculator.minus(10, 5);
```

[바람직하지 못한 사용]

```
Calculator myCalcu = new Calculator();
double result1 = 10 * 10 * myCalcu.pi;
int result2 = myCalcu.plus(10, 5);
int result3 = myCalcu.minus(10, 5);
```

1. 정적 멤버와 static

인스턴스 멤버 선언 vs 정적 멤버 선언의 기준

필드

객체 마다 가지고 있어야 할 데이터 ➔ 인스턴스 필드
공용적인 데이터 ➔ 정적 필드

```
public class Calculator {
   String color;                    //계산기 별로 색깔이 다를 수 있다.
   static double pi = 3.14159;   //계산기에서 사용하는 파이( π)값은 동일하다.
}
```

메소드

인스턴스 필드로 작업해야 할 메소드 ➔ 인스턴스 메소드
인스턴스 필드로 작업하지 않는 메소드 ➔ 정적 메소드

```
public Calculator {
   String color;
   void setColor(String color) { this.color = color; }
   static int plus(int x, int y) { return x + y; }
   static int minus(int x, int y) { return x – y; }
}
```

1. 정적 멤버와 static

싱글톤(Singleton)

하나의 애플리케이션 내에서 단 하나만 생성되는 객체

싱글톤을 만드는 방법

외부에서 new 연산자로 생성자를 호출할 수 없도록 막기

private 접근 제한자를 생성자 앞에 붙임

클래스 자신의 타입으로 정적 필드 선언

자신의 객체를 생성해 초기화

private 접근 제한자 붙여 외부에서 필드 값 변경 불가하도록

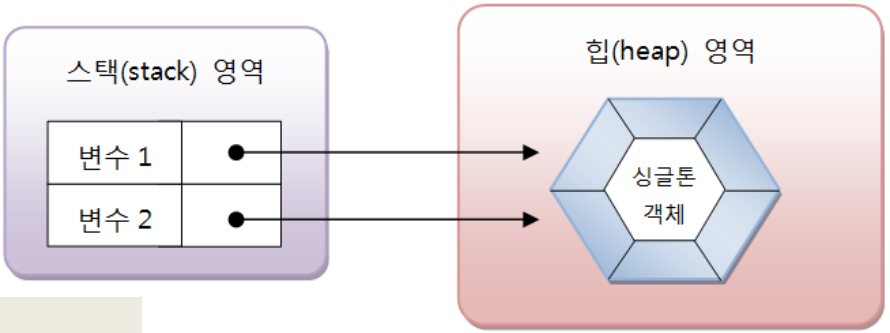
외부에서 호출할 수 있는 정적 메소드인 getInstance() 선언

정적 필드에서 참조하고 있는 자신의 객체 리턴

1. 정적 멤버와 static

싱글톤 얻는 방법

```
클래스 변수 1 = 클래스.getInstance();
클래스 변수 2 = 클래스.getInstance();
```




```
/*
Singleton obj1 = new Singleton();  //컴파일 에러
Singleton obj2 = new Singleton();  //컴파일 에러
*/

Singleton obj1 = Singleton.getInstance();
Singleton obj2 = Singleton.getInstance();

if(obj1 == obj2) {
   System.out.println("같은 Singleton 객체 입니다.");
} else {
   System.out.println("다른 Singleton 객체 입니다.");
}
```

2. Final 필드와 상수(static final)

final 필드

최종적인 값을 갖고 있는 필드 = 값을 변경할 수 없는 필드

final 필드의 딱 한번의 초기값 지정 방법

필드 선언 시

생성자

```
public class Person {
   final String nation = "Korea";
   final String ssn;
   String name;

   public Person(String ssn, String name) {
      this.ssn = ssn;
      this.name = name;
   }
}
```

2. 패키지

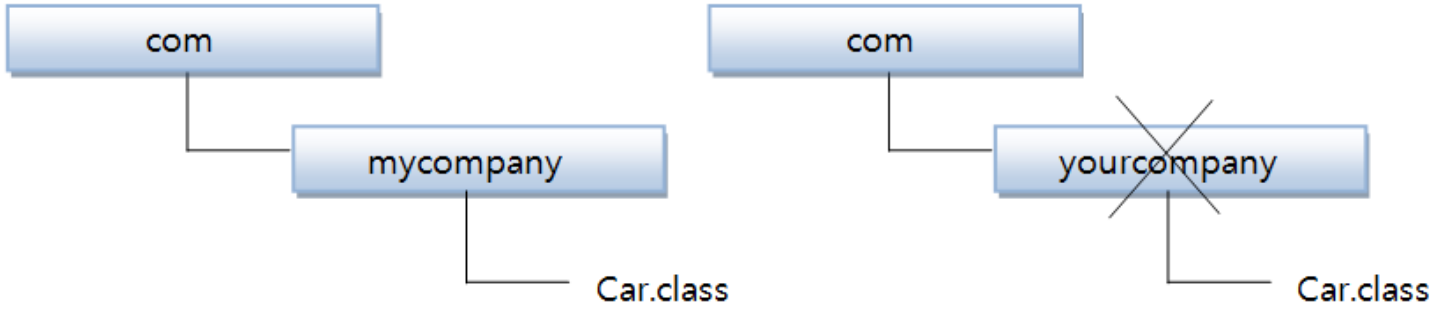
패키지란?

클래스 선언할 때 패키지 결정

클래스 선언할 때 포함될 패키지 선언

클래스 파일은(~.class) 선언된 패키지와 동일한 폴더 안에서만 동작

클래스 파일은(~.class) 다른 폴더 안에 넣으면 동작하지 않음



2. 패키지

import 문

패키지 내에 같이 포함된 클래스간 클래스 이름으로 사용 가능

패키지가 다른 클래스를 사용해야 할 경우

패키지 명 포함된 전체 클래스 이름으로 사용

```
pckage com.mycompany;

public class Car {
    com.hankook.Tire tire = new com.hankook.Tire();
}
```

Import 문으로 패키지를 지정하고 사용

```
package com.mycompany;

import com.hankook.Tire;
[ 또는 import com.hankook.*; ]

public class Car {
 Tire tire = new Tire();
}
```

Source>Organize imports (단축키: Ctrl+Shift+O)

1. Window>Preference>Java>Code Style>Organize imports 를 선택
2. Number of imports needed for .*의 99 를 1 로 변경하고 [OK] 버튼을 클릭한다.
3. 다시한번 Ctrl+Shift+O 를 클릭한다.

2. 패키지

접근 제한자(Access Modifier)

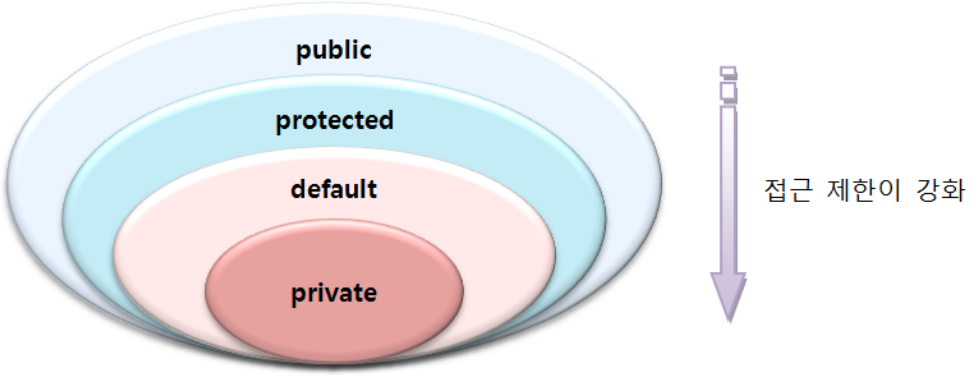
클래스 및 클래스의 구성 멤버에 대한 접근을 제한하는 역할

다른 패키지에서 클래스를 사용하지 못하도록 (클래스 제한)

클래스로부터 객체를 생성하지 못하도록 (생성자 제한)

특정 필드와 메소드를 숨김 처리 (필드와 메소드 제한)

접근 제한자의 종류



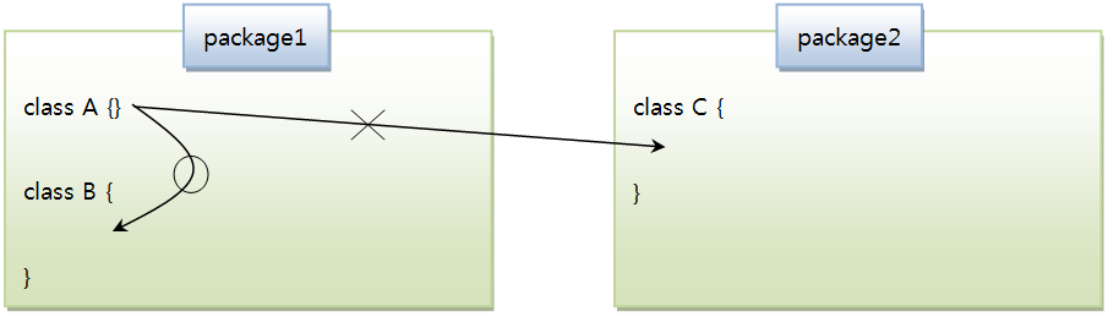
접근 제한	적용 대상	접근할 수 없는 클래스
public	클래스, 필드, 생성자, 메소드	없음
protected	필드, 생성자, 메소드	자식 클래스가 아닌 다른 패키지에 소속된 클래스
default	클래스, 필드, 생성자, 메소드	다른 패키지에 소속된 클래스
private	필드, 생성자, 메소드	모든 외부 클래스

2. 패키지

클래스의 접근 제한

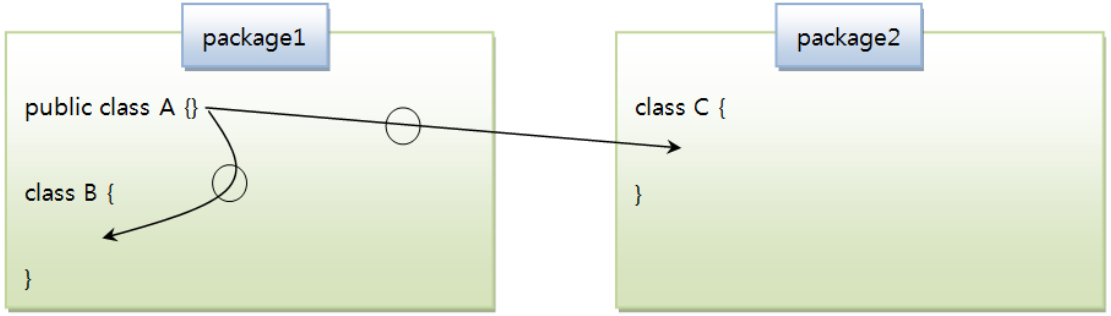
default

클래스 선언할 때 public 생략한 경우
다른 패키지에서는 사용 불가



public

다른 개발자가 사용할 수 있도록 라이브러리 클래스로 만들 때 유용



2. 패키지

생성자 접근 제한

생성자가 가지는 접근 제한에 따라 호출 여부 결정

필드와 메소드의 접근 제한

클래스 내부, 패키지 내, 패키지 상호간에 사용할 지 고려해 선언

3. Getter와 Setter

클래스 선언할 때 필드는 일반적으로 private 접근 제한
- 읽기 전용 필드가 있을 수 있음 (Getter의 필요성)
- 외부에서 엉뚱한 값으로 변경할 수 없도록 (Setter의 필요성)

Getter
- private 필드의 값을 리턴 하는 역할 - 필요할 경우 필드 값 가공
- getFieldName() 또는 isFieldName() 메소드
  - 필드 타입이 boolean 일 경우 isFieldName()

Setter
- 외부에서 주어진 값을 필드 값으로 수정
  - 필요할 경우 외부의 값을 유효성 검사
- setFieldName(타입 변수) 메소드
  - 매개 변수 타입은 필드의 타입과 동일