**package**

관련된 클래스들을 모아서 관리할 수 있도록 한다.  
한 곳에서 수백, 수천 개의 클래스를 찾는 수고를 덜어준다.  
배포할 때 관련된 소스 코드들을 묶어서 배포하고 재사용할 수 있도록 한다.  
각 프로젝트나 소프트웨어 간의 소스 코드 충돌을 방지한다.

■ Package Convention

* 일반적으로 패키지명은 회사 도메인명을 사용한다
* 모든 패키지 이름은 소문자
* 패키지명이 java와 javax로 시작해선 안된다. 이 둘은 JDK에서 독점적으로 사용중
* 보통 src/main/java/.. , src/test/java/.. 형태

■ import  
모든 클래스앞에 패키지명을 사용하는 것은 비효율적.  
java.util.Scanner sc = new java.util.Scanner(System.in);  
import를 활용해 바로 사용

# Class vs Instance

Class : Instance를 생성하기 위한 틀(template). 클래스만으론 상태가 없음.  
Instance : Class를 통해 실체화되어 생성됨. 상태를 갖고 메소드를 통해 상태가 변경된다.

#### 인스턴스 변수의 초기화

클래스를 가지고 객체를 생성하면, 해당 객체는 메모리에 즉시 생성됩니다.

하지만 이렇게 생성된 객체는 모든 인스턴스 변수가 아직 초기화되지 않은 상태입니다.

자바에서 클래스 변수와 인스턴스 변수는 별도로 초기화하지 않으면 자동 초기화됩니다.

하지만 사용자가 원하는 값으로 인스턴스 변수를 초기화하려면, 일반적인 초기화 방식으로는 초기화할 수 없습니다.

인스턴스 변수 중에는 private 변수도 있으며, 이러한 private 변수에는 사용자나 프로그램이 직접 접근할 수 없기 때문입니다.

따라서 private 인스턴스 변수에도 접근할 수 있는 초기화만을 위한 public 메소드가 필요해집니다.

이러한 초기화만을 위한 메소드는 객체가 생성된 후부터 사용되기 전까지 반드시 인스턴스 변수의 초기화를 위해 호출되어야 합니다.

#### 생성자(constructor)

자바에서는 객체의 생성과 동시에 인스턴스 변수를 원하는 값으로 초기화할 수 있는 생성자(constructor)라는 메소드를 제공합니다.

자바에서 생성자(constructor)의 이름은 해당 클래스의 이름과 같아야 합니다.

즉, Car 클래스의 생성자의 이름은 Car가 됩니다.

이러한 생성자는 다음과 같은 특징을 가집니다.

1. 생성자는 반환값이 없지만, 반환 타입을 void형으로 선언하지 않습니다.

2. 생성자는 초기화를 위한 데이터를 인수로 전달받을 수 있습니다.

3. 객체를 초기화하는 방법이 여러 개 존재할 경우에는 하나의 클래스가 여러 개의 생성자를 가질 수 있습니다.

    즉, 생성자도 하나의 메소드이므로, 메소드 오버로딩이 가능하다는 의미입니다.

선언방법

1. 클래스이름() { ... }                  *// 매개변수가 없는 생성자 선언*

2. 클래스이름(인수1, 인수2, ...) { ... } *// 매개변수가 있는 생성자 선언*

#### 생성자의 호출

자바에서는 new 키워드를 사용하여 객체를 생성할 때 자동으로 생성자가 호출됩니다.

##### 예제

**class** Car {

**private** **String** modelName;

**private** **int** modelYear;

**private** **String** color;

**private** **int** maxSpeed;

**private** **int** currentSpeed;

**Car**(**String** *modelName*, **int** *modelYear*, **String** *color*, **int** *maxSpeed*) {

this**.**modelName **=** modelName;

this**.**modelYear **=** modelYear;

this**.**color **=** color;

this**.**maxSpeed **=** maxSpeed;

this**.**currentSpeed **=** 0;

}

**public** **String** **getModel**() {

**return** this**.**modelYear **+** "년식 " **+** this**.**modelName **+** " " **+** this**.**color;

}

}

**public** **class** Method02 {

**public** **static** **void** **main**(**String**[] *args*) {

**Car** myCar **=** **new** **Car**("아반떼", 2016, "흰색", 200); *// 생성자의 호출*

**System.**out**.**println(myCar**.**getModel()); *// 생성자에 의해 초기화되었는지를 확인함.*

}

}

#### 기본 생성자(default constructor)

자바의 모든 클래스에는 하나 이상의 생성자가 정의되어 있어야 합니다.

하지만 특별히 생성자를 정의하지 않고도 인스턴스를 생성할 수 있습니다.

이것은 자바 컴파일러가 기본 생성자(default constructor)라는 것을 기본적으로 제공해 주기 때문입니다.

기본 생성자는 매개변수를 하나도 가지지 않으며, 아무런 명령어도 포함하고 있지 않습니다.

자바 컴파일러는 컴파일 시 클래스에 생성자가 하나도 정의되어 있지 않으면, 자동으로 다음과 같은 기본 생성자를 추가합니다.

##### 문법

클래스이름() {}

다음 예제는 자바 컴파일러가 Car 클래스에 자동으로 추가해 주는 기본 생성자의 예제입니다.

##### 예제

Car() {}

위와 같이 기본 생성자는 어떠한 매개변수도 전달받지 않으며, 기본적으로 아무런 동작도 하지 않습니다.

다음 예제는 Car 클래스에 생성자를 정의하지 않고, 기본 생성자를 호출하는 예제입니다.

##### 예제

**class** Car {

**private** **String** modelName **=** "소나타";

**private** **int** modelYear **=** 2016;

**private** **String** color **=** "파란색";

**public** **String** **getModel**() {

**return** this**.**modelYear **+** "년식 " **+** this**.**color **+** " " **+** this**.**modelName;

}

}

**public** **class** Method03 {

**public** **static** **void** **main**(**String**[] *args*) {

**Car** myCar **=** **new** **Car**();                *// 기본 생성자의 호출*

**System.**out**.**println(myCar**.**getModel()); *// 2016년식 파란색 소나타*

}

}