클래스와 객체

자바의 객체 지향 이해하기

# 0. 학습 목표

1. 객체 지향의 개념과 특성 이해
2. 자바 생성자 이해하기
3. 객체 배열의 선언과 활용
4. 객체의 소멸과 가비지 컬렉션 이해
5. 클래스와 멤버에 대한 접근 지정자
6. static 속성을 가진 멤버의 특성
7. final로 선언된 클래스, 메소드, 필드에 대한 이해

# 1. 객체 지향의 개념과 특성 이해

### 객체란?

* **클래스**  : 객체의 속성과 행위 선언. 객체의 설계도 혹은 틀.
* **객체** : 세상의 모든 것이 객체다! **클래스(틀로)로 찍어낸 실체.**
* **인스턴스** : 클래스로부터 만들어진 객체.

### 자바의 객체 지향 특성

* 1. **캡슐화** : 객체를 캡슐로 싸서 내부를 볼 수 없게 하는 것.
  2. **상속** : 상위 개체의 속성이 하위 개체에 물려지는 것.

하위 개체가 상위 개체의 속성을 모두 가진다.

* 상위 클래스 : 수퍼 클래스
* 하위 클래스 : 서브 클래스. 수퍼 클래스 코드 재사용, 코드 추가 가능
  1. **다형성** : 같은 이름의 메소드가 클래스 혹은 객체에 따라 다르게 구현되는 것.
* **메소드 오버로딩** : **한 클래스 내에서 같은 이름이지만 다르게 작용하는 여러 메소드**
* **메소드 오버라이딩** : **슈퍼 클래스의 메소드를 동일한 이름으로 서브 클래스마다 다르게 구현**

### 객체 지향 언어의 목적

1. 소프트웨어의 생산성 향상
2. 실세계에 대한 쉬운 모델링

# 2. 자바 생성자 이해하기

### **생성자**

* 객체가 생성될 때 초기화를 목적으로 실행되는 메소드
* 객체가 생성되는 순간에 자동으로 호출됨.

### 생성자의 특징

* 생성자의 이름은 클래스 이름과 동일해야 함.
* 생성자는 여러 개 작성할 수 있다. (생성자 중복)
* 생성자는 객체 생성 시 딱 한 번만 호출되며, 자바에서 객체 생성은 반드시 new 연산자로 함!
* 생성자는 리턴 타입을 지정할 수 없다.

### 생성자의 종류

* 기본 생성자 : 매개 변수가 없고, 아무 작업 없이 단순하게 객체를 리턴하는 생성자.

'디폴트 생성자' 라고도 부른다!

* 기본 생성자가 자동으로 생성되는 경우 : 클래스에 생성자가 하나도 선언되어 있지 X

> 컴파일러에 의해 기본 생성자가 자동으로 생성된다.

* 기본 생성자가 자동으로 생성되지 X : 클래스에 생성자가 선언되어 있는 경우

### this 레퍼런스 이해하기

* **this** : 객체 자신에 대한 레퍼런스. 컴파일러에 의해 자동으로 관리된다.

객체 자신의 멤버에 접근할 때 사용한다.

* this의 특징들

1. this와 this()는 다르다.
2. **this는 메소드에서 사용하며 현재 객체를 가리킨다.**
3. 하지만 static 메소드에서는 사용할 수 없다. (객체가 생성되지 않았을 수도 있다)

* **this()** : 같은 클래스의 다른 생성자를 호출하며, 생성자 내에서만 사용 가능하다.

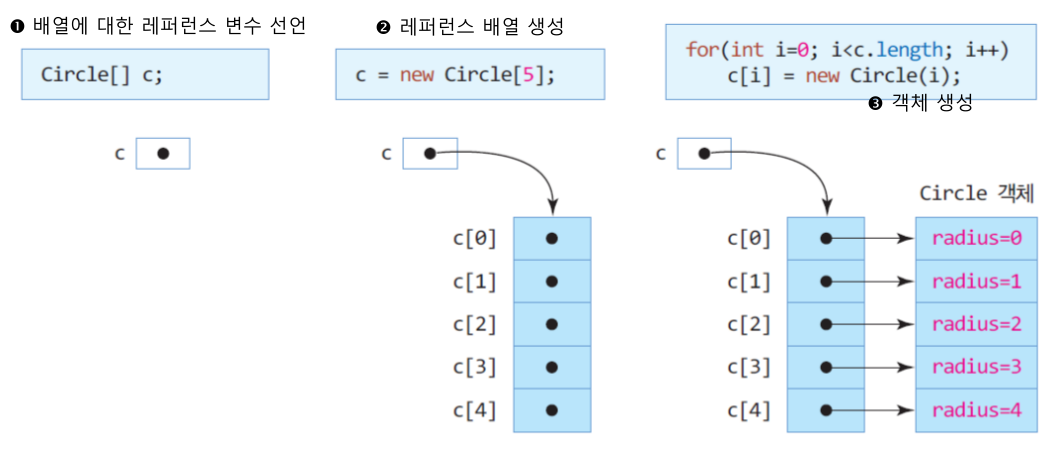
྿ 생성자 코드의 제일 처음에 있어야 한다.

# 3. 객체 배열의 선언과 사용

### 객체 배열이란?

* 객체에 대한 레퍼런스 배열임.
* 자바의 객체 배열 만들기

1. 배열 레퍼런스 변수를 선언한다.
2. 레퍼런스 배열을 생성한다.
3. 방금 만들어낸 배열의 각 방마다 원소 객체를 생성한다.



### 인자 전달

* 기본 타입의 값이 전달되는 경우 : 호출자가 건네는 값이 매개 변수에 복사되어 전달됨.

실제 인자 값은 변경되지 않는다. (call by value)

* 객체가 전달되는 경우 : 객체의 레퍼런스가 전달되어 매개 변수가 실제 인자 객체를 공유함!
* 배열이 전달되는 경우 : 배열의 레퍼런스가 전달되어 매개 변수가 실제 배열 객체를 공유함!

### 메소드 오버로딩

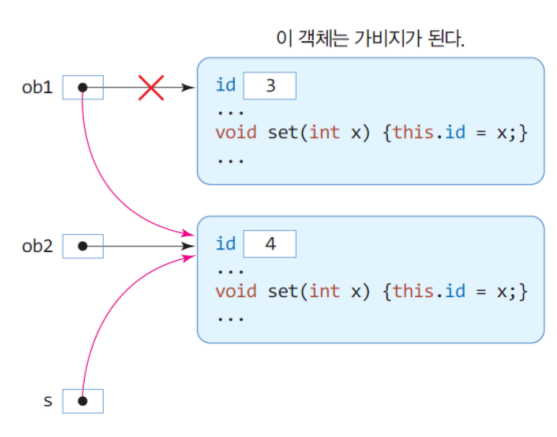
* **한 클래스 내에서 두 개 이상의 이름이 같은 메소드가 작성됨.**
* 메소드 이름이 동일해야 하고,
* 매개 변수의 개수가 서로 다르거나, 타입이 서로 달라야 한다.

# 리턴 타입은 오버로딩과 관련 없다!

# 4. 객체의 소멸과 가비지 컬렉션 이해

### 객체 치환

* **객체 치환은 객체 복사가 아니며, 레퍼런스의 복사이다.**
* 예를 들어, ob1 = ob2를 하면 기존에 ob1이 가리키던 객체와의 연결이 끊기는 것.



### 객체 소멸

* new에 의해 할당 받은 객체와 배열 메모리를 JVM에게 되돌려 주는 것.
* 소멸된 객체 공간은 가용 메모리에 포함된다.
* 자바에서는 사용자가 임의로 객체를 소멸할 수 없고, JVM이 알아서 다 해준다.

### 가비지

* **가비지** : 가리키는 레퍼런스가 하나도 없는, 더 이상 접근할 수 없게 된 메모리.
* 가비지 컬렉션 : JVM의 가비지 컬렉터가 자동으로 가비지를 수집해 반환한다.

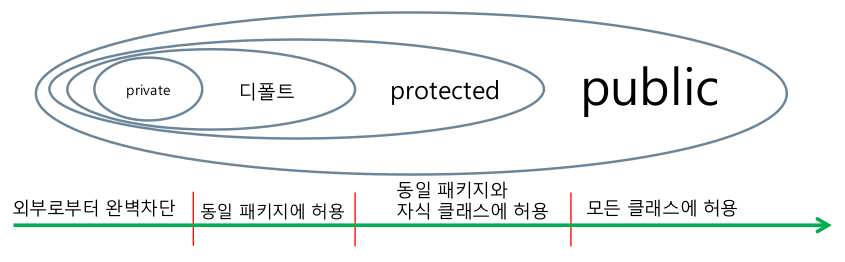
# 5. 클래스와 멤버에 대한 접근 지정자

### 패키지

* 상호 관련 있는 클래스 파일 (컴파일된 .class)을 저장해 관리하는 디렉터리
* 자바의 응용 프로그램은 하나 이상의 패키지로 구성된다.

### 접근 지정자

* 클래스나 일부 멤버를 공개해 다른 클래스에서 접근하도록 허용하거나,
* 객체 지향 언어의 캡슐화 정책에 따라 멤버를 보호하는 것.
* 접근 지정자는 캡슐화에 싸인 보호를 일부 해제할 목적으로 쓰인다.



### 멤버 접근 지정



# 6. static 속성을 가진 멤버의 특징

### static 멤버의 특징

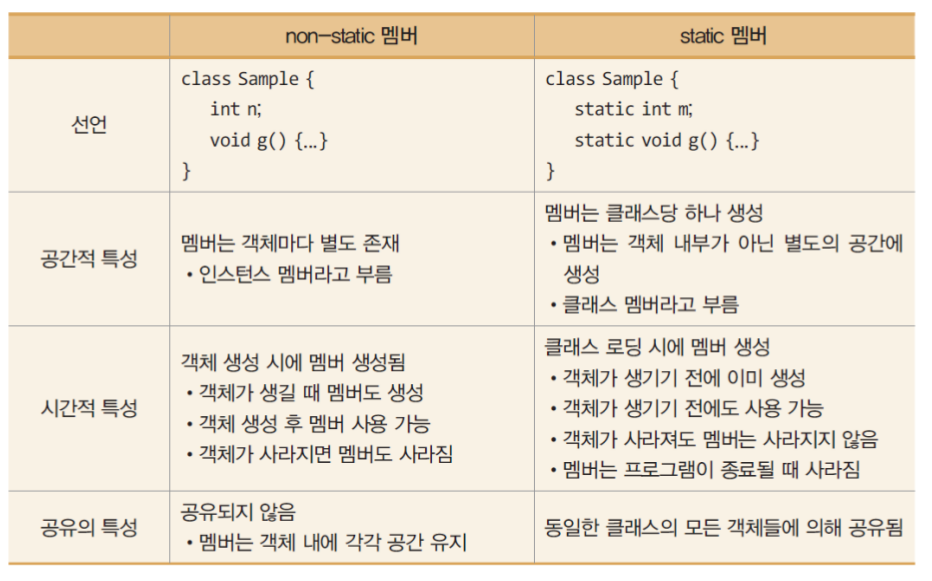
* 클래스 당 하나만 생성되며, 객체들에 의해서 공유된다
* static 멤버는 클래스 이름이나 객체의 멤버로 접근할 수 있다.

### static 멤버의 활용

1. 전역 변수와 전역 함수를 만들 때 활용
2. 객체들 간 공유 멤버를 만들고자 할 때 활용

### non-static 멤버의 특징

* 객체가 생성될 때, 각 객체마다 생긴다.
* non-static 멤버는 클래스 이름으로 접글할 수 없다.



# 7. final로 선언된 클래스, 메소드, 필드에 대한 이해

### final 클래스 : 더이상 클래스를 상속해 줄 수 없다!

### final 메소드 : 더 이상 오버라이딩을 할 수 없다!

### final 필드

* 상수를 선언할 때 사용하고, 실행 중 값을 변경할 수 X
* 선언 시에 초기 값을 지정해 줘야 한다.