상속(inheritance)

1. 기존에 선언된 클래스의 필드를 다른 클래스에서 사용하고자 할 때(코드의 재사용)

2. 두 클래스를 부모와 자식으로 관계를 맺어주는 것

- 상속 문법

```
// 부모클래스
class A {
        부모클래스의 필드
}

// 자식클래스
class B extends A {
        자식클래스의 필드
}
```

A : 상위클래스, 슈퍼 클래스

B : 하위클래스, 서브 클래스

B b = new B ( );

b 객체는 실제로는 A의 필드와 B의 필드 둘 다 가지고 있다.

- extends?

상속의 키워드로 부모의 맴버변수를 상속받은 자식은 계속 확장된다.

- 상속의 특징

1. 자식은 부모의 모든 멤버를 상속받는다. (생성자 제외)

2. 자식의 멤버 개수는 부모의 개수보다 같거나 많다.

3. 부모의 변경은 자식에 영향을 미치지만, 자식의 변경은 부모에 영향을 미치지 않는다.

※ Java는 단일상속만 허용한다.

오버로딩(new)

이름만 같은 "새로운 메소드"를 정의하는 것

- 오버로딩의 조건

1. 메소드 이름이 일치 해야 한다.

2. 매개변수의 갯수 또는 매개변수의 타입이 달라야한다.

오버로딩 된 메소드 사용시, 전달된 값의 타입 혹은 개수를 알아서 구분하여 알맞은 메소드가 자동으로 호출된다.

오버라이딩(change, modify)

1. 상속받은 부모의 메소드를 자식클래스에서 자신에 맞게 변경하는것

2. "재정의"를 했기 때문에 자식클래스에서 호출할 때 재정의된 메소드가 호출된다.

3. 구현부(내용)만 변경이 가능하다.

- 오버라이딩 조건

1. 선언부(리턴타입, 메소드명, 매개변수 목록)가 부모클래스의 메소드와 일치 해야 한다.

2. 접근 제어자를 부모 클래스의 메소드보다 좁은 범위로 변경할 수 없다.

(public, protected, default, private)

3. 예외는 부모클래스의 메소드 보다 많이 선언할 수 없다.

- 오버라이딩과 오버로딩의 차이점?

1. 오버로딩은 **상속과 전혀 관계가 없으며** 이름만 같은 다른 메소드를 만드는 것.

따라서 선언부인 매개변수의 갯수 또는 타입을 다르게 만들어야 한다.

2. 오버라이딩은 **상속을 받았을 경우만 가능**하고, 메소드의 구현부인 내용만 변경가능하다.

따라서 선언부는 변경불가능 하고 부모메소드와 일치 해야 한다.

 # super( )

부모생성자 이다.

1. 자식생성자에서 첫줄에 항상 super( )를 통해 부모생성자를 호출 해야한다.

why? 자식의 생성자는 항상 자신이 선언한 것에 대해서만 초기화 해야 한다.

부모클래스의 인스턴스 변수는 부모생성자가 초기화하는게 정석이다.

2. 자식생성자의 첫줄에 super( )를 써주지 않으면 **컴파일러가 자동으로 삽입**한다.

 # 접근 권한 제어자

다른 패키지 혹은 다른 클래스에서 해당 필드에 접근할 수 있는 범위(권한)을 설정해주는 키워드들

접근제어자는 **4개 중에 하나만 사용가능**

같은 클래스 < 같은 패키지 < 다른 패키지의 자손 클래스 < 전체

- 접근 제어자의 역할

1. 외부로 부터 데이터를 보호하기 위해

2. 외부에 불필요한 정보를 노출시키지 않고 내부적으로만 사용되는 부분을 감추기 위해

따라서 접근제어자의 범위는 좁을수록 좋다.

제어자	같은 클래스	같은 패키지	자손 클래스	전체
public	O	O	O	O
protected	O	O	O	-
default	O	O	-	-
private	O	-	-	-

- private

접근제어자가 **private로 설정된 필드는 직접 접근할 방법이 하나도 없기 때문에**

무조건 **public으로 설정된 getter, setter** 메소드를 통해서 **간접접근 한다.**

getter = 값 사용, setter = 값 대입

- private의 역할

1. private를 이용하여 **외부에서 직접 접근을 못하도록 막고** setter를 통해 **간접 접근**한다.

2. 메소드 또는 변수에 접근제어자를 private로 설정하여 사용범위를 줄인다.

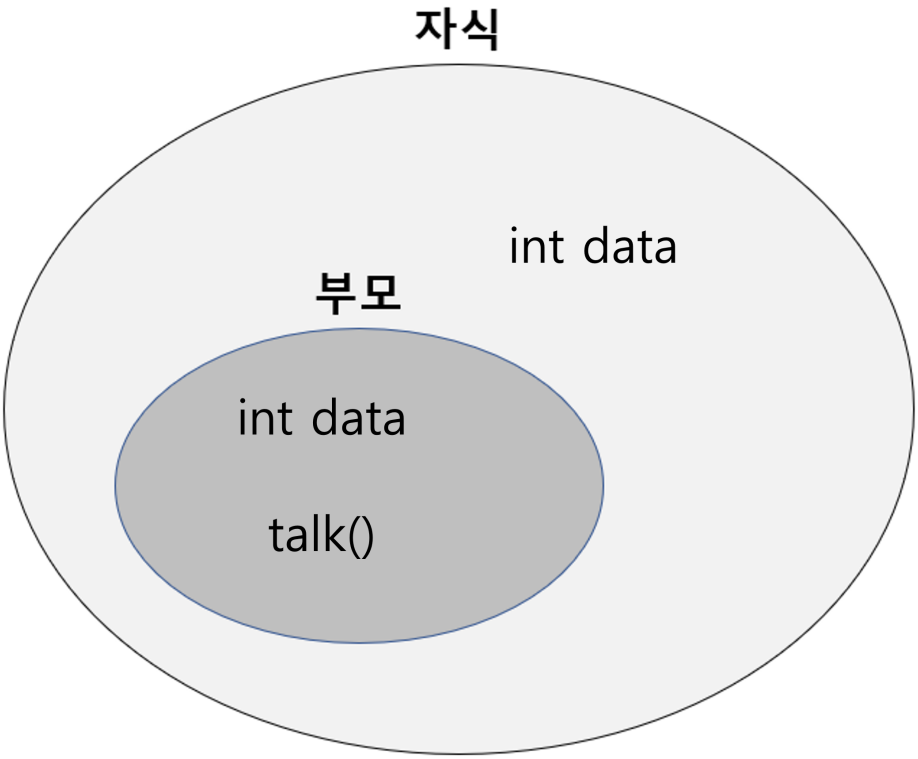
그렇게 하면 코드수정 시 범위가 class 내로 줄어들기 때문에 **유지보수가 편하다.**

 # 클래스 배열(참조변수 배열, 객체배열)

객체를 여러개 선언해야 하는 경우 배열 타입으로 한번에 선언 후 사용

각 객체들은 규칙성이 없기 때문에 규칙성을 부여하기 위해서 사용한다.

상속 (부모와 자식)



객체배열