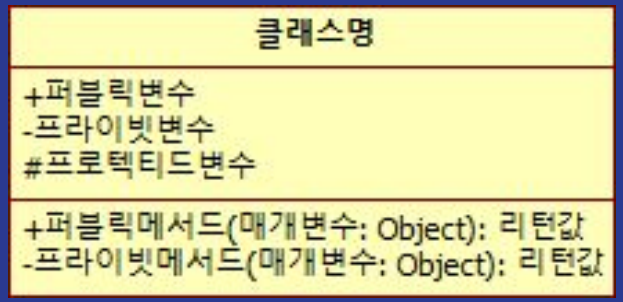
클래스 다이어그램

클래스 정의



①상단: 클래스명

②중단:데이터(멤버변수)

☞데이터 타입

-(Private): 외부에 노출이 되지 않는 한정자

+(Public): 외부에 노출이 되는 한정자

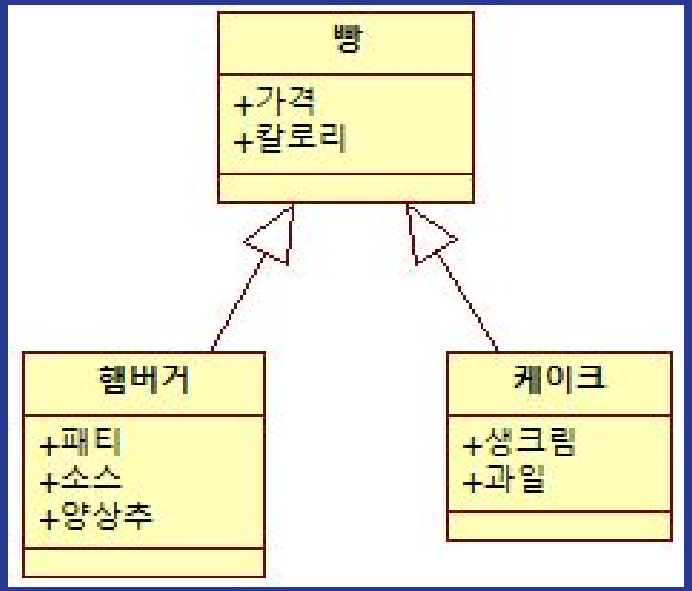
#(Protected): 클래스나 상속된 클래스에서 접근가능한 한정자

③하단:행동양식(메서드)

클래스 다이어그램

클래스 간 관계

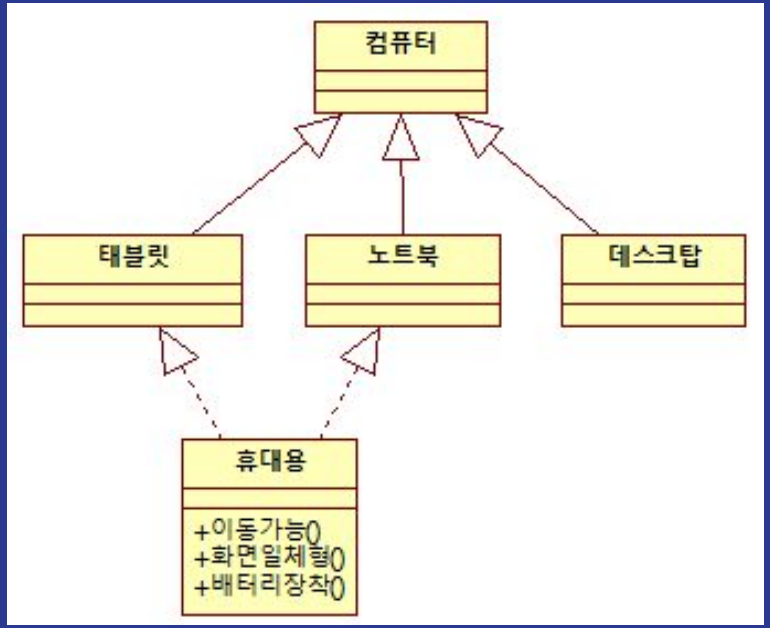
Generalization(일반화 관계) : 일반적인 것(동물)에서 특화된 것(사자)과의 관계를 나타낸다.
보통 상속을 표현한다.



클래스 다이어그램

클래스 간 관계

Realization(실체화 관계) : 인터페이스와 그것을 구현한 것과의 관계를 나타낸다.



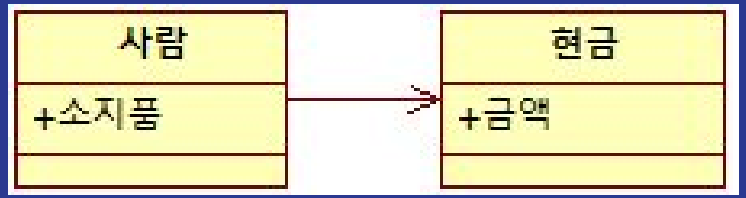
클래스 다이어그램

클래스 간 관계

Association(연관 관계) : 한 객체가 다른객체를 소유하거나 파라미터로 객체를 받아서 처리하는 관계를 나타낸다.

association :
- 한 객체가 다른객체를 소유(사용)하거나, 참조하여 사용할때
- 단방향과 양방향이 존재한다.

단방향: 클래스간의 관계가 "->" 이렇게 구현이 되며, 화살표의 대상은 자신을 가리키는 클래스의 존재여부를 알지 못한다.



양방향: 클래스간의 관계가 "-" 로 구현되며 서로 연관이 되어있다.

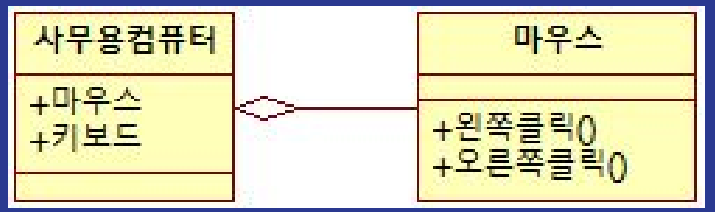
클래스 다이어그램

클래스 간 관계

Association(연관 관계) : 한 객체가 다른객체를 소유하거나 파라미터로 객체를 받아서 처리하는 관계를 나타낸다.

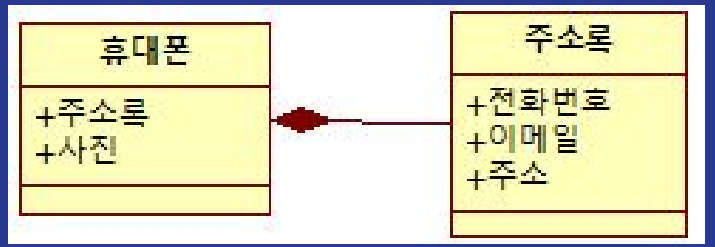
aggregation(집합) :

메인 클래스가 삭제될시 대상 클래스는 같이 삭제가 안됨 (라이프 사이클이 다름) 분리가 되도 독립적으로 동작됨 약한 결합



Composition(합성) :

메인 클래스가 삭제될시 대상 클래스도 같이 삭제가 됨(라이프 사이클이 동일) 분리가 되면 의미가 없어짐
강한결합



클래스 다이어그램

클래스 간 관계

Dependency(의존관계) : 한 객체가 다른객체를 소유하지는 않지만, 다른객체의 변경에 따라서 같이 변경을 해주어야 한다.

(1) 다른 객체를 파라미터로 받아서 그 메서드를 사용한다.
(2) 객체의 메서드 안에서 다른 객체를 생성해서 리턴한다.