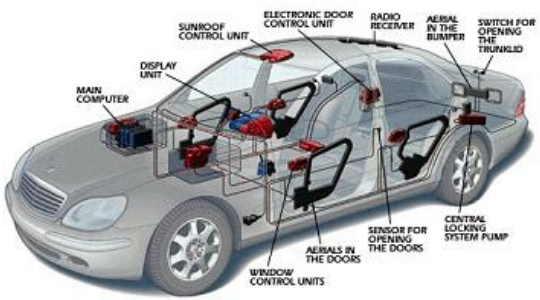
객체지향 프로그래밍 1

객체 Vs 클래스

- **클래스는** 여러 유사 **객체**들이 공통적으로 갖는 속성이나 행위를 기술하는 **명세 장치**
- **클래스의 특성**
  - 클래스는 고유한 이름을 가진다.
  - 클래스는 속성을 지닌다.
  - 클래스는 잘 정의된 행위를 가진다.
- **객체는 캡슐화와 정보은닉 도구이다.**
  - 모듈화 향상
  - 추상화 향상
  - 재사용성 향상
  - 유지보수 용이

- 객체는 관련된 데이터와 기능의 묶음(모듈)
- 객체 = 데이터 + 기능(함수)
- 프로그램 = 객체 + 객체 + … + 객체

'자동차' 객체



엔진, 몸체
브레이크

데이터 (attribute)

운행
정지

기능 (behavior)

- **객체의 모태는 클래스이기 때문에 클래스의 특성과 유사함.**
- **객체는 유일한 식별자를 가져야 한다.**
  - 객체 참조(Object Reference) : 메모리 주소 저장

클래스

클래스의 특성

❖ 클래스는 고유한 이름을 가진다.

- 특정 도메인에서 클래스가 중복될 수 없기 때문에 클래스도 다른 클래스와 구별되기 위한 고유한 이름을 갖는다.

❖ 클래스는 속성을 지닌다.

- 클래스는 의미 있는 정보 저장소 역할을 하기 위해 속성을 내포한다.
- 속성(Property 또는 Attribute)을 자바 언어와 같은 구현 관점에서는 상태 변수, 멤버 변수 혹은 멤버 데이터 등으로 표현한다.
- 클래스는 상태(값)을 갖지는 않고 다만 속성 선언만 할 뿐이다.
  상태는 개별 객체들이 갖게 된다.

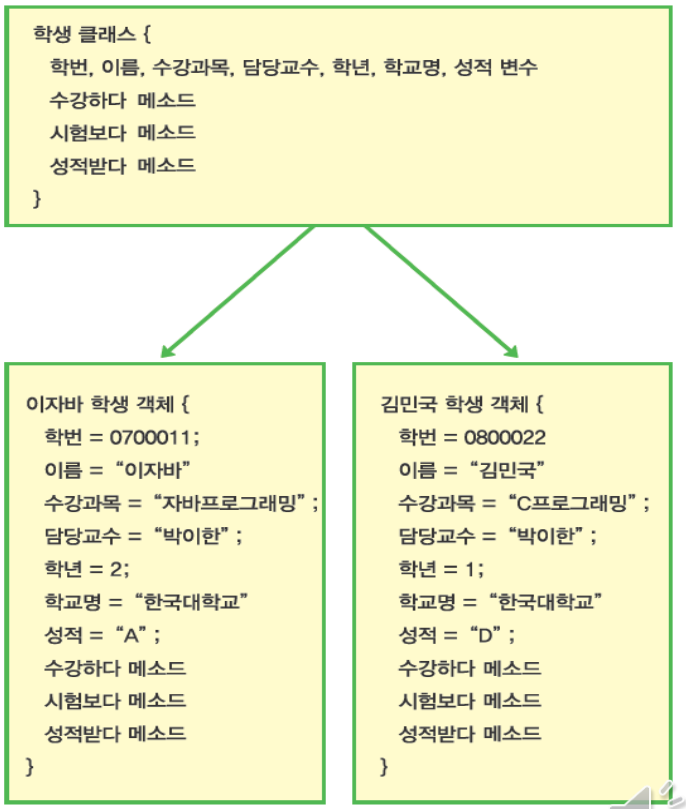
❖ 클래스는 잘 정의된 행위를 가진다.

- 행위란 클래스가 내포하고 있는 속성들을 사용하여 처리하는 기능을 의미한다.
- 행위를 구현 관점에서 표현하면 메소드, 멤버 함수라 할 수 있다.

객체지향 개념	객체지향 분석/설계	객체지향
객체	객체	객체
속성(Attribute)	속성, 프로퍼티(Property)	데이터(Data), 멤버 데이터(Member Data), 멤버 변수(Member Variable)
행위(Behavior)	메소드(Method)	오퍼레이션(Operation), 멤버 함수(Member Function)

클래스와 객체

- 클래스와 객체의 관계
  - ❖ 클래스는 틀이고 객체는 실사례이다.
  - ❖ 클래스는 상태가 없고 객체는 있다.
    - 클래스는 속성에 대한 선언만 내포하고 있는 반면에 객체는 선언된 각 속성들에 대해 값을 지닌다. 이러한 값을 객체의 상태라고 표현한다.
  - ❖ 클래스와 객체는 동일한 속성수와 메소드 수를 갖는다




```
//고객(Customer) 클래스 정의
class Customer{
        String userId;
        String password;
        String name;
        String ssn;
        String birthDate;
        String homeAddress;
        String homePhone;
        String homeZipCode;
        String companyName;
        String deptName;
        String officeAddress;
        String officePhone;
        String officeZipCode;
        String email;
        public void addCustomer(){...};
        public void updateCustomer() {...};
        public checkMember(String id, String password){...};
        public void changeAddress() {...}
        //...
}
// Customer 클래스로부터 c1 객체 생성
Customer c1 = new Customer("1234", "cust01", "강사랑", "8012102951417", "19801210", "서울시 중랑구 면목동 39-2",
                           "492-2632", ""131-702", "", "", "", "", "", "lovekang@hanmail.net");
```

클래스 표기법

UML에서 클래스 표기

- 3 행(row)으로 구성된 박스(Box)로 표기한다.
- 클래스 다이어그램의 핵심 구성 요소로서 클래스명, 속성, 행위(함수)로 구성된다.

속성

- 가시성(Visibility), 속성명, 데이터 타입으로 순으로 표기되며 초기값은 선택적으로 지정할 수 있다.
- 가시성_속성명 : 데이터_타입 , 가시성_속성명 : 데이터_타입 = 초기값

가시성 (visibility)

- 객체지향 개념의 정보은닉(Information Hiding)을 설계하기 위한 도구로서 정보나 행위의 공개 여부를 결정할 수 있다.
- Public : 해당 클래스의 모든 정보를 공개한다. UML에서는 '+' 로 표기 (예) +productName:String
- Private : 클래스의 정보를 외부에 공개하지 않고 내부에서만 사용. UML에서는 '-' 로 표기. (예) –productId:String
- Protected : 상속받은 서브 클래스에게만 공개하고 외부에는 공개하지 않는다. UML에서는 '#'로 표기한다. (예) #amount:float

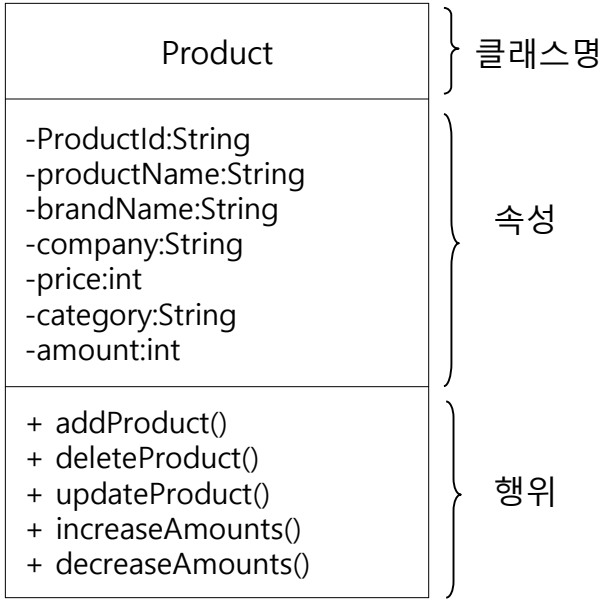
행위(behavior)

- 클래스의 행위는 가시성, 행위명, 매개변수, 데이터 타입 순으로 표기.
- 예) 가시성 행위명 (매개변수) : 데이터_타입

  \+ processOrder (orderId : String)

  \+ processOrder (orderId : String, userId : String)
- 행위 가시성 Public('+'), Private('-'), Protected('#'), friendly('~')

(가) 클래스 구성요소에 의한 클래스 표기



(나) 클래스 명에 의한 클래스 표기

가시성(visibility)

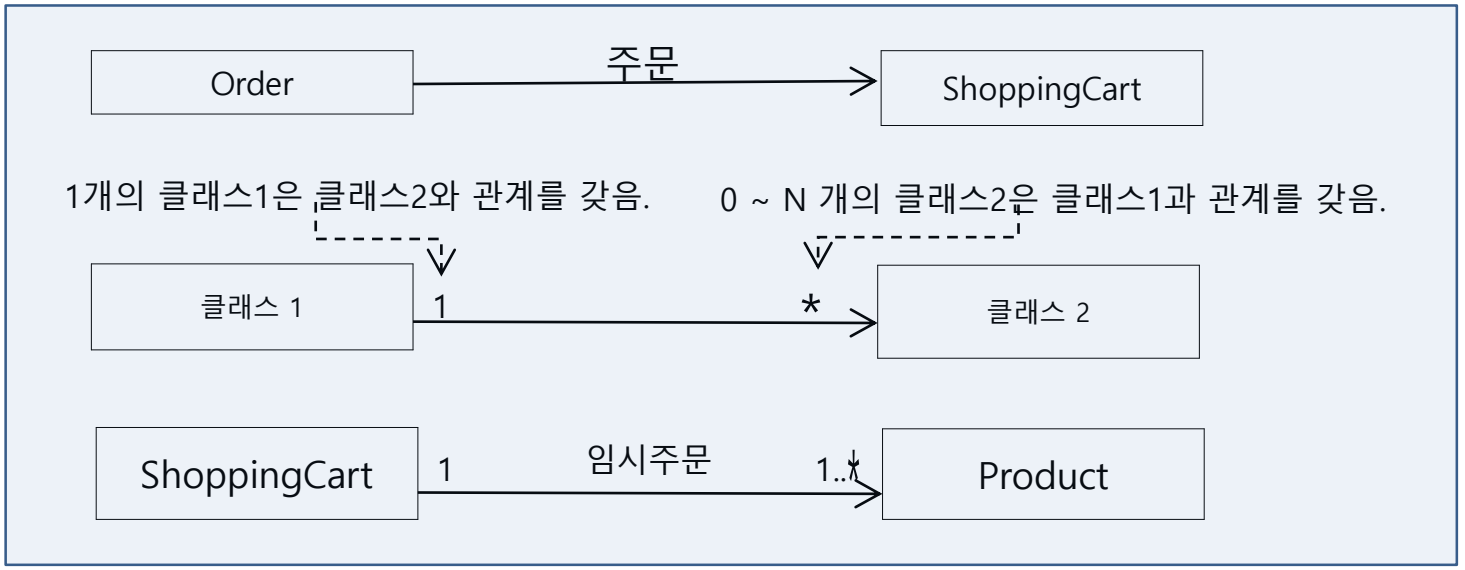
- 접근 제한자(access modifier)

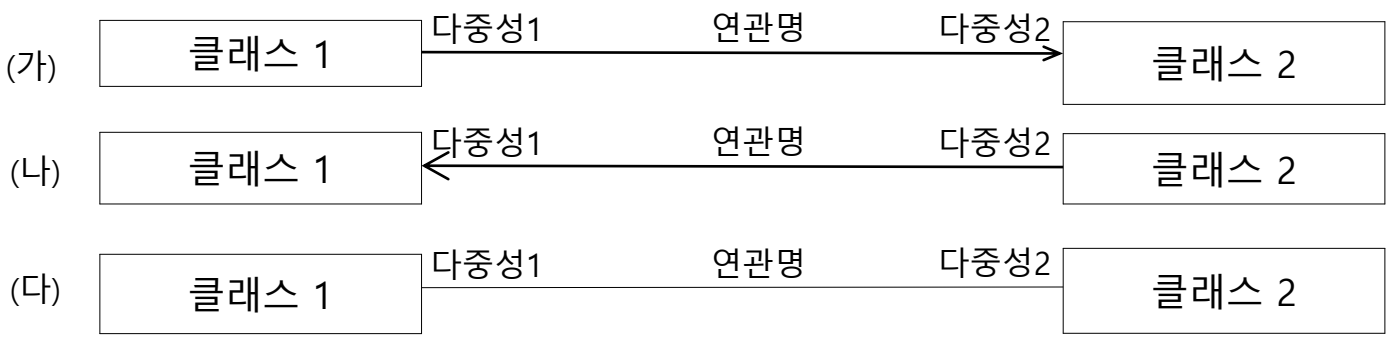
public	모든 클래스에서 접근이 가능함을 의미합니다.
protected	동일 패키지에 속하는 클래스와 하위 클래스 관계의 클래스에 의해 접근이 가능하다는 의미입니다.
private	클래스 내에서만 접근이 가능하다는 의미입니다.

종류	클래스	하위 클래스	동일 패키지	모든 클래스
private	O	X	X	X
(default)	O	X	O	X
protected	O	O	O	X
public	O	O	O	O

클래스들 간의 관계 표기법

- 클래스 다이어그램의 연관(Association), 포함(Aggregation, Composition), 상속(Inheritance) 관계 표기를 이용하여 클래스 간의 관계를 설계할 수 있다.
  - 연관관계(Association)는 클래스들 간에 요구하는 행위를 사용하는 관계를 의미한다.





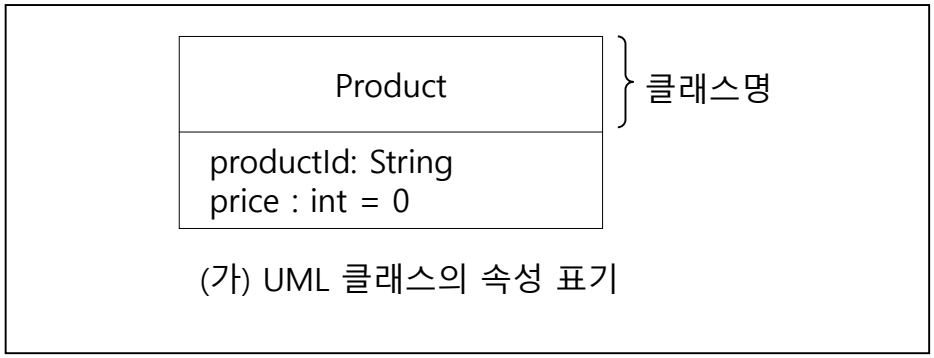
클래스 구현

- 자바에서 클래스 코드는 가시성, class 예약어, 클래스명, 중괄호({ })로 선언하며 중괄호 사이에 속성 및 행위(함수)를 구현한다

가시성 → public, 클래스명 → Product

```
public class Product  {

}
```

- **속성**



(가) UML 클래스의 속성 표기

```
public class Product  {
   String   productId;
   int        price = 0;

}
```

(나) 자바 클래스의 속성 코드

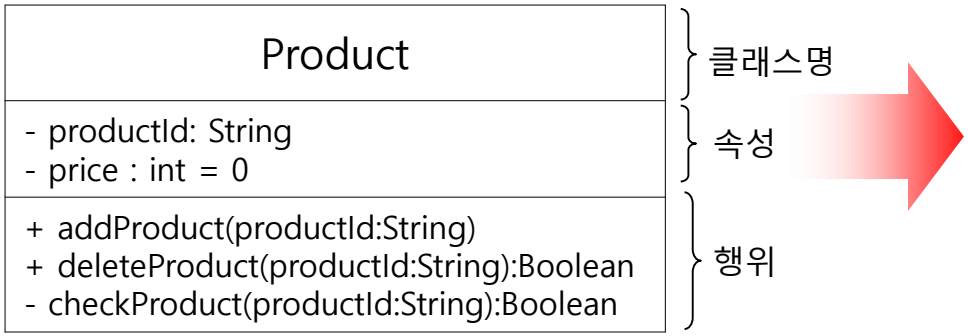
Payment 클래스 → Payment

Delivery 클래스 → Delivery

```
public class Order  {
   String        orderId;
   Payment     paymentType;
   Delivery     deliveryType;
}
```

클래스 구현

- **행위 (behavior) :** 가시성, 반환값의 데이터 타입, 행위명(함수명), 매개변수, 중괄화({}) 순.



(가) UML 클래스의 행위 표기

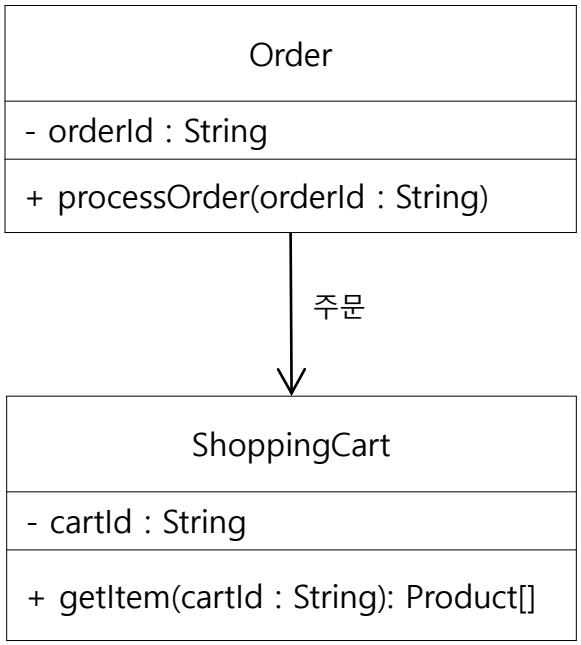
```
public class Product  {
   String   productId;
   int         price = 0;

   public void addProduct(String productId) {
    …
   }

   public Booelan deleteProduct(String  productId) {
    …
   }

   private Boolean checkProduct(String productId) {
    …
   }

}
```

(나) 자바 클래스의 행위 코드

클래스 구현

연관관계 구현

- 자바에서 클래스들 간의 연관관계는 함수 호출 형태로 구현된다.
- 호출하는 클래스의 특정 행위에서 연관관계를 갖는 다른 클래스의 특정 행위를 호출할 경우를 의미한다.
- 포함 관계 (has a relationship)



(가) UML 클래스 연관관계 표기

```
public class Order  {
   String    orderId;
   ShoppingCart sCart;

   public void processOrder(String orderId) {
      ShoppingCart sCart = new ShoppingCart();
      Product[] product = sCart.getItem(orderId);
      …
   }
}
```

```
public class ShoppingCart  {
   String    cartId;

   public Product[] getItem(String cartId) {
     …
   }
}
```

(나) 자바 클래스 연관관계 코드