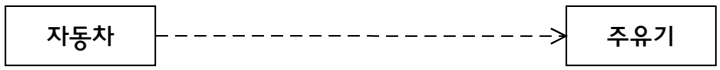
6. 의존(dependency)

- **의존**은 어떤 클래스가 다른 클래스를 **참조**하는 것이다.
- **의존**은 어떤 클래스가 **다른** 클래스의 **객체**를 **메서드** 내에서 **사용**하는 관계(**매개변수, 메서드 호출**)
- **의존**은 한 클래스의 변화가 다른 클래스에 영향을 주는 관계이다.**(역은 성립하지 않는다)**
- **의존** 관계는 **사용(using)** 관계를 나타낸다.**(종속 관계)**
- **의존**은 한 클래스가 다른 클래스를 **속성(멤버변수)으로 가지는 않는 경우**이다.

예제 1	어떤 클래스가 **다른** 클래스의 **객체**를 **메서드**의 **매개변수**로 받아 **사용**하는 경우

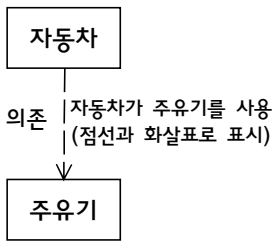
〈의존(dependency)〉



↓Java 원시코드

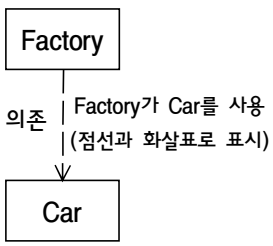
〈Java 원시코드〉

```
class 자동차
{
  public void 주유하다(주유기 p){ p.휘발유(); }  //객체를 메서드의 매개변수로 받아 사용
}
class 주유기
{
  public void 휘발유( ){ System.out.println("휘발유 주입"); }
}
public class Test
{
  public static void main(String args[])
  {
    자동차 car = new 자동차();
    car.주유하다(new 주유기());
  }
} //출력 : 휘발유 주입
```



예제 2	어떤 공장에서 버스 100대를 생산한 경우(**메서드에서 객체 생성 및 반환**)

```
class Factory
{
  public Car makeCar(){ return new Car(); }      //객체 생성 및 반환
  void how_many()
  {
     Car m = makeCar();
     m.make_bus();
  }
}
class Car
{
  int bus = 100;
  void make_bus(){ System.out.println("버스 : " + bus); }
}
public class Test
{
  public static void main(String args[])
  {
     Factory ft = new Factory();
     ft.how_many();
  }
} //출력 - 버스 : 100
```



의존에서 참조 형태	• 메서드 내에서 대상 클래스의 객체 생성 • 메서드 내에서 대상 클래스의 객체 반환 • 메서드 내에서 대상 클래스의 객체 사용 • 메서드 내에서 대상 클래스의 메서드 호출 • 객체를 메서드의 매개변수로 받는 것 등	**의존** 관계의 **특징**은 해당 객체 참조를 계속 유지하지 않는 원리이다. **(일시적 참조)**

• 메서드는 스택영역에서 생성/소멸되는 방식으로 실행된다.
• 해서, 메서드 실행이 종료되면 클래스 사이의 **의존** 관계는 종료된다.**(일시적 참조)**

의존 관계 예	• 요리사는 요리도구에 의존한다. 요리도구가 바뀌면 요리방법도 바뀐다. • 세금은 세금법에 의존한다. 세금법이 바뀌면 세금 계산 방법도 바뀐다.

기출문제 분석

1. 클래스 간의 관계에 대한 설명으로 옳지 않은 것은? [2015년 국가 7급]

① 연관 관계의 다중성(multiplicity)은 두 클래스의 연관 관계에서 실제로 연관을 가지는 객체의 수를 나타낸다.

② 집합(aggregation) 관계는 전체와 그 객체의 구성요소 사이의 관계를 나타내며, whole-part 관계를 나타낸다.

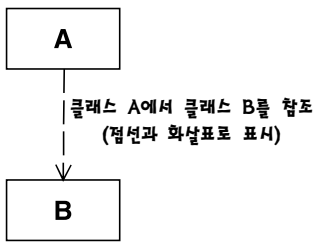
③ 일반화(generalization) 관계는 일반적인 클래스와 구체적인 클래스 간의 관계 즉 상속 개념을 나타내며, is a kind of의 관계를 나타낸다.

④ 의존(dependency) 관계는 하나의 클래스에 있는 멤버 함수의 인자가 변해도 다른 클래스에 영향을 미치지 않는 관계를 의미한다.

♤ **의존**

- **의존**은 어떤 클래스가 다른 클래스를 **참조**하는 것이다.
- **의존**은 어떤 클래스가 **다른** 클래스의 **객체**를 **메서드** 내에서 **사용**하는 관계(**매개변수, 메서드 호출**)
- **의존**은 한 클래스의 변화가 다른 클래스에 영향을 주는 관계이다.**(역은 성립하지 않는다)**
- **의존** 관계는 **사용(using)** 관계를 나타낸다.**(종속 관계)**

```
class A //예제 : 클래스 A는 클래스 B를 사용(A, B는 의존 관계)
{
  public void mooA(B bp){ bp.mooB(); }        //메서드 인수가 클래스 B의 객체
}
class B
{
  public void mooB( ){ System.out.println("의존관계"); }
}
public class Test
{
  public static void main(String args[]){
    A ap = new A();
    ap.mooA(new B());
  }
} //출력 : 의존관계
```



정답 : ④

2. UML의 클래스 다이어그램에서 클래스 사이의 관계에 대한 설명으로 옳지 않은 것은? [2021년 계리]

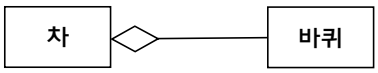
① 일반화(generalization) 관계는 일반화한 부모클래스와 실체화한 자식클래스 간의 상속 관계를 나타낸다.
② 연관(association) 관계에서 다중성(multiplicity)은 관계 사이에 개입하는 클래스의 인스턴스 개수를 의미한다.
③ 의존(dependency) 관계는 한 클래스가 다른 클래스를 참조하는 것으로 지역변수, 매개변수 등을 일시적으로 사용하는 관계이다.
④ 집합(aggregation) 관계는 강한 전체와 부분의 클래스 관계이므로 전체 객체가 소멸되면 부분 객체도 소멸된다.

♤ **UML의 클래스 다이어그램**

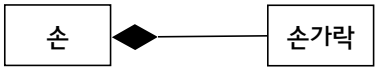
• 집합(aggregation) 관계는 **강한** 전체와 **부분**의 클래스 관계이므로 전체 객체가 소멸되면 부분 객체도 소멸된다.**(×)** → 집합은 **약한 전체 부분관계**이다.

집합 (aggregation)	• **약한 전체 부분관계** • 집합은 생명주기가 일치하지 않는다. • 부분은 자체적으로 존재할 수 있고, 없어질 수도 있다. • 집합은 집단화라고도 한다. • 그리는 방법 : **포함**하고 있는 **클래스 쪽**에 끝이 **빈 다이아몬드**로 표시 • 예 : 회사와 부서의 관계 – 회사에는 여러 부서가 존재한다.
합성[복합] (composition)	• **강한 전체 부분관계** • 합성은 생명주기를 공유한다. • 부분만으로는 스스로 존재할 수 없다. • 합성은 집합 관계보다 더 끈끈한 관계이다.(더 강한 집합) • 그리는 방법 : **포함**하고 있는 **클래스 쪽**에 **검은 다이아몬드**로 표시 • 예 : 손(hand)은 손가락을 포함하고 있다.

• **집합**(aggregation) : 차와 바퀴의 관계(생명주기가 불일치하는 경우)



• **합성**(composition) : 손과 손가락의 관계(생명주기가 일치하는 경우)



정답 : ④