12주차 3차시 객체지향 개발단계와 방법론

【학습목표】

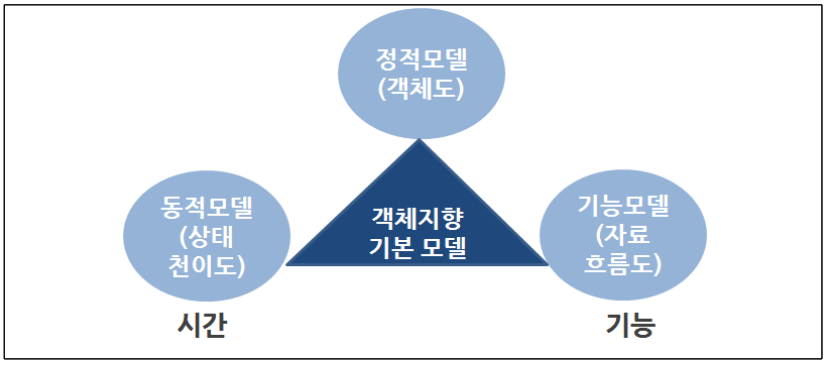
1. 객체지향방법에 의한 소프트웨어 개발을 3단계로 설명할 수 있다.
2. 객체의 정의와 자격을 설명할 수 있으며, 객체 사이의 관계를 분류할 수 있다.

학습내용1 : 객체지향단계와 방법론

1. 객체지향 개발단계의 개요

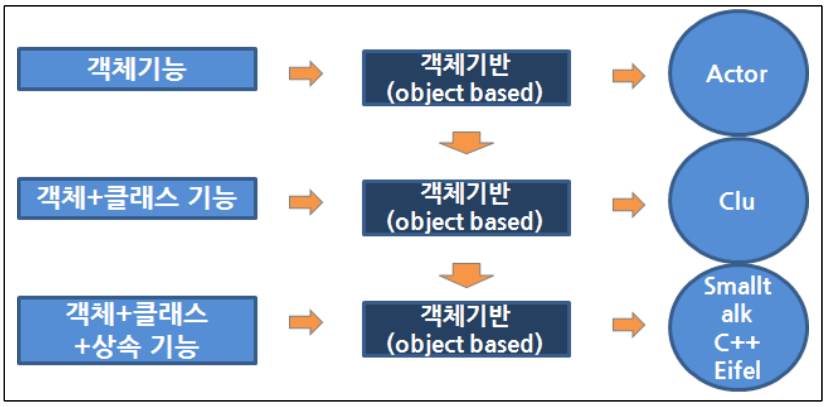
1) 객체지향방법에 의한 소프트웨어 개발은 객체지향분석 객체지향설계, 객체 지향 프로그래밍 등의 3 단계로 진행됨

2) 객체지향 분석
- 문제의 특성·정의한 결과에 의거하여 객체모델, 동적모델, 기능모델을 작성하여 실세계(real world)의 내용을 모델화(modeling)하는 단계임.
- 이 모델화는 시스템의 어떤 대상(객체모델)으로부터, 언제(동적모델), 어떤 일(기능모델)이 발생하는지 기술(description)하는 작업임.
- 객체모델화(object modeling)를 위하여 객체의 특징을 식별해서 객체 다이어그램(object diagram)을 작성함.
- 동적모델화(dynamic modeling)를 위해서 시간의 흐름에 따라 변하는 시스템의 상태를 표현하는 상태 다이어그램(state diagram)을 작성함.
- 기능모델화(function modeling)를 위해서 시스템 내에서 데이터의 변화과정을 자료흐름도(data flow diagram : DFD)를 작성함.

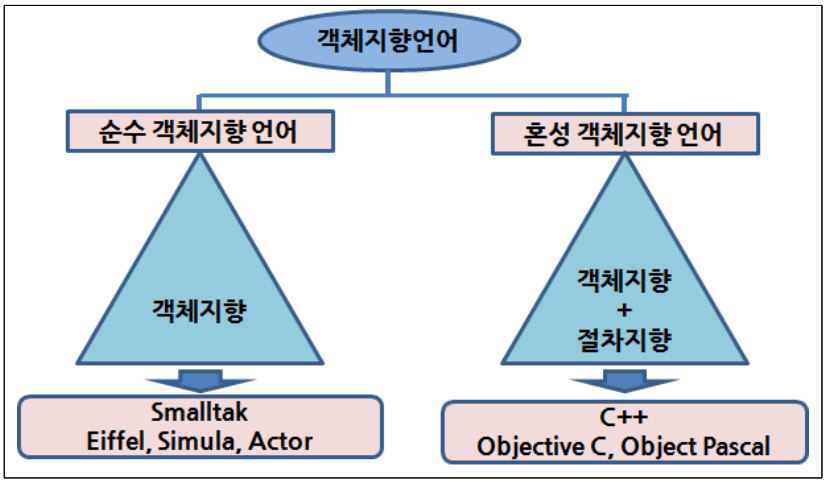


3) 객체지향 설계
- 정의한 객체를 구현하기 위해 자세히 설계하는 과정이므로 다음 사항을 정의해야 함.
    - 각 객체의 활동을 나타내는 연산의 명확한 정의.
    - 연산 처리 대상인 자료구조의 속성을 구체적으로 정의.
    - 자료처리에서 사용하는 알고리즘의 정확한 정의.
- 거대·복잡한 시스템은 설계 이전에 분할해야 함.

4) 객체지향 프로그래밍
- 설계단계에서 정의된 내용을 적정한 프로그래밍 언어로 구현하는 과정임.
- 전문적인 객체지향 프로그래밍 언어·객체지향 DBMS 사용이 효과적임.
- 다음 그림은 Peter Wegner가 객체 지원 언어를 구분한 내용



- 객체지향 언어를 구분하면 다음과 같이 나타낼 수 있음.



2. 객체지향방법론의 진화

1) 소프트웨어 공학의 발전
- 구조적 프로그래밍(1970년대 초반) 구조적 코딩, 추상화, 하향식 프로그래밍, 단계적 세분화, 정보은익 등등.
- 설계방법론(1970년대 중반) Yourdon & Constantine 중심의 구조적 설계, Jackson의 JSP 설계방법, Warnier-Orr 설계방법 등.
- 분석방법론(1970년대 후반) DeMarco·Gane·Sarson 중심의 구조적분석, SoftTech 의 SADT 등등.
- 자동화 도구 및 객체지향설계·프로그래밍(1980년대 초반) 코드 자동 생성, 객체지향 설계, 소프트웨어공학 도구, PSL/PSA 등이 연구됨. Smalltalk, Ada, Modula-2 등이 본격적으로 사용되기 시작함.
- 컴퓨터 응용 소프트웨어 공학(1980년대 후반) 프로그래밍 환경, 사용자 인터페이스 관리시스템, 분석용 대화식 그래픽 도구 등이 활발히 연구됨.
- 객체지향 소프트웨어 공학(1990년대) 객체지향 언어, 객체지향 분석·설계·프로그래밍, 소프트웨어 재사용, 등등.

2) 객체지향방법론의 발전
- 개념도입·연구단계(1980~1990년대 초반까지) 객체지향 개념을 도입하여 기존의 다른 기법들과 혼합하여 시험적인 적용·

테스트하면서 초보적인 연구가 주류를 이룸.
- 개별연구단계(1990년대 중반부터) Booch94, OMT(object modeling technique), Fusion처럼 연구자들이 개별적으로 적용하고 보완 연구를 하던 시기임. 이 시기에 대표적으로 사용하던 방식은 Booch와 Rambaugh의 방식임.

① 통합단계(1995년도 이후)
- 「문서버전 0.8」(1995년 10월 발표)은 1994년 부치방법의 개발자인 Booch, OMT의 개발자인 Rambaugh, Use case(Jacobson이 개발한 Objectory 방법) 개념을 통합하여 만든 것임.
- 「문서버전 0.91」(1996년 9월 발표)은 자콥슨이 위 프로젝트에 정식으로 참여해서 만든 것임. 이 과정을 거치면서 UML(unified modeling language)이 탄생함.
- 「문서 1.0」(1997년) 위의 세 사람 이외에 다수의 학자들이 UML 개발에 참여하여 만든 것임.
- 이 「문서 1.0」은 OMG(object management group : 국제 객체지향 표준화 기구) 의 표준안으로 공인을 받음.
- 한편 UML과 경쟁관계에 있는 「OPEN 콘소시엄」 의 「OML(open modeling language)」 과「OPEN 방법론」이 있음.
- 이 그룹은 1995년부터 시작하여 객체지향방법론의 메타모델 개발을 목표로 하고 있음.
- 이 연구 결과가 COMMA (common object methodology metamodel architecture) 임.

3) UML
- 람바우, 부치, 자콥슨 등의 선구자적 학자들의 연구결과를 수정보완 통합함으로써 다양한 표현에 무리가 없도록 하여 소프트웨어 개발과정의 산출물(products)들을 기술(description)하고, 시각화(visual) 할 수 있도록 만든 문서화(documentation) 방법임.

- UML의 메리트는 숙지하기 용이하고, 확장적용이 용이하며, 다양한 표기법 을 통하여 여러 가지 방법론의 특징을 효과적으로 표현 할 수 있다는 점임.
- UML에서 사용하는 다이어그램은 다음과 같음.

① 클래스 다이어그램(class diagram)
② 유즈케이스 다이어그램(use case diagram)
③ 상태 다이어그램(state diagram)
④ 시퀀스 다이어그램(sequence diagram)
⑤ 패키지 다이어그램(package diagram)
⑥ 협력 다이어그램(collaboration diagram)
⑦ 배치 다이어그램(deployment diagram)
⑧ 활동 다이어그램(activity diagram)

학습내용2 : 객체의 정의와 상속

1. 객체의 정의와 자격

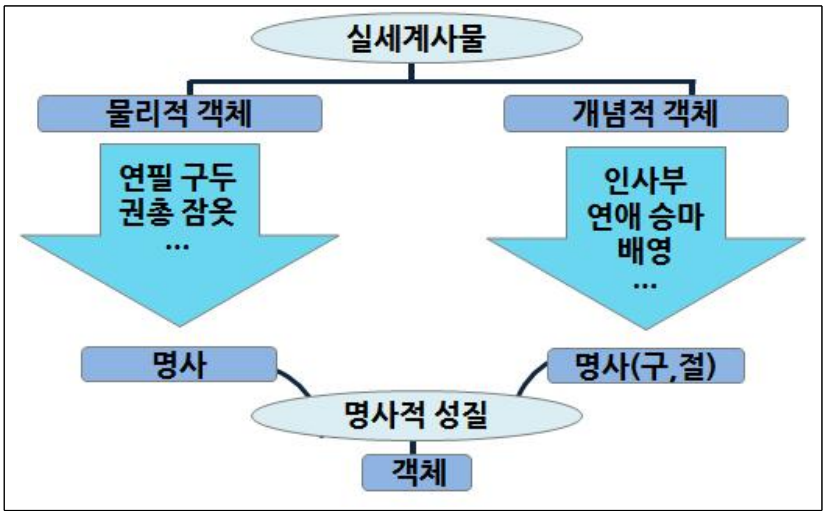
1) 현실세계에서 인간이 식별할 수 있는 「실세계 (real world)의 사물」 들을 객체 (object)라고 함.

2) 모든 객체는 「객체이름(object name)」 을 가짐(핸드폰, 자장면, 복숭아 등등) 이처럼 이름이 붙여진 대상은 「명사(noun)의 성질」 을 가짐.

3) 「명사적 성질을 가진 모든 사물」 은 객체가 될 수 있음. 그러므로 다음 사항들은 객체가 될 수 없음.
- 의태어(사뿐사뿐), 의성어(끼륵끼륵)
- 전치사(외에), 형용사(못생긴), 동사(....하다), 부사(게으르게), 감탄사(어머나)

3. 객체의 분류기준

1) 객체는 다음 그림과 같이 물리적 객체(physical object), 개념적 객체 (conceptual object)로 구분됨.



1) 결국 객체는 명사·명사구·명사절이거나, 물리적·개념적을 막론하고 「모두 명사의 성질을 가진다」라는 사실에 유념해야 함.

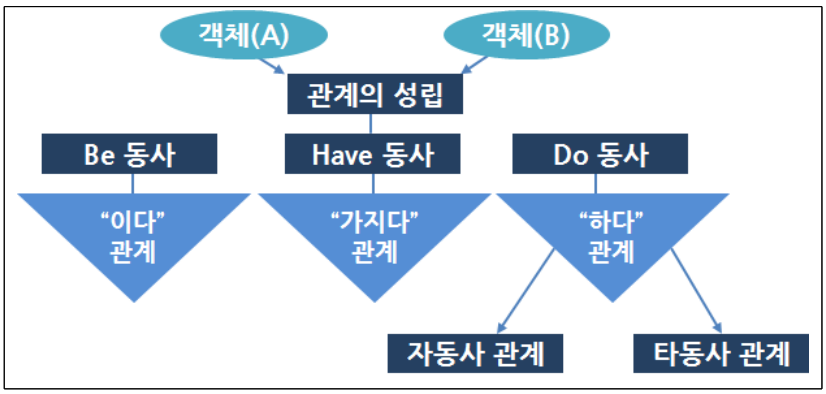
4. 객체 사이의 관련 개요

1) 독립적으로 존재하는 객체 사이에 필요에 따라서는 특별한 「관계(relation or association)」이 이루어짐.
- 어떤 순간의 만남으로 갑순이와 갑돌이가 사랑을 하게 된다면, 갑순이와 갑돌이 라는 객체 사이에는 「사랑한다」 라는 관계가 이루어지는 것임.

2) 일반적으로 「객체와 객체 사이의 관계」 는 「동사(verb)」 로 나타냄.

6. 객체 사이의 관계 분류

1) 객체 사이의 관계를 나타내는 「동사관계」는 대체적으로 다음 그림과 같이 세 가지 형태로 분류됨.



- 「이다 관계」「be 동사」의 성질을 나타내는 「~는 ~이다」의 관계임.
- 「가지다 관계」「have 동사」의 성질을 나타내는 「~는 ~를 가진다」의 관계 임.
- 「하다 관계」「DO 동사」의 성질을 나타내는「~는 ~를 한다」의 관계임.

2) 「하다 관계」는 「이다 관계」와 「가지다 관계」를 제외한 모든 「일반동사 관 계」를 의미함.
- 그러므로 「하다 관계」는 「일반동사 관계」라고 생각해야 함.

7. 이다 관계

1) 영어에서 「be 동사」는 「이다」라는 의미를 가짐.
- 그런 의미에서 「이다 관계」를 「is-a 관계」 혹은 「is 관계」라고도 함.

2) 일반적으로 「이다 관계」는 일반화(generalization)와 특수화(specialization)의 관계로 규정 가능함. 다음 그림을 참조.

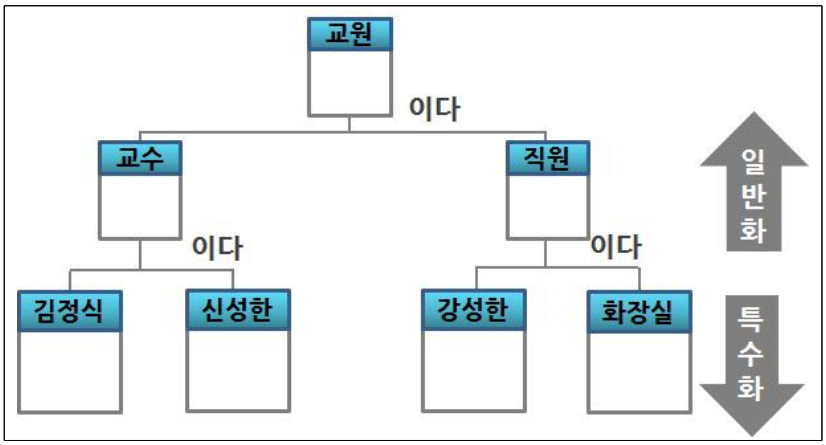
- 그림의 왼쪽에 나타낸 내용을 「김정식씨는 교수이다」라는 문장으로 표현했을 경우에 위쪽의 「김정식」이라는 객체는 아래쪽에 있는 객체 「교수」의 「특수화」에 해당함.
  - 아래쪽에 있는 객체 교수는 김정식이라는 객체를 일반화시킨 대상이 됨.

- 일반화(generalization) : 기존의 객체들 사이에 존재하는 유사성을 이용하여 새로운 추상화(abstraction) 객체를 만드는 것을 의미함.
- 특수화(specialization) : 상위클래스로부터 하위클래스를 만들어내는 것을 의미함.

3) 「이다 관계」에서특수화된 쪽의 객체는, 일반화된 쪽에 객체의 속성을 모두 상속(inheritance) 받음.

- 여기서 속성(attribute)은 객체가 가지고 있는 자료 혹은 특성을 의미함.

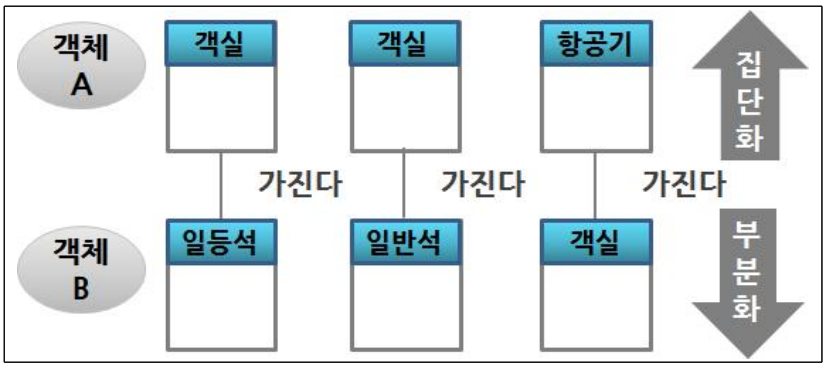
4) 「이다 관계」는 상속에 의한 계층구조 (inheritance hierarchy) 로 나타낼 수 있는 계층관계 (hierarchy relation) 로 간주할 수 있음.
<다음 그림을 참조.>



8. 가지다 관계

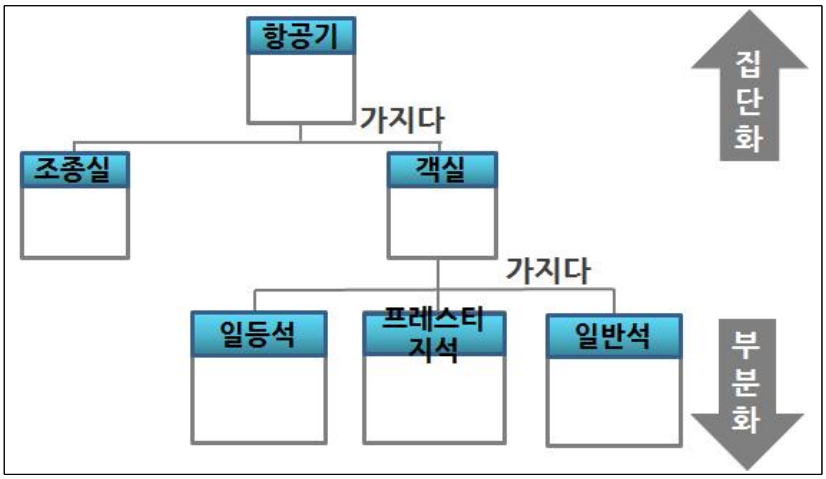
1) 「have 동사」는 「가지다」라는 의미를 함축하고 있음.
- 그러므로「가지다 관 계」를 「has-a 관계」 혹은 「has 관계」 라고 함.

2) 「가지다 관계」는 다음 그림에서 보듯이 「객체 B」 는 「객체 A」 의 일부분에 해당하기 때문에 「part-of 관계」 라고도 함.



3) 결국「가지다 관계」는 집단화 (aggregation)와 부분화(partialization)의 관계라 고 규정할 수 있음.
- 집단화 (aggregation)
  - 그림에서 왼쪽에 위치한 「객체 A와 B」 에서 「객실은 일등석을 가진다」 라고 표현할 때 위쪽의 「객실」 이라는 「객체 A」 는 「일등석, 프레스티지석, 일반석」 등의 「객체B」와 같은 부분들이 모여져서 집단화(aggregation)된 것임
  - 부분화(partialization) 아래쪽 「일등석」 이라는 「객체 B」 는 「객실」 이라는 「객체 A」 를 분해하여 부분으로 만든 부분화(partialization)된 것임.

4) 부분화된 쪽의 객체는 집단화된 쪽에 객체의 일부분이 됨을 알 수 있음.
- 그러므로 「가지다 관계」를 종합적으로 정리하면 다음 그림과 같음



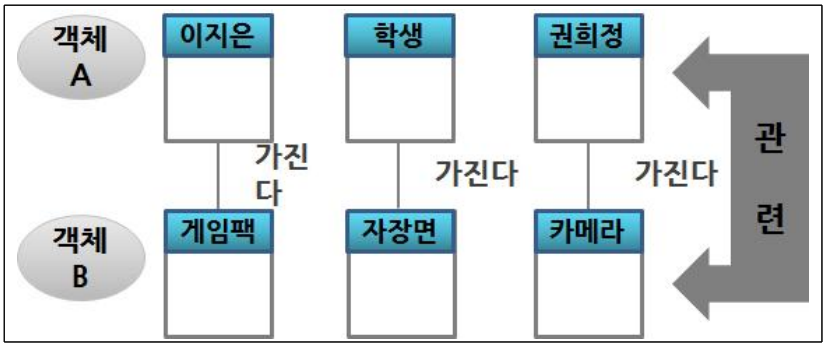
5) 「가지다 관계」를 식별하는 기준은 객체의 한쪽이 다른 한쪽의 일부분을 구성하는 요소가 되는지 여부로 판정하면 됨.

9. 하다 관계

1) 「DO 동사」는 하다라는 의미이기 때문에 「하다 관계」를 「does-a 관계」 혹은 「does 관계」라고 함
- 그러나 「이다 관계」 와 「가지다 관계」 를 제외한 모든 관계를 「하다 관계」 로 봄

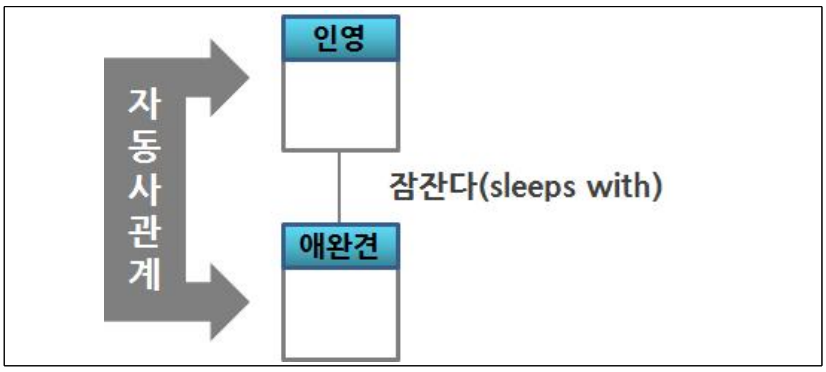
2) 결국「하다 관계」는 「서로 다른 성질을 가지고 있는 객체 사이의 특정한 관계」 를 맺어주는 「관련(association)의 관계」 라고 정의 가능함.

3) 아래 그림에서 왼쪽에 위치한 내용을 기초로 하여 「이지은이 게임팩을 원한다」라고 표현했을 경우에 위에 위치한 객체 「이지은」 은, 아래에 위치한 「게임팩」 과 전혀 상관이 없으나 「원한다」 라는 동사를 통해서 상호간에 관련을 맺음



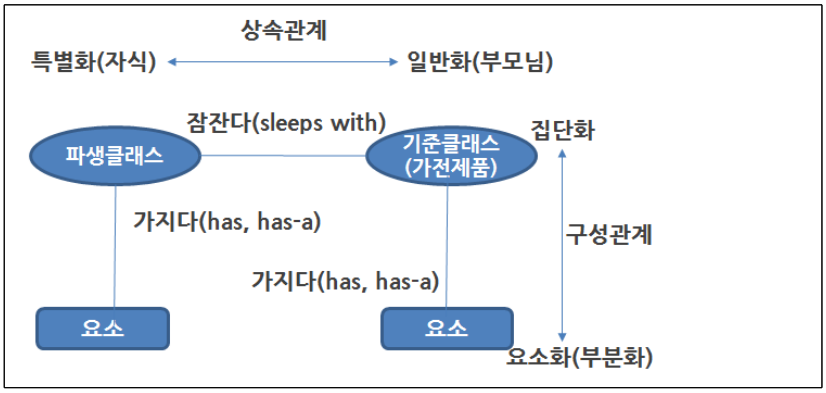
4) 관련(association)으로 표현되는 「하다 관계」 는 다음과 같이 두 가지로 구분됨.
- 자동사 관계(intransitive verb relationship)
  - 관계의 주체가 되는 객체는 명사 (noun)의 성질을 가짐.
  - 그러나 관계의 대상이 되는 객체는 전치사(preposition) 와 명사가 결합된 전명구(前名句)로서의 부사적용법(adverb)과 같은 성질을 가짐.
- 타동사 관계(transitive verb relationship) 관계의 주체가 되는 객체와 대상이 되는 객체가 모두 명사(noun)의 성질을 가짐.

- 다음 그림처럼 「인영」 이가 「애완견」 을 좋아하여 함께 놀다가 「인영이와 애완견이 같이 잠을 잔다」 라고 할 경우에 「인영」 이와 「애완견」 의 관계는 「자동사 관계」 로 볼 수 있음,



10. 상속 기법

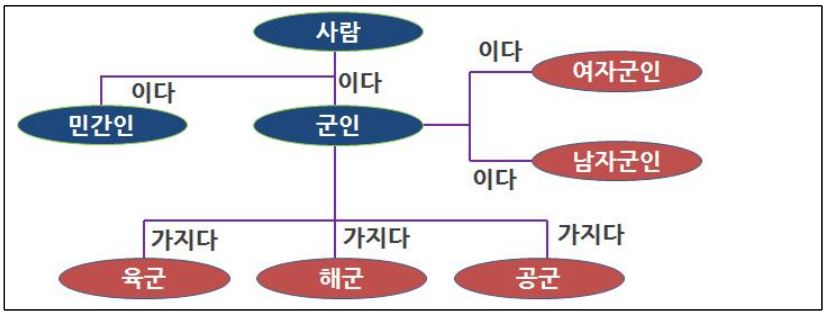
1) 클래스에서 관계 : 원칙적으로 클래스(class)는 성질상 「이다 관계」 , 「가지다 관계」 를 가지고 있음.
- 「이다 관계」는 「is」또는「is-a」로 표현됨.
  - 이 관계는 기준 클래스(base class)와 파생 클래스(derived class) 사이의 상속관계(inheritance relation)를 의미함.
- 「가지다 관계」는 「has」혹은「has-a」로 표현됨.
  - 이 관계는 클래스와 클래스를 구성하는 요소(member) 사이의 구성관계(composition relation)를 나타내는 것을 의미함.
- 「이다 관계」 와 「가지다 관계」 의 개념을 그림으로 요약하여 나타내면 다음과 같음.



2) 클래스 계층
- 기준 클래스와 파생 클래스 사이에 이다 관계가 성립되면 이들 사이에는 상속관계 (inheritance relation)가 성립됨을 의미함.
  - 예를 들어 「여자군인」 과 「남자군인」 은 「사람」의 일부임.
  - 이들 관계에서 「군인」 은 「사람」 의 기본속성을 상속받은 상태에서 자신들만의 특별한 속성을 추가하여 보유함.

- 한편 「여자군인」 은 「군인」 과 「사람」 의 모든 속성을 상속받은 상태에서 여군만의 특별한 속성을 추가하여 보유함.
- 이러한 관계는 상속이 진행되면서 더더욱 특별화(specialization)된 형태로 계속하여 진화함.

- 이들 관계를 요약하여 그림으로 나타내면 다음과 같음.



- 클래스 사이에 상속관계를 계층적으로 체계화해 나가면 기존의 속성을 효율적으로 진화시킬 수 있고, 기존속성과 확장속성의 구분이 된다는 장점이 있음.
- 상속을 명확히 이해하기 위하여 의복을 예로 들어 고려해 보기로 함.
  - 의복은 원래 사람의 부끄러운 부분을 가려주는 역할, 몸을 보호하는 기능을 위해서 만들어짐.
  - 이「의복」이 「운동복」,「작업복」,「평상복」,「연미복」 등의 용도로 특별화 되면, 이 특별화 된 하위의 파생 클래스 (자식(child))는 상위의 기준 클래스 (부모(parent)) 의 속성을 상속받은 상태에서 특별화 된 클래스의 속성을 추가하는 형식으로 차별화 됨.
- 상속(inheritance)은 기본적으로 사람의 경우 자식 (child : 파생 클래스)은 부모(parent : 기준 클래스) 의 속성을 물려받는데, 그 자식은 부모로부터 물려받은 속성에다가 자신만의 새로운 속성을 추가하여 자신의 속성을 구축하여 진화하는 식으로 이루어짐.

【학습정리】

1. 객체지향단계와 방법론을 이해한다.

2. 객체의 정의와 상속을 알아본다.

3. 상속 기법을 파악한다.