강의노트

객체지향 개발 방법론

❖ 학습안내

이번 시간의 학습내용과 학습목표를 확인해보세요.

▣ 학습내용

- 객체지향의 개념
- 객체지향의 구성요소
- 객체지향의 기본원칙

▣ 학습목표

- 객체지향의 개념을 설명할 수 있다.
- 객체지향의 구성요소들을 하나씩 설명할 수 있다.
- 객체지향 개발 방법론의 기본원칙에 대하여 설명할 수 있다.

❖ 학습내용

[1] 객체지향의 개념

1. 객체지향 접근

◈ 객체(Object)의 정의

- 실 세계에 **존재**하거나 **생각**할 수 있는 것
- 책상, 의자, 전화기 같은 **사물**은 물론이고 강의, 수강 신청 같은 **개념**으로 존재하는 것
- 사전에 나와 있는 **명사**뿐 아니라 **동사의 명사형**까지도 모두 객체
- 인간이 **생각**하고 **표현**할 수 있는 모든 것이 **객체**

◈ 객체지향 접근(Object-oriented Approach)

- 기능이나 데이터 대신 **객체**가 **중심**이 되어 시스템을 개발하고자 하는 접근

데이터(속성)와 그 데이터를 조작하는 메서드(함수)를 객체라고 하며
객체를 중심으로 모듈 을 구성

→이 객체 속에는 기능과 데이터가 모두 포함

모듈을 결합, 모듈 간에 연관성을 식별해 **모델링**한 뒤,
이를 기반으로 소프트웨어를 완성

전체 시스템을 여러 개의 **독립된 객체**로 정의하고
필요한 객체를 중심으로 **하나의 모듈(컴포넌트)**을 만듦

❖ 학습내용

[1] 객체지향의 개념

1. 객체지향 접근(계속)

객체지향 접근 방법의 특징

1 실 세계를 사람이 생각하는 방식으로 표현

- 실 세계의 문제를 사람이 생각하는 방식대로 자연스럽게 표현(모델링)하여 컴퓨터에 옮기는 방식
- 개발 환경을 단순화함

2 임의로 데이터에 접근할 수 없음

- 메서드와 데이터가 객체로 묶여 있으므로 객체에서 제공하는 인터페이스를 통해서만 데이터에 접근할 수 있음

3 시스템은 객체들의 모음으로 쉽게 구성

4 요구 사항 변경에 유연하게 대처가능

- 객체를 추가·삭제가 용이 요구 사항 변경에 대처 가능함

5 확장성과 재사용성이 높아짐

- 필요한 객체 추가가 쉬운 구조로 확장성이 높고, 개발된 코드를 재사용함 하여 개발 시간과 비용을 줄일 수 있음

6 추상화를 통해 생산성과 품질이 높아짐

❖ 학습내용

[1] 객체지향의 개념

2. 객체지향 분석 및 설계

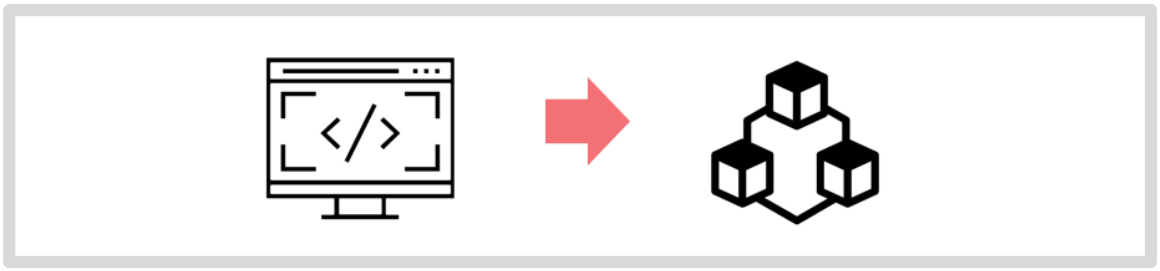
◈ 객체지향(OO: Object Oriented)

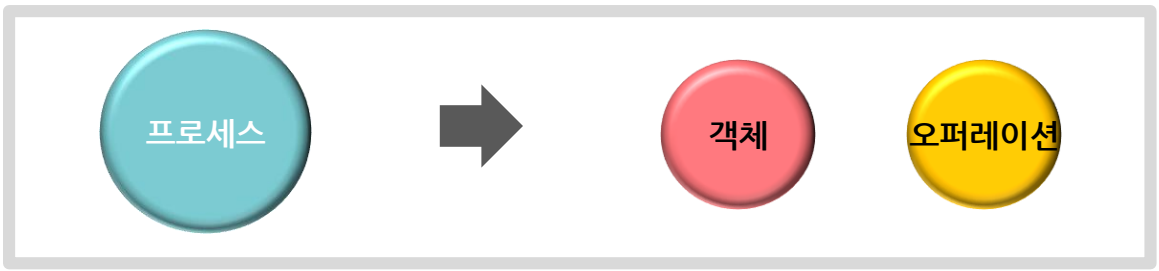
- 전체 시스템을 몇 개의 서비스 단위(Object)로 나누고 상태와 오페레이션의 상호작용으로 보는 기법

객체지향 접근 = 객체지향

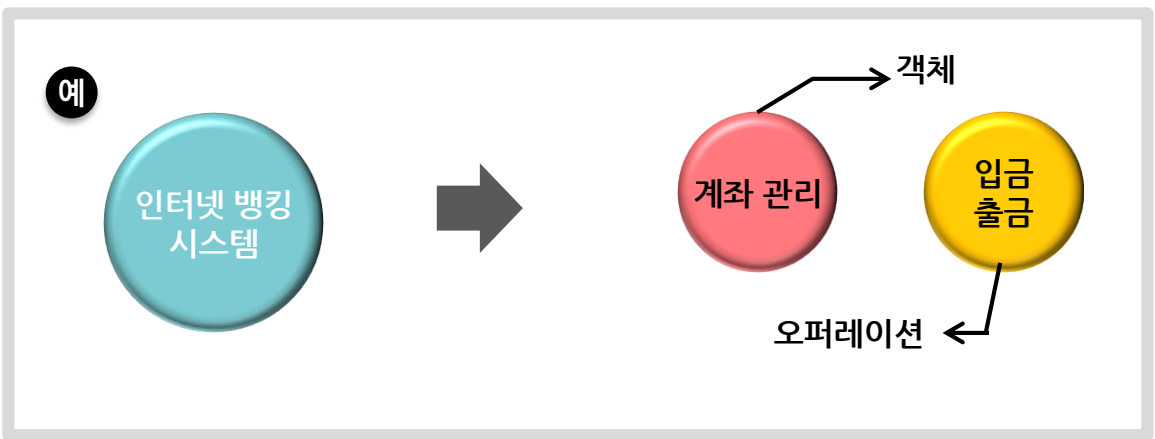
◈ 객체지향 분석(OOA: Object Oriented Analysis)

- 응용프로그램 영역을 객체모델로 개발하는 것을 의미
- 모델내의 객체(Object)는 해결하여야 하는 문제와 관련된 개체(Entity)와 오퍼레이션(Operation)을 나타냄





- 시스템으로 구현될 어떤 프로세스들이 있다면 이를 객체와 오퍼레이션으로 분석하는 절차를 의미함



❖ 학습내용

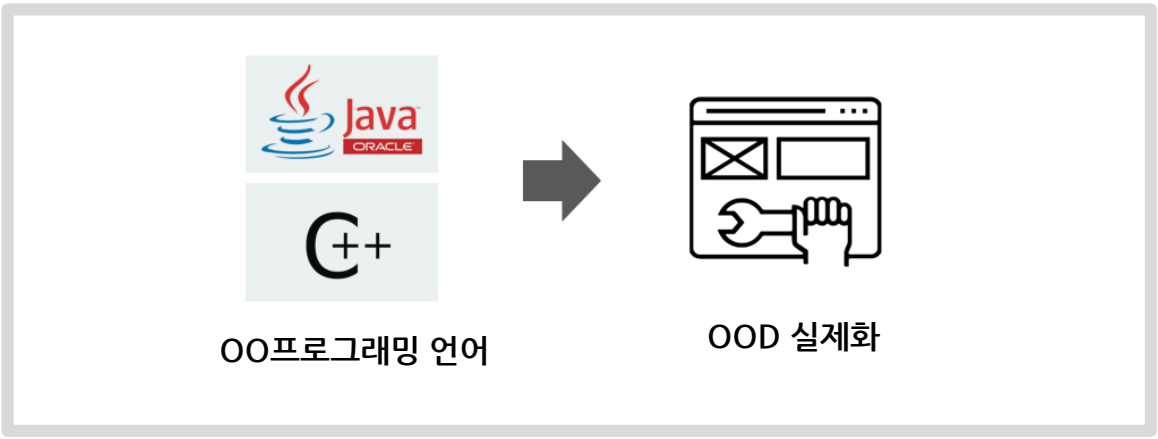
[1] 객체지향의 개념

2. 객체지향 분석 및 설계(계속)

◈ 객체지향(OO: Object Oriented)(계속)

- 도출된 요구사항의 구현을 위하여 객체지향 시스템 모델을 개발하는 것을 의미
- 객체지향 분석을 보다 구체화하여 프로그램 구현 전 단계의 설계작업의 진행을 의미

◈ 객체지향 프로그래밍(OOP: Object Oriented Programming)



- 객체(Object)를 프로그램으로 표현(객체모델화)한다면 단위 Object는 Class로 표현
- 클래스는 메소드(Method: 함수의 역할)와 인터페이스(Interface:변수-데이터의 역할)로 구성

3. 객체지향 프로그래밍

클래스(Class)와 인스턴스(Instance)

클래스	인스턴스
같은 종류 및 특성을 가진 객체들을 모아서 공통의 특성으로 분류하고 템플릿화 하는 것	클래스의 실체들로서 템플릿화 된 클래스로부터 파생된 하나의 실제 객체 (예: 붕어빵 틀과 붕어빵)

❖ 학습내용

[1] 객체지향의 개념

3. 객체지향 프로그래밍(계속)

구조체(Struct)

- 일반적인 프로그래밍 언어에서 많이 사용하는 구조체
- 구조체는 변수들을 묶어서 레코드 개념으로 프로그램에서 사용
- 단순히 프로그램 내에서 변수들의 묶음으로 사용될 뿐임

엘리베이터의 속성

```
struct  elevator
{
   int limit-up-floor;
   int limit-down-floor;
   int floor;
   char help[100];
}
```

❖ 학습내용

[1] 객체지향의 개념

3. 객체지향 프로그래밍(계속)

클래스(Class)

- 클래스의 기본은 변수(데이터)와 함수(기능, Method)를 가짐
- 클래스를 설계하는 사람과 전체 프로그램을 구현하는 사람이 다를 수 있음
- 클래스의 세부 내용은 복잡성을 가지기 때문에
  내용을 은닉하고 인터페이스만으로 클래스를 사용하도록 단순화 함

  → 사용자는 TV에서 내부의 처리 과정 내용(Object)은 관심 없고, 채널과 볼륨 스위치(인터페이스)만 밖에 보여지면 됨

- 인터페이스를 Up, Down 정도만 프로그램에서 제어할 수 있도록 한다면 프로그램은 간단히 구현될 것임
- 엘리베이터라는 실 세계(Real World)의 개념(Object)을 구현한 사례

```
Class  elevator
{
  int limit-up-floor=10; //최상위 층
  int limit-down-floor=0; //최하위층
  int floor; //현재층
  char help[100];
  void up()  //엘리베이터가 올라감
  {
    if (floor == limit-up-floor)
      help="last-floor";
    else
      floor++; //최상층이 아닐 때 한 층씩 올라감
  }
```

```
void down() //엘리베이터가 내려감
  {
    if (floor == limit-down-floor)
      help=" first-floor";
    else
      floor--; //최하층이 아닐 때 한 층씩 내려감
  }
}
```

Class Elevator를 이용,
쉽게 프로그램 구현

❖ 학습내용

[1] 객체지향의 개념

3. 객체지향 프로그래밍(계속)

클래스(Class)(계속)

```
 void main()
{
   Class *myClass elevator;

  if(올라가라는 전기 신호가 왔다면)
 {
      myClass->up();
 }
 else if(내려가라는 전기 신호가 왔다면)
 {
      myClass->down();
 }

}
```

일단 프로그램을
개념중심으로 먼저 구현 후 , Class의 내부를
구현하는 방법도 적용 가능함

❖ 학습내용

[2] 객체지향의 구성요소

1. 객체지향 구성 요소

◈ 구성 요소와 특징 비교

▪ 객체지향 설계에 공통으로 사용되는 구성 요소와 특징

객체지향 구성 요소	객체지향 특징
① 객체(Object) ② 클래스(Class) ③ 인스턴스(Instance)	① 캡슐화(Encapsulation) ② 정보은닉(Information Hiding) ③ 상속(Inheritance) ④ 다형성(Polymorphism)

1. 1. 객체(Object)

◈ 실 세계에 존재하거나 생각할 수 있는 것

◈ 책상, 의자, 전화기 같은 사물은 물론이고 강의, 수강 신청 같은 개념으로 존재하는 것

◈ 각 관점에 따라 다음과 같은 개념을 가짐

관점	개념
모델링 관점	객체는 명확한 의미를 담고 있는 대상 또는 개념임
프로그래머 관점	객체는 클래스에서 생성된 변수
소프트웨어 개발 관점	객체는 소프트웨어 개발 대상으로, 어떤 한 시점에 객체 상태를 나타내는 데이터와 해당 데이터를 처리하고 참조하는 동작을 의미하는 메서드(함수)를 모아놓은 '데이터+메서드' 형태의 소프트웨어 모듈
객체지향 프로그래밍 관점	객체는 데이터와 함수를 속성(Attribute)과 메서드(Method) 용어로 구현함

❖ 학습내용

[2] 소프트웨어의 개요

1. 1. 객체(Object)(계속)

소프트웨어 개발 관점에서 객체의 특성

식별자(Idendity) 존재	객체를 구별하는 유일한 식별자를 가짐
상태(State) 존재	자료구조에 해당하는 상태를 가짐
메서드(Method)존재	연산을 수행할 수 있는 행위에 해당하는 잘 정의된 메서드를 가짐
클래스로 선언 및 사용	객체지향 프로그램에서 객체는 비슷한 객체의 구조와 행위가 클래스로 선언되어 사용

1. 2. 클래스(Class)

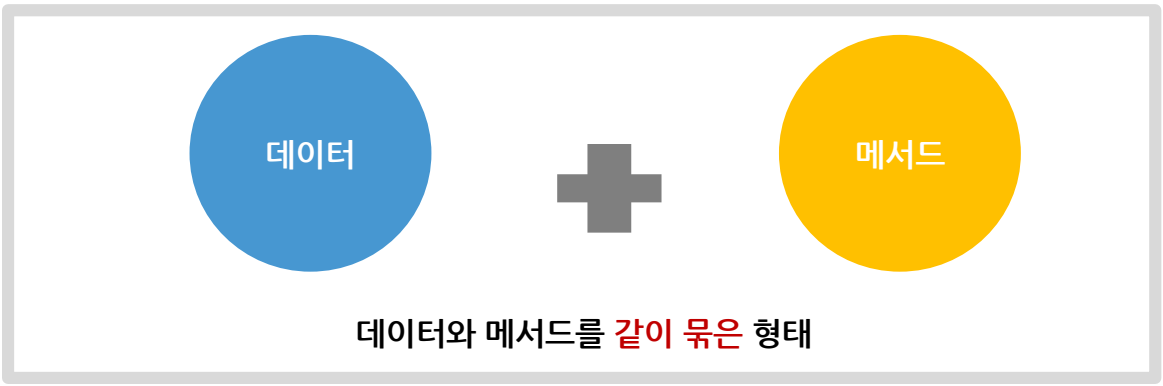
- 객체 = 붕어빵
- 클래스 = 붕어빵을 만드는 틀
- 클래스(Class): 공통되는 것들을 묶어서 대표적인 이름을 붙인 것
- 클래스는 개념적, 객체는 구체적

클래스	객체
승용차	소나타, 그랜져, SM5
자동차	승용차, 버스, 트럭
운송 수단	자동차, 배, 비행기

❖ 학습내용

[2] 소프트웨어의 개요

1. 2. 클래스(Class)(계속)



◈ 엘리베이터 클래스에서의 데이터와 메서드 구분

데이터	limit-up-floor limit-down-floor floor help[100]
메소드	up() down()

```
Class  elevator
{
  int limit-up-floor=10; //최상위층
  int limit-down-floor=0; //최하위층
  int floor; //현재 층
  char help[100];
  void up()  //엘리베이터가 올라감
  {
    if (floor == limit-up-floor)
      help=“ last-floor”;
    else
      floor++; //최상층이 아닐 때 한 층씩 올라감
  }
```

- 구조체: 데이터만 나열
- 클래스: 데이터와 메소드를 표현

❖ 학습내용

[2] 소프트웨어의 개요

1. 2. 클래스(Class)(계속)

◈ 엘리베이터 클래스에서의 데이터와 메서드 구분(계속)

데이터	limit-up-floor limit-down-floor floor help[100]
메소드	up() down()

```
void down() //엘리베이터가 내려감
  {
     if (floor == limit-down-floor)
       help="first-floor";
     else
        floor--; //최하층이 아닐 때 한 층씩 내려감
  }
}
```

- 구조체: 데이터만 나열
- 클래스: 데이터와 메소드를 표현

1. 3. 인스턴스(Instance)

같은 클래스에 속하는 개개의 객체로,
하나의 클래스에서 생성된 객체를 의미

클래스가 구체화되어, 클래스에서 정의된 속성과 성질을 가진
실제적인 객체로 표현된 것을 의미

❖ 학습내용

[2] 소프트웨어의 개요

1. 3. 인스턴스(Instance)(계속)

인스턴스화(Instantiation)

◈ 추상적인 개념인 클래스에서 실제 객체를 생성하는 것

인스턴스화(Instantiation)의 예시

데이터	limit-down-floor limit-down-floor floor help[100]
메소드	up() down()

엘리베이터 객체를 정의한 클래스

limit-up-floor=10 limit-down-floor=-2 floor help[100]
up() down()

최상층10층, 지하2층의
건물 속성을 가진
인스턴스로 인스턴스화

limit-up-floor=20 limit-down-floor=-5 floor help[100]
up() down()

최상층20층, 지하5층의
건물 속성을 가진
인스턴스로 인스턴스화

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징

- 캡슐화(Encapsulation)
- 실 사용자들에게 해당 객체의 기능(서비스)과 사용법만 제공하고

내부는 감추어 → 변경할 수 없게 함

쉽게 사용할 수 있게 하는 개념

- 소프트웨어 모듈인 객체 내부에 서로 관련된 데이터와 그 데이터를 조작할 수 있는 메서드를 같이 포장하는 방식
- 그 안에 포함된 메서드만 사용하여 데이터 값을 변경할 수 있는 구조
- 캡슐화는 클래스를 사용하여 정의하는데, 사용자는 클래스 내부를 알지 못하도록 처리
  - 클래스 내부를 알 필요가 없고, 안다면 프로그램 이해만 복잡해짐
- 캡슐화는 클래스를 사용하여 서로 관련된 정보와 처리 방식을 같이 묶고, 외부에는 감추어두는 것

예 우리는 TV를 간단히 음성 볼륨과 채널 선택 정도의 리모콘 기능만 알면 되지 TV의 LED의 구현동작원리를 알아야 TV시청이 가능한 것이 아님

캡슐화의 장점

- 객체지향 방식은 사용자가 모듈의 내부 구조를 알 필요도 없고, 변경할 필요도 없음 단지 어떤 기능을 하고, 어떻게 사용하는지만 알면 됨

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

캡슐화의 장점

1 데이터 보호

- 이용자가 데이터에 직접 접근하는 것을 차단하므로 객체 내 데이터 및 코드의 손상과 오용을 막고 안전하게 보호할 수 있음

2 추상화 용이

- 추상화를 통해 프로그래밍 문제를 쉽게 개념화할 수 있음

3 제공자와 이용자를 명확히 분리

- 객체 제공자와 객체 이용자(외부 객체)의 분담을 명확히 할 수 있음

4 이용자에게 편리성 제공

- 메서드의 구현 방법이 바뀌어도 이를 사용하는 데 어려움이 없음

예 버블 정렬이 퀵 정렬로 바뀜

5 사용법이 쉬움

- TV버튼의 기능만 알면 사용할 수 있는 것처럼, 메서드의 기능만 알면 객체를 사용할 수 있음

6 변화에 대한 국지적 영향

- 객체 내의 데이터 구조가 바뀌어도 다른 객체에는 영향을 주지 않음

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

캡슐화의 장점

7 객체 간의 독립성 보장

- 캡슐화(데이터+메서드)로 인해 객체 사이의 독립성이 구조적으로 보장됨

8 변경 용이성과 재사용성 증대

- 설계의 변경 용이성과 재사용성을 높여줌

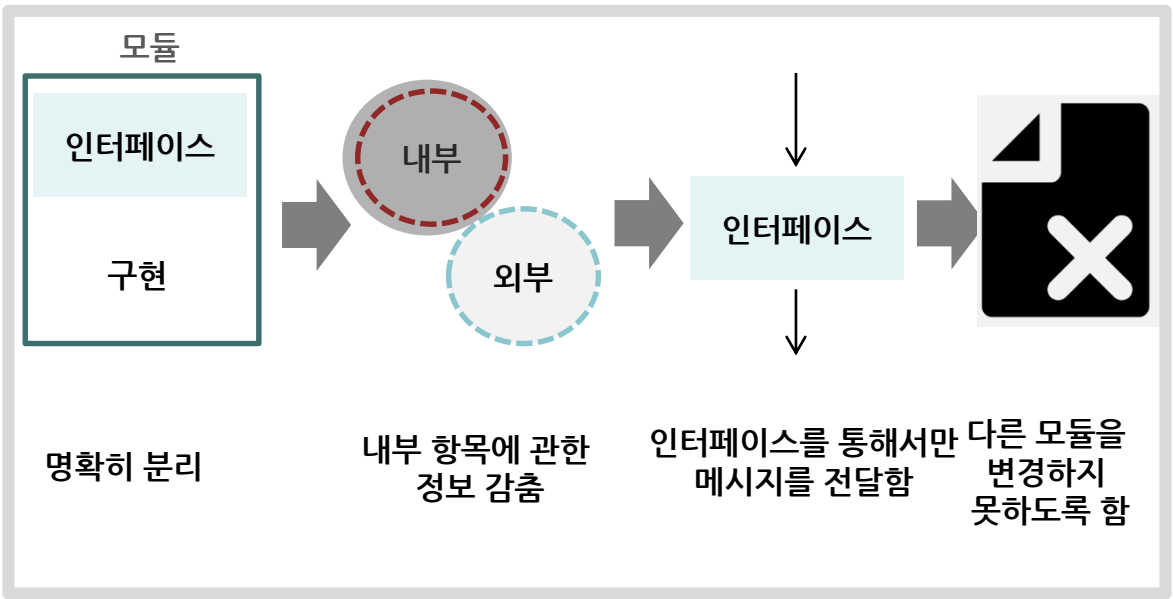
◈ 정보은닉(Information Hiding)

◈ 외부(다른 객체)에서 객체의 내부(데이터)를 들여다볼 수 없다는 개념

◈ 다른 객체가 한 객체 내의 데이터 값을 직접 참조하거나 접근할 수 없으므로 마치 친구끼리 대화를 하듯 메서드를 통해 객체에 요청해서 값을 넘겨받아야 함

◈ 캡슐화는 소프트웨어 모듈인 객체 내부의 관련된 데이터와 메서드를 함께 포장하는 방식으로, 캡슐 내부와 외부를 구별됨

- 그러나 캡슐화했다고 해서 그 자체로 내부의 정보가 외부에 숨겨지지는 않음



❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

◈ 정보은닉(Information Hiding)(계속)

◈ 클래스 내부의 데이터와 메소드 중 사용자가 접근을 차단할 필요가 있는 부분은 정보은닉으로 정의함

예

TV를 케이스로 싸고 버튼만 나오게 하여 사용자는 버튼만 눌러서 사용하게 하는 부분은 캡슐화

사용자가 만지면 고장의 원인이 되는 부품이나 기술적으로 중요한 부품은 분해할 수 없도록 해놓는 개념이 정보은닉

◈ C++,Java에서 변수나 함수에 private, protect, public의 개념이 정보은닉의 개념(프로그램 개발 경험이 있으면 쉽게 이해)

공개 (+, public)	▪ public으로 설정된 요소들은 같은 시스템에 있는 모든 클래스가 접근할 수 있음 ▪ 클래스는 클래스 이름, 속성, 메서드로 구성되는데, public으로 설정되는 부분은 대부분 메서드
은닉 (-, private)	▪ private으로 설정된 요소들은 같은 시스템 내의 다른 클래스는 직접 접근불가 ▪ 해당 클래스의 메서드를 통해서만 접근 가능 ▪ 클래스에서 대부분의 속성은 private으로 설정
부분 공개 (#, protected)	▪ protected로 설정된 요소들은 다른 클래스 접근 불가 ▪ 해당 클래스의 메서드와 이 클래스를 상속받은 하위 클래스만 접근 가능함

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

정보은닉의 특징

1 블랙박스 역할

- 정보은닉은 블랙박스처럼 데이터와 메서드를 숨기는 장치로, 객체의 사용자와 제공자의 역할을 명확히 분리

2 인터페이스를 통한 접근

- 사용자가 공개 인터페이스를 사용함으로써 객체 내부의 자료구조를 몰라도 객체를 쉽게 이용할 수 있음
- 사용자는 삽입 정렬, 버블 정렬 등의 모듈의 구현 형태를 몰라도 인터페이스를 통한 사용법만 알면 쉽게 이용할 수 있음

3 자료구조 변경이 용이

- 제공자가 객체 내부의 자료구조를 변경해도(버블 정렬을 퀵 정렬로 변경) 그 객체와 인터페이스를 통해 통신하는 사용자에게는 영향을 주지 않으므로 부담 없이 자료구조를 변경할 수 있음
- 사용자는 인터페이스만 바뀌지 않는다면 호출하는 모듈은 종전대로 사용할 수 있음

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

정보은닉의 장점

1 독립성 향상

- 다른 모듈과 관계가 적어 모듈의 독립성을 높여줌

2 수정 용이

- 기능을 중심으로 하는 프로세스 지향 방식에서는 다른 모듈에 미치는 영향을 충분히 검토한 후에 모듈을 수정해야 함
- 그러나 객체지향의 정보은닉 개념을 적용하면 인터페이스가 바뀌지 않는 한 다른 모듈에 미치는 영향을 고려할 필요가 없음

3 이해도 증진

- 프로그램 개발 시 다른 모듈의 구현 내용을 제공하는 기능만 알면 메시지를 통해 사용할 수 있어 프로그래머가 모듈을 이해하기가 쉬움

4 확장성 증가

- 모듈 내의 데이터와 알고리즘을 변경하기 쉬워 기능을 추가하기도 쉬움

◈ 상속(Inheritance)

◈ 객체지향에서도 상위 클래스(super Class)의 모든 것을 하위 클래스(sub Class)가 물려받아 내 것처럼 사용함을 의미

◈ 물려주는 클래스를 상위 클래스 또는 부모 클래스라고 하고, 물려받는 클래스를 하위 클래스 또는 자식 클래스라고 함

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

◈ 상속(Inheritance)(계속)

Class elevator	
데이터	limit-up-floor limit-down-floor floor help[100]
메소드	up() down()

엘리베이터 객체를 정의한 클래스

Class fastelev
기존 elevator 클래스를 모두 사용 + fastup(), fastdown() : 빠르게 오르고 내려가는 기능 추가

엘리베이터 객체를
상속 받고
다시 정의한
fastelev클래스

Class twoelev
기존 elevator 클래스를 모두사용 + up2(), down2() : 짝수 층만 오르고 내려가는 기능 추가

엘리베이터 객체를
상속 받고
다시 정의한
twoelev클래스

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

상속의 장점

1 이해 용이

- 개별 클래스를 상속 관계로 묶음으로써 클래스 간의 체계화된 전체 구조를 파악하기 쉬움

2 재사용성 증대

- 데이터와 메서드의 오버로딩을 피하고 기존 클래스에 있는 것을 재사용할 수 있음

3 확장 용이

- 새로운 클래스, 데이터, 메서드를 추가하기가 쉬움

4 유지보수 용이

- 데이터와 메서드를 변경할 때 상위에 있는 것만 수정하여 전체적으로 일관성을 유지

5 추상화 가능

- 일반화(Generalization), 특수화(Specialization)의 관계를 통해 추상화 단계를 표현할 수 있음

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

상속의 일반화와 특수화

Class elevator	
데이터	limit-up-floor limit-down-floor floor help[100]
메소드	up() down()

엘리베이터 객체를 정의한 클래스

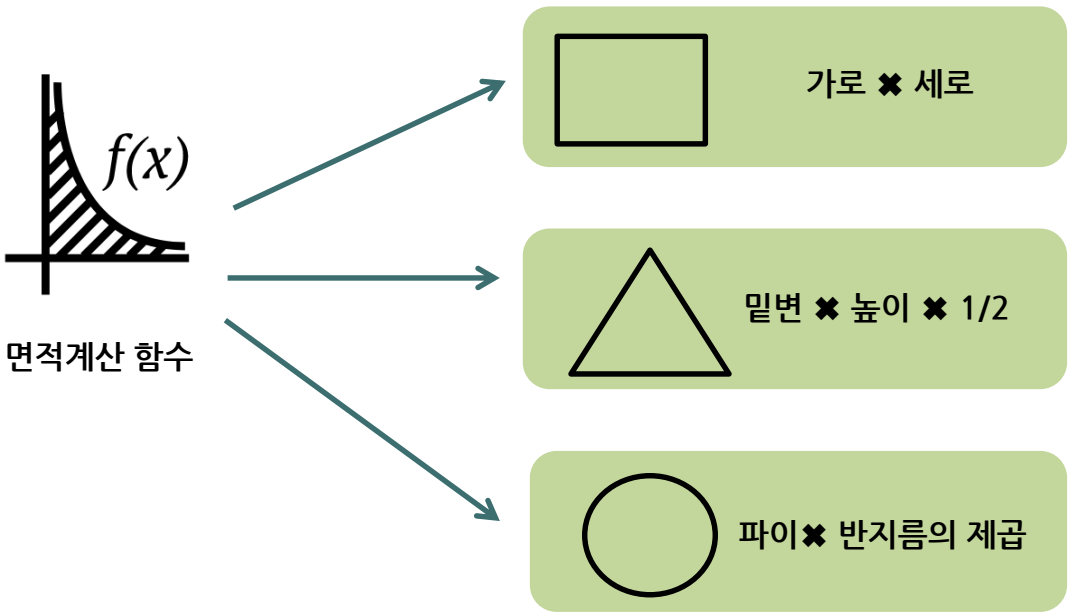
Class fastelev	Class twoelev
기존 elevator 클래스를 모두 사용 + fastup(), fastdown() : 빠르게 오르고 내려가는 기능 추가	기존 elevator 클래스를 모두사용 + up2(), down2() : 짝수 층만 오르고 내려가는 기능 추가
엘리베이터 객체를 상속 받고 다시 정의한 fastelev클래스	엘리베이터 객체를 상속 받고 다시 정의한 twoelev클래스

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

- 다형성(Polymorphism)
- '여러 개의 형태를 갖는다'라는 의미의 그리스어에서 유래
- 객체지향에서 다형성의 개념이 적용된 두 가지 형태인 오버로딩(Overloading)과 오버라이딩(Overriding)이 있음
  - 오버로딩은 메서드 중복 정의, 오버라이딩은 메서드 재정의를 뜻함



이러한 다형성을 객체지향에서는 표현이 가능함

- 메소드 오버로딩(Overloading)
  - 같은 함수명의 매개변수의 성격에 따라 함수가 정의됨
  - 숫자의 합: 2+3 = 5 , 문자열의 합 “2”+“3”= “23”
  - 그러나 메소드(함수) 파라메터 변수형태의 정의에 따라 같은 이름의 함수이나(add) 자동으로 합이 정의됨

❖ 학습내용

[2] 객체지향의 구성요소

2. 객체지향 특징(계속)

◈ 다형성(Polymorphism)(계속)

```
Class A{
    int add(int a,int b){
                return a+b;
    }
    String add(String a, String b){
                return a+b;
   }
}
```

```
Class a = new A();

a.add(1,2); //자동으로 수치연산 수행

a.add("1"."2"); //자동으로 문자연산 수행
```

◈ 메소드 오버라이딩(Overloading)

- 클래스를 상속받으면서 부모 클래스의 메소드(함수)를 새롭게 정의하는 방법

```
Class rectangle{
…
    int area(){
                return a*b;
    }
}
```

✓ 부모 클래스(사각형)에서 면적은 a*b로 되어있음

```
Class triangle()::rectangle {
    int area(){
                return a*b/2;
    }
}
```

✓ 삼각형클래스는 사각형(부모) 클래스를 상속받아 만듦

✓ 면적이라는 메소드를 재정의 함(오버라이딩)

❖ 학습내용

[3] 객체지향 기본원칙

1. 객체지향 기본 원칙

◈ 시스템 개발을 객체지향 방식으로 추진 시 지켜야 하는 기본원칙

1 시스템 컨텍스트 모델과 시스템 사용 모델의 파악 및 이해

- 설계 대상 소프트웨어와 외부환경 사이의 관계를 이해하는 것임

시스템 컨텍스트 모델	시스템 사용 모델
시스템과 관련 환경을 정적으로 표현한 것	시스템과 관련 환경을 사용관점에서 표현한 것
예 하드웨어 네트워크 구성도	예 시스템 처리 프로세스 도면, UML 등

2 시스템 구조 설계

- 시스템 컨텍스트 모델과 시스템 사용모델을 가지고 시스템 구조(Architecture)를 설계할 수 있음
- 아키텍처 설계는 일반적인 설계 단계와 유사함
- 인터페이스(프리젠테이션), 데이터, 비즈니스(컨트롤)영역으로 계층(Tier)을 나누어 설계하는 것도 한 방법임

3 주요 객체 식별

- 아키텍처 설계가 끝나면 그 설계 안을 중심으로 어떤 것이 반복되거나, 중요한 객체(Object)인지 찾아냄 ⇒ '식별'
- 객체로 정의될 것을 클래스로 표현함

❖ 학습내용

[3] 객체지향 기본원칙

1. 객체지향 기본 원칙(계속)

4 설계 모델 개발

- 객체와 객체 클래스 그리고 객체 사이의 관계를 설계함
- 이 과정에서의 설계물은 프로그래머가 구현 가능한 수준이어야 함

5 객체 인터페이스 명세화

- 최종적으로 정해진 객체는 캡슐화(은닉)하고 기능 등을 교류할 수 있는인터페이스로 정의하고 명세화 하여야 함

2. OOP와 CBD(Component Based Development)

◈ 컴포넌트(Component)와 CBD

◈ 소프트웨어 공학측면에서 프로그램을 쉽게 구현하고 재사용 가능하게 하기 위하여 프로그램을 모듈단위로 구현해 둔 방식

컴포넌트 기반 개발 방법(CBD)

컴포넌트를 이용하여 마치 조립식 건물이나 레고 장난감 만들듯이 소프트웨어를 개발하는 방식

- 컴포넌트를 구현 및 등록하는 과정과
  컴포넌트를 이용하여 개발을 쉽게 할 수 있는 과정이 있음
- 재사용 가능한 부분들은 컴포넌트 기반으로 구축 활용한다면
  개발 노력과 비용을 줄일 수 있음
- 전체 시스템을 각각의 객체(Object)로 나누어 프로그램으로 접근하는 OOP는 CBD를 쉽게 할 수 있는 방법 중 하나임

  - CBD를 구현하기 위하여 꼭 OOP로만 구현하여야 하는 것은 아님

❖ 핵심정리

1. 객체지향의 개념

- 실 세계를 사람이 생각하는 방식으로 표현하는 방식인 객체지향 접근방법으로 소프트웨어를 구축하면 많은 장점이 있음
- 객체지향 개발방법의 구체적인 프로그래밍표현은 클래스임

2. 객체지향의 구성요소

- 객체지향 구성요소 : 객체 (Object),클래스 (Class),인스턴스 (Instance)
- 객체지향 특징 : 캡슐화 (Encapsulation),정보은닉(Information Hiding), 상속(Inheritance), 다형성(Polymorphism)

3. 객체지향의 기본원칙

- 시스템 개발을 객체지향 방식으로 추진 시 기본원칙을 충실히 지켜서 개발하여야 함
- 전체 시스템을 각각의 객체로 나누어 프로그램으로 접근하는 OOP는 CBD를 쉽게 할 수 있는 방법 중 하나임