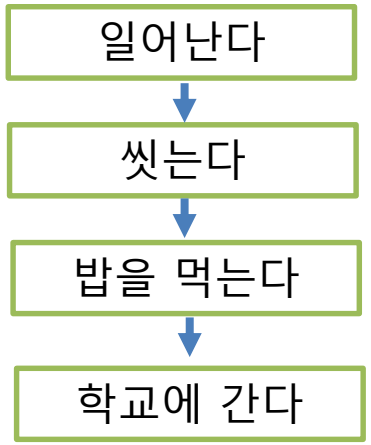
7장. 클래스와 상속

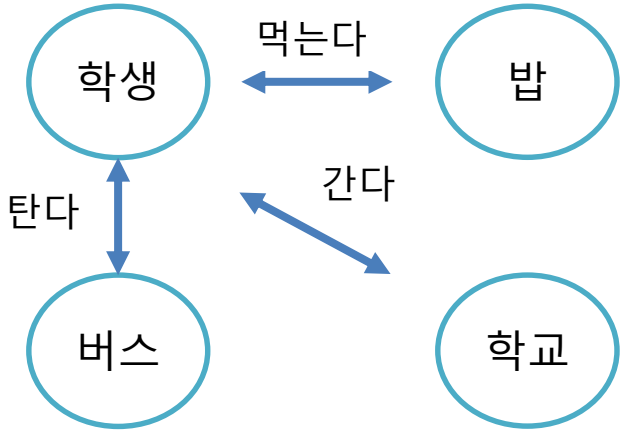
목 차

객체 지향 프로그래밍

- **객체(Object)란?**
  - "의사나 행위가 미치는 대상" -> 사전적 의미
  - 구체적, 추상적 데이터 단위 (구체적- 책상, 추상적-회사)
- **객체지향 프로그래밍(Objected Oriented Programming, OOP)**
  - 객체를 기반으로 하는 프로그래밍
  - 먼저 객체를 만들고 객체 사이에 일어나는 일을 구현함.



<절차지향 –C언어>



<객체지향 –Java>

클래스(class)

- **클래스란?**

  객체에 대한 속성과 기능을 코드로 구현 한 것

  객체에 대한 설계도 또는 청사진.

  "클래스를 정의 한다"라고 함,

- **객체의 속성과 기능**
  - 객체의 특성(property), 속성(attribute) -> **멤버 변수**
  - 객체가 하는 기능 -> **멤버 함수**

- **클래스 정의하기**

```
class 클래스 이름 :
    def __init__(self):
        멤버변수
    def 함수이름(self):
        return
```

학생 클래스

- 속성(멤버변수) : 학번, 이름, 학년, 사는 곳 등..
- 기능(함수) : 수강신청, 수업듣기, 시험 보기 등..

클래스(class) 정의

- 학생 클래스 정의

  멤버변수와 멤버 함수에 self 키워드 사용

  객체(인스턴스) = 클래스 이름()

Student
name grade
learn()

student_cls.py

```
class Student:
    def __init__(self):
        self.name = "콩쥐"    # 멤버 변수
        self.grade = 1
        print("생성자 함수")

    def learn(self):          # 멤버 함수
        print("수업을 듣습니다.")
```

```
s = Student()    # s는 Student 클래스의 객체(인스턴스)
print(s.name + " 학생은 " + str(s.grade) + "학년입니다.")
s.learn()
```

__init__() 초기자

- **초기자, 생성자(constructor)**

  클래스를 생성할 때 호출되는 명령어 집합, 초기자라고도 한다.
  생성자(초기화함수)는 **__init__()**의 형태로 작성하고, 리턴값이 없다.

```
class Student:
    def __init__(self, name, grade): # 생성자 함수(매개 변수)
        self.name = name       # 멤버 변수에 매개변수 저장
        self.grade = grade

    def learn(self):
        print("수업을 듣습니다.")
```

student_cons.py

```
s1 = Student2("김하나", 1)  # 이름과 학년을 인자로 전달하여 객체 생성
print(s1.name + " 학생은 " + str(s1.grade) + "학년입니다.")
s1.learn()

s2 = Student2("이두울", 2)
print(s2.name + " 학생은 " + str(s2.grade) + "학년입니다.")
s2.learn()
```

__init__() 초기자

- **__str(self)__**

  문자열을 return하는 함수이다. 객체의 정보를 담고 있다.

student.py

```
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

    def learn(self):
        print("수업을 듣습니다.")

    def __str__(self):  # 멤버를 문자열로 리턴하는 함수
        return "이름 : {}, 학년 : {}".format(self.name, self.grade)
```

```
s1 = Student("김하나", 1)
print(s1)
s1.learn()

s2 = Student("이두울", 2)
print(s2)
s2.learn()
```

계산기 클래스 만들기

- 계산기 클래스 만들기

Calculator
x, y
add() sub()

calculator.py

```
# 계산기 클래스 정의 및 사용
class Calculator:
    def __init__(self):
        self.x = 0

    def add(self, y):
        self.x += y
        return self.x

    def sub(self, y):
        self.x -= y
        return self.x
```

```
c = Calculator()
print(c.add(10))  # 10을 더하기
print(c.sub(4))   # 4를 빼기
```

클래스(class) 모듈 사용

- 클래스를 모듈로 import 하는 방법 – 외부 파일에서 사용

외부에서 사용할 때
실행되지 않음

```
class Student:
    def __init__(self, name, grade):
        self.name = name
        self.grade = grade

    def learn(self):
        print("수업을 듣습니다.")

    def __str__(self):  # 멤버를 문자열로 리턴하는 함수
        return "이름 : {}, 학년 : {}".format(self.name, self.grade)
```

```
if __name__ == "__main__":
    s1 = Student("김하나", 1)
    print(s1)
    s1.learn()

    s2 = Student("이두울", 2)
    print(s2)
    s2.learn()
```

클래스(class) 모듈 ^사용

- 클래스를 모듈로 import 하는 방법 – 외부 파일에서 사용

```
from class_lib.student import Student

s1 = Student("흥부", 1)
print(s1)
s1.learn()

s2 = Student("놀부", 2)
print(s2)
s2.learn()
```

student_test.py

객체 리스트

- 학생 리스트 만들기

```
******* 학생 명단 *******
이름 : 김하나, 학년 : 1
이름 : 이두울, 학년 : 2
이름 : 박세엣, 학년 : 3
```

student_list.py

```python
from class_lib.student import Student

# 객체 리스트 생성
s = [
    Student("김하나", 1),
    Student("이두울", 2),
    Student("박세엣", 3),
]

#print(s[0])
print("******* 학생 명단 *******")
for i in s:
    print(i)
```

정보 은닉(Information Hiding)

◆ 정보 은닉 – 클래스에 접근을 제한하여 보안을 좋게한 것

- 멤버 변수에 언더스코어(__) 1개(2개)를 붙이면 직접 접근할 수 없음(private)
- get(), set()으로 만들면 가능함(public)
- **get + 변수 이름(), set + 변수이름()**

```
class Person:
    def __init__(self, name, age):
        self._name = name
        self._age = age


p1 = Person()
p1.
```

```
__init__(self, name, age)
while
__str__(self)
```

person.py

정보 은닉(Information Hiding)

```
class Person:
    def __init__(self):
        self._name = ""  #멤버 변수 초기화
        self._age = 0

    def setname(self, name): # _name에 접근할 setname()함수
        self._name = name

    def getname(self):
        return self._name

    def setage(self, age): # _age에 접근할 setage()함수
        self._age = age

    def getage(self):
        return self._age

    def __str__(self):
        return "이름 : {}, 나이 : {}".format(self.getname(), self.getage())
```

```
p1 = Person()
p1.setname("콩쥐")
p1.setage(17)
print(p1)

p2 = Person()
p2.setname("팥쥐")
p2.setage(18)
print(p2)
```

정보 은닉(Information Hiding)

- 건강 상태 클래스 만들기

  술을 마시면 체력이 1감소, 운동을 하면 체력이 1증가하는 클래스

```
1시간 운동하다.
5잔 마시다.
이몽룡 hp : 86
3시간 운동하다.
1잔 마시다.
성춘향 hp : 99
```

```python
class health:
    def __init__(self, name):
        self.__name = name
        self.__hp = 100

    def getname(self):
        return self.__name

    def sethp(self, hp):
        if hp < 0:
            hp = 0
        if hp > 100:
            hp = 100
        self.__hp = hp

    def gethp(self):
        return 'hp : ' + str(self.__hp)
```

정보 은닉(Information Hiding)

- health 클래스 사용.

```
def exercise(self, hours):
    self.sethp(self.__hp + hours)
    print("{}시간 운동하다.".format(hours))

def drink(self, cups):
    self.sethp(self.__hp - cups)
    print("{}잔 마시다.".format(cups))
```

```
p1 = health("이몽룡")
p1.sethp(90)
p1.exercise(1)
p1.drink(5)
print(p1.getname(), p1.gethp())

p2 = health("성춘향")
p2.sethp(100)
p2.exercise(3)
p2.drink(1)
print(p2.getname(), p2.gethp())
```

인스턴스 변수와 클래스 변수

◆ 클래스 변수

해당 클래스를 사용하는 모두에게 공용으로 사용되는 변수.
클래스 이름으로 직접 접근한다.

인스턴스 변수

```
class Counter:
    def __init__(self):
        self.x = 0
        self.x += 1

    def get_count(self):
        return self.x

c1 = Counter()
print(c1.get_count())    # 1
c2 = Counter()
print(c2.get_count())    # 1
c3 = Counter()
print(c3.get_count())    # 1
```

클래스 변수

```
class Counter:
    x = 0    # 클래스 변수

    def __init__(self):
        Counter.x += 1
        # 클래스이름으로 직접 접근

    def get_count(self):
        return self.x

c1 = Counter()
print(c1.get_count())    # 1
c2 = Counter()
print(c2.get_count())    # 2
c3 = Counter()
print(c3.get_count())    # 3
```

인스턴스 변수와 클래스 변수

◆ 클래스 리스트

```
['계란']
['계란', '두부']
['계란', '두부', '커피']
```

```
class Cart:
    li = []

    def __init__(self, goods):
        Cart.li.append(goods)

    def __str__(self):
        return Cart.li
```

```
cart1 = Cart("계란")
print(cart1.li)
cart2 = Cart("두부")
print(cart2.li)
cart3 = Cart("커피")
print(cart3.li)
```

사번 자동 부여

- 사번 자동 발급

```
사번 : 1001, 이름 : 김하늘
사번 : 1002, 이름 : 안산
사번 : 1003, 이름 : 최바다
```

```python
class Employee:
    serial_num = 1000  #기준값 (클래스 변수)

    def __init__(self):
        Employee.serial_num += 1      # serial_num을 1증가
        self.id = Employee.serial_num # id에 serial_num 저장

    def get_id(self):
        return self.id

    def set_name(self, name):   # name을 매개변수로 전달
        self.name = name

    def get_name(self):
        return self.name
```

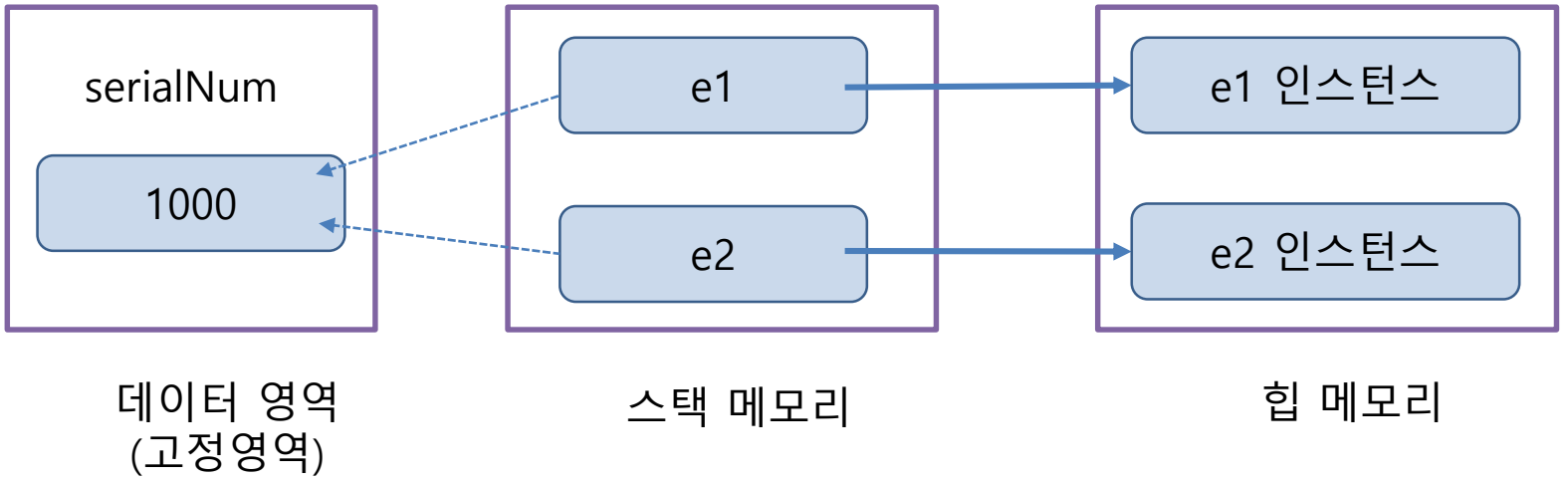
사번 자동 부여

- 사번 자동 발급

```
from class_lib.employee import Employee

e1 = Employee()
e1.set_name("김하늘")
print("사번 : " + str(e1.get_id()) + ", 이름 : " + e1.get_name())

e2 = Employee()
e2.set_name("안산")
print("사번 : " + str(e2.get_id()) + ", 이름 : " + e2.get_name())

e3 = Employee()
e3.set_name("최바다")
print("사번 : " + str(e3.get_id()) + ", 이름 : " + e3.get_name())
```

학번 자동 부여

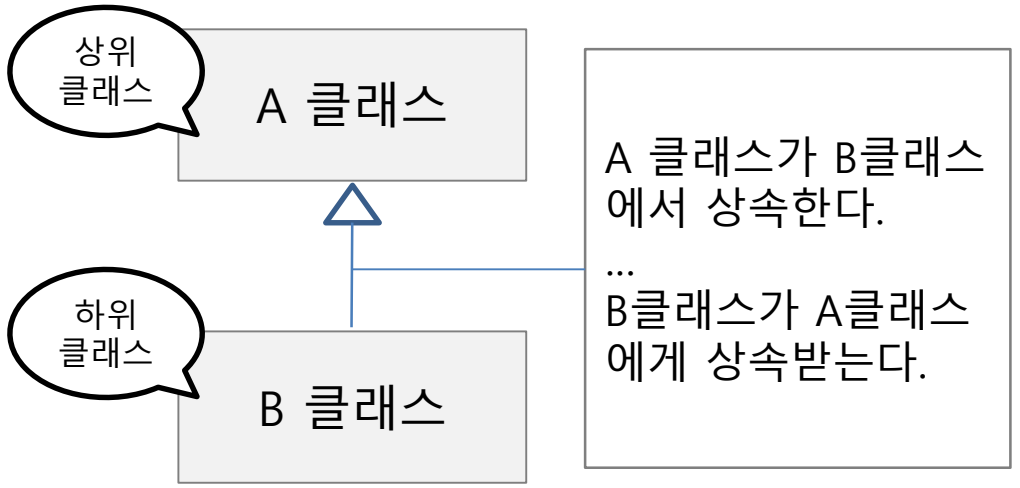
- 클래스의 메모리 영역



상속(Inheritance)

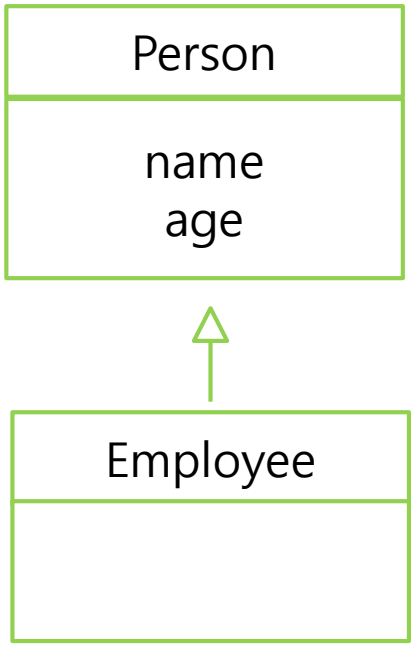
- **상속이란?**
  - 클래스를 정의할때 이미 구현된 클래스를 상속(inheritance) 받아서 속성이나 기능이 확장되는 클래스를 구현함.
  - 클래스 상속 문법

class 클래스 이름(상속할 클래스 이름)



상속(inheritance)

- 클래스 상속

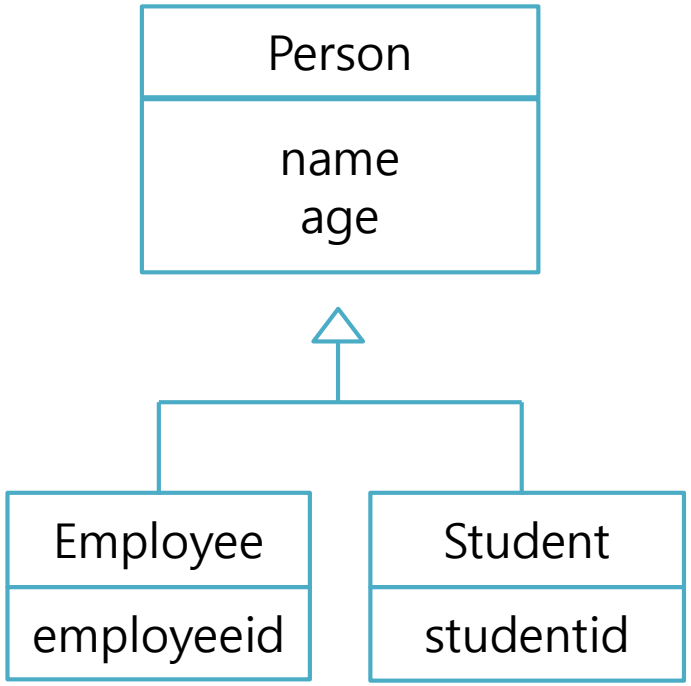


class 클래스 이름(상속할 클래스 이름)

```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Employee(Person):  #Person 상속 받음
    pass

p1 = Person("한강", 25)
print(p1.name, p1.age)

e1 = Employee("남한강", 30)
print(e1.name, e1.age,)
```

상속(inheritance)

- 매개 변수가 있는 상속의 경우 – **super()** 사용




```
class Person:
    def __init__(self, name, age):
        self.name = name
        self.age = age

class Employee(Person):  #Person 상속 받음
    def __init__(self, name, age, employeeid):
        super().__init__(name, age)
        self.employeeid = employeeid

class Student(Person):
    def __init__(self, name, age, studentid):
        super().__init__(name, age)
        self.studentid = studentid
```

상속(inheritance)

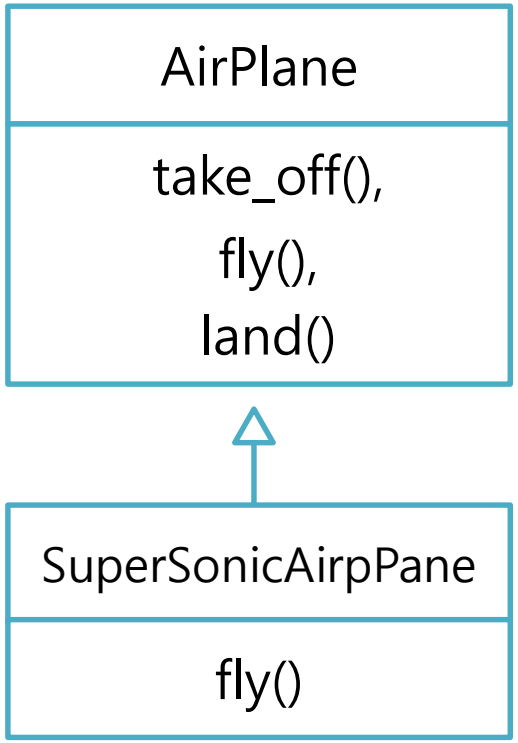
- 매개 변수가 있는 상속의 경우 – **super()** 사용

```
p = Person("한강", 25)
print(p.name, p.age)

e = Employee("남한강", 30, 301)
print(e.name, e.age, e.employeeid)

s = Student("북한산", 20, 101)
print(s.name, s.age, s.studentid)
```

함수 재정의(오버라이딩)

- **메소드 재정의(Method Overriding) 및 상속 – super() 사용**

  상속된 메소드의 내용이 자식 클래스에 맞지 않을 경우, 자식 클래스에서 동일한 메소드를 재정의 하는 것을 말한다.




```
비행기가 이륙합니다.
비행기가 일반 비행합니다.
초음속으로 비행합니다.
비행기가 일반 비행합니다.
비행기가 착륙합니다.
```

함수 재정의(오버라이딩)

- **비행 모드 전환하기**

__init__(self) –기본 생성자는 생략 가능.

```
class AirPlane:

    def take_off(self):
        print("비행기가 이륙합니다.")

    def fly(self):
        print("비행기가 일반 비행합니다.")

    def land(self):
        print("비행기가 착륙합니다.")
```

함수 재정의(오버라이딩)

- **비행 모드 전환하기**

클래스 상수는 대문자로 표기.. 클래스 이름으로 직접 접근

```
class SuperSonicAirPlane(AirPlane):
    NORMAL = 1          # 클래스 상수 - 대문자
    SUPERSONIC = 2

    def __init__(self):
        self.fly_mode = SuperSonicAirPlane.NORMAL

    def fly(self):
        if self.fly_mode == SuperSonicAirPlane.SUPERSONIC:
            print("초음속으로 비행합니다.")
        else:
            super().fly()
```

부모 클래스 메서드 상속 – super()

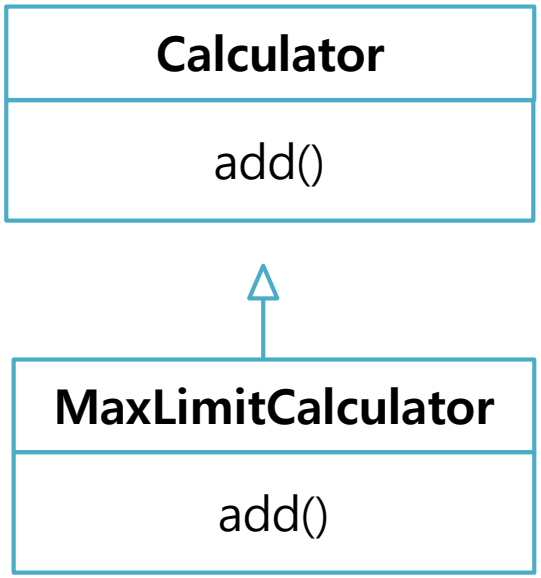
함수 재정의(오버라이딩)

- 비행 모드 전환하기

```
sa = SuperSonicAirPlane()
sa.take_off()
sa.fly()
sa.fly_mode = SuperSonicAirPlane.SUPERSONIC
sa.fly()
sa.fly_mode = SuperSonicAirPlane.NORMAL
sa.fly()
sa.land()
```

함수 재정의(오버라이딩)

- 계산기 클래스의 기능 확장 및 메서드 재정의




```
class Calculator:
    def __init__(self):
        self.value = 0

    def add(self, val):
        self.value += val
        return self.value
```

함수 재정의(오버라이딩)

- 객체 변수 value가 100 이상의 값을 가질 수 없도록 제한함

```
class MaxLimitCalculator(Calculator):
    def add(self, val):
        self.value += val
        if self.value >= 100: #재정의
            self.value = 100
        return self.value

cal = MaxLimitCalculator()
cal.add(50)
cal.add(60)

print(cal.value)
```

상속(inheritance)

- 단위변환- inch(인치)를 mm로 변환하는 클래스

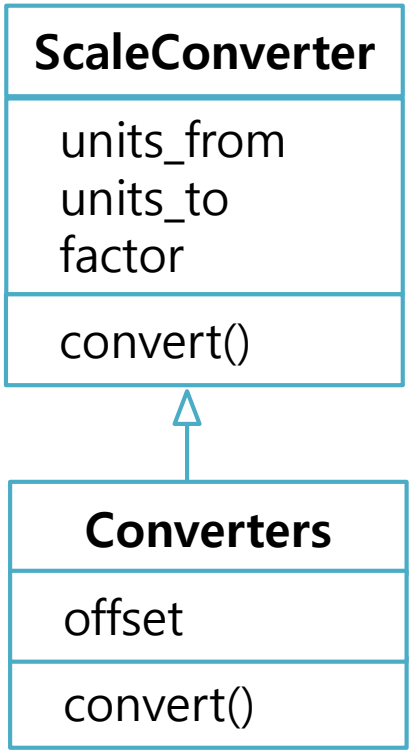
ScaleConverter
units_from units_to factor
convert()

```
# inch를 mm로 변환하기 : 1inch -> 25mm
class ScaleConverter:
    def __init__(self, units_from, units_to, factor):
        self.units_from = units_from    # 단위
        self.units_to = units_to        # 변환할 단위
        self.factor = factor            # 변환 값

    def convert(self, value):  # 변환 계산 함수
        return self.factor * value  # 단위 기본값 x 수
```

```
if __name__ == "__main__":
    c = ScaleConverter("inches", "mm", 25)
    print("Converting 2 inches")
    print(str(c.convert(2)) + c.units_to)
```

상속 실습예제

- 단위 변환기 확장 클래스 만들기
  - 화씨온도(F) = 섭씨온도(C) x 1.8 + 32



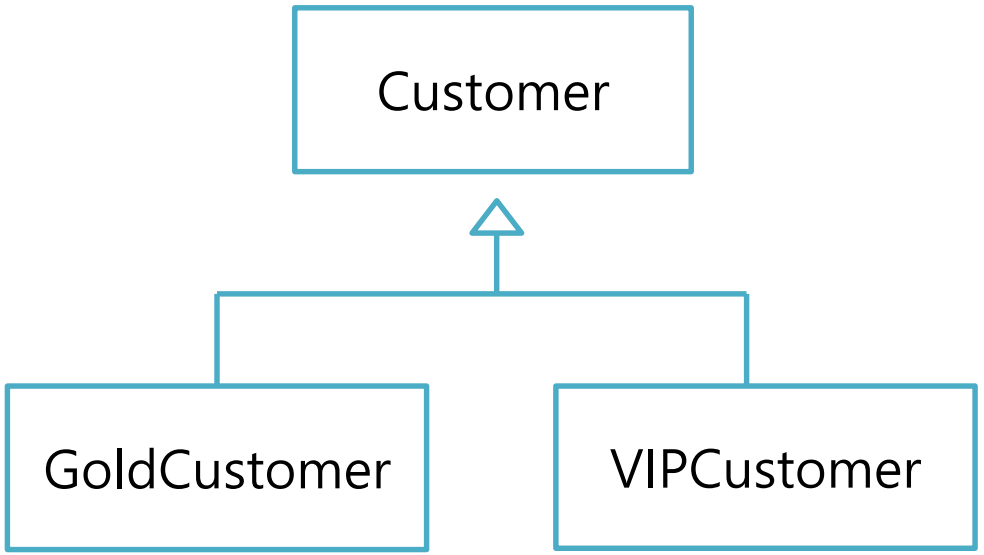
상속 실습예제

- 단위 변환기 확장 클래스 만들기
  - 화씨온도(F) = 섭씨온도(C) x 1.8 + 32

```
from ch07.class_lib.scaleconverter import ScaleConverter
# 단위 변환 클래스 - 온도 변환
# F(화씨온도) = C(섭씨온도) * 1.8 + 32
class Converter(ScaleConverter):
    def __init__(self, units_from, units_to, factor, offset):
        super().__init__(units_from, units_to, factor)  #부모 클래스 멤버 상속
        self.offset = offset   # 단위 값

    def convert(self, value):
        return self.factor * value + self.offset  # (단위1 값 x 수) + 단위2 값

con = Converter('C', 'F', 1.8, 32)
print("Converting 20C")
print(str(con.convert(20)) + con.units_to)
```

고객 관리 프로그램

- 고객 관리 프로그램



고객 관리 프로그램

Customer 클래스

예제 시나리오

고객 등급은 SILVER 이고, 다음과 같은 혜택을 제공합니다.

- **보너스 포인트를 1% 적립해 줍니다.**

멤버 변수	설 명
cid	고객 아이디
name	고객 이름
cgrade	고객 등급
bonus_point	고객의 보너스 포인트(마일리지)
bonus_ratio	고객의 포인트 적립 비율

고객 관리 프로그램

customer.py

```
class Customer:
    def __init__(self, cid, name):
        self.cid = cid              # 고객 아이디
        self.name = name            # 고객 이름
        self.cgrade = "SILVER"      # 고객 등급
        self.bonus_point = 0        # 보너스 포인트
        self.bonus_ratio = 0.01     # 보너스 적립율 1%

    def calc_price(self, price):  # 가격을 입력받아 계산하기
        self.bonus_point += int(price * self.bonus_ratio)
        #보너스 포인트 = 보너스 포인트 + (가격 x 보너스 적립율)
        return price

    def __str__(self):  # 고객 정보 문자열 출력
        return self.name + "님의 등급은 " + self.cgrade + \
               " 이고, 보너스 포인트는 " + str(self.bonus_point) + "점 입니다."
```

고객 관리 프로그램

GoldCustomer 클래스

예제 시나리오

고객이 늘어 VIP 고객만큼 물건을 많이 구매하지는 않지만, 그래도 단골인 고객들에게 혜택을 주고 싶습니다.

GOLD 고객 등급을 하나 추가하고 혜택을 줍니다.

- **제품을 살 때는 항상 10%를 할인해 줍니다.**
- **보너스 포인트를 2% 적립해 줍니다.**

고객 관리 프로그램

goldcustomer.py

```python
from ch07.class_lib.customer import Customer

class GoldCustomer(Customer):    # Customer를 상속
    def __init__(self, cid, name):
        super().__init__(cid, name)  # 부모 멤버 상속
        self.cgrade = "GOLD"         # 골드 등급
        self.sale_ratio = 0.1        # 구매 할인율
        self.bonus_ratio = 0.02      # 보너스 적립율

    def calc_price(self, price):  # 부모 메서드(함수) 재정의
        price -= int(price * self.sale_ratio)
        # 할인된 가격 = 가격 - (가격 x 구매할인율)
        self.bonus_point += int(price * self.bonus_ratio)
        # 보너스 포인트 = 할인된 가격 x 보너스 적립율
        return price
```

고객 관리 프로그램

VIPCustomer 클래스

예제 시나리오

고객이 점점 늘어나고 판매도 많아지다 보니 단골 고객이 생겼습니다. 단골 고객은 회사 매출에 많은 기여를 하는 우수 고객입니다. 우수 고객 등급은 VIP이고, 다음과 같은 혜택을 제공합니다.

- **제품을 살 때는 항상 10%를 할인해 줍니다.**
- **보너스 포인트를 5% 적립해 줍니다.**
- **담당 전문 상담원을 배정합니다.**

고객 관리 프로그램

vipcustomer.py

```
from ch07.class_lib.customer import Customer

class VIPCustomer(Customer):
    def __init__(self, cid, name, agent):
        super().__init__(cid, name)
        self.agent = agent      # 멤버 변수 상담원 초기화
        self.cgrade = "VIP"     # VIP 등급
        self.sale_ratio = 0.1
        self.bonus_ratio = 0.05

    def calc_price(self, price):  #부모 메서드 재정의(GOLD와 동일)
        price -= int(price * self.sale_ratio)
        self.bonus_point += int(price * self.bonus_ratio)
        return price

    def __str__(self):  # 부모 메서드 재정의
        return super().__str__() + \
               "\n상담원 ID는 " + str(self.agent) + "입니다."
```

고객 관리 프로그램

customer_main.py

```python
from ch07.class_lib.goldcustomer import GoldCustomer
from ch07.class_lib.customer import Customer
from ch07.class_lib.vipcustomer import VIPCustomer

s = Customer(101, "놀부")            #Customer 객체 생성
g = GoldCustomer(201, "흥부")        #GlodCustomer 객체 생성
v = VIPCustomer(301,"심청이", 1234) #VIPCustomer 객체 생성

# 상품 가격 계산
s.calc_price(10000)
g.calc_price(10000)
v.calc_price(10000)

# 고객 정보
print(s)
print(g)
print(v)
```

```
놀부님의 등급은 SILVER 이고, 보너스 포인트는 100점 입니다.
흥부님의 등급은 GOLD 이고, 보너스 포인트는 180점 입니다.
심청이님의 등급은 VIP 이고, 보너스 포인트는 450점 입니다.
상담원 ID는 1234입니다.
```

고객 관리 프로그램

리스트로 고객 관리

```
***** 구매 가격과 보너스 포인트 계산 *****
놀부님의 지불 금액은 10,000원 입니다.
팥쥐님의 지불 금액은 10,000원 입니다.
흥부님의 지불 금액은 9,000원 입니다.
콩쥐님의 지불 금액은 9,000원 입니다.
심청이님의 지불 금액은 9,000원 입니다.
***** 고객 정보 출력 *****
놀부님의 등급은 SILVER 이고, 보너스 포인트는 100점 입니다.
팥쥐님의 등급은 SILVER 이고, 보너스 포인트는 100점 입니다.
흥부님의 등급은 GOLD 이고, 보너스 포인트는 180점 입니다.
콩쥐님의 등급은 GOLD 이고, 보너스 포인트는 180점 입니다.
심청이님의 등급은 VIP 이고, 보너스 포인트는 450점 입니다.
상담원 ID는 1234입니다.
```

고객 관리 프로그램

customer_main2.py

```
customer = [
    Customer(101, "놀부"),
    Customer(102, "팥쥐"),
    GoldCustomer(201, "흥부"),
    GoldCustomer(202, "콩쥐"),
    VIPCustomer(301,"심청이", 1234)
]

print("***** 구매 가격과 보너스 포인트 계산 *****")
for c in customer:
    cost = c.calc_price(10000)
    print(c.getname() + "님의 지불 금액은 " + format(cost, ',d') + "원 입니다.")

print("***** 고객 정보 출력 *****")
for c in customer:
    print(c)
```