



AI Programming

Lecture 17

Review

- OOP via class

단계	작업	형식	예
1단계	클래스 선언	<pre>class 클래스명: # 필드 선언 # 메서드 선언</pre>	<pre>class Car : color = " def upSpeed(self, value) : ...</pre>
			
2단계	인스턴스 생성	<pre>인스턴스 = 클래스명()</pre>	<pre>myCar1 = Car()</pre>
			
3단계	필드나 메서드 사용	<pre>인스턴스.필드명 = 값 인스턴스.메서드()</pre>	<pre>myCar1.color = "빨강" myCar1.upSpeed(30)</pre>

Preview

- Ch. 12 객체지향 프로그래밍
 - 12.4 인스턴스 변수와 클래스 변수
 - 12.5 클래스의 상속
 - 12.6 객체지향 프로그래밍의 심화내용

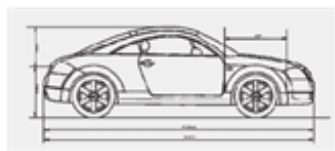
12.4 인스턴스 변수와 클래스 변수

Instance Variables

- Instance variables

- `myCar.speed`

자동차 설계도(클래스)



색상
속도



자동차(인스턴스)



색상
속도



색상
속도

```
class Car :  
    color = ""  
    speed = 0
```



```
myCar1 = Car()
```

myCar1.color

myCar1.speed

```
myCar2 = Car()
```

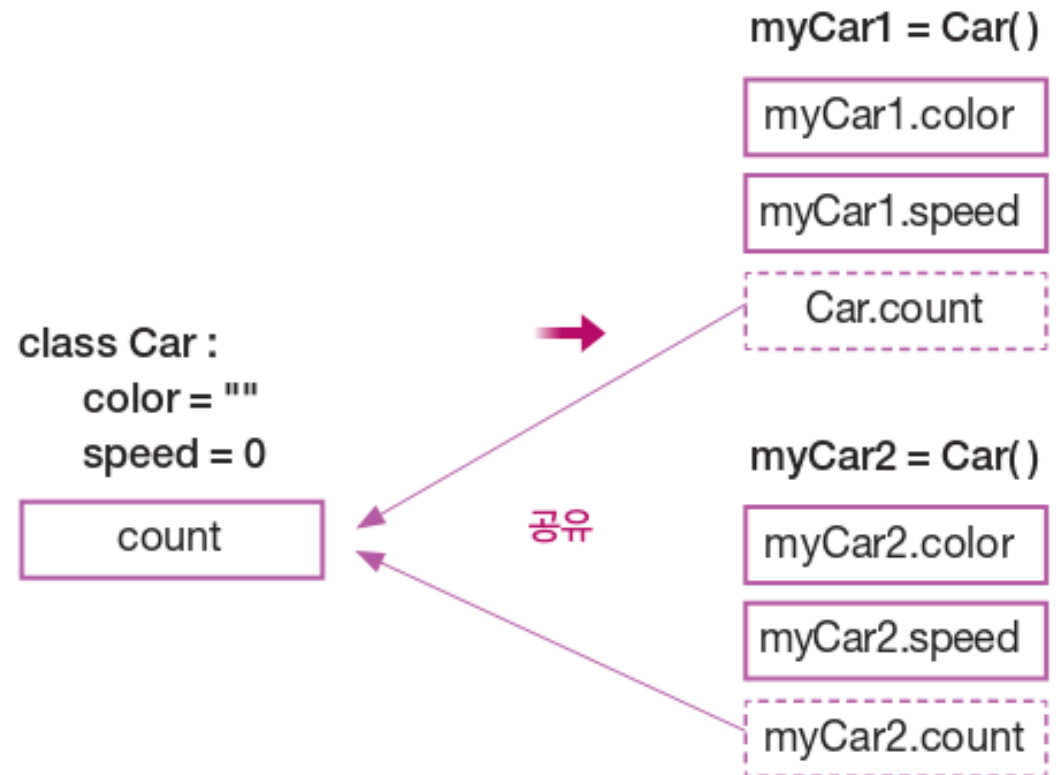
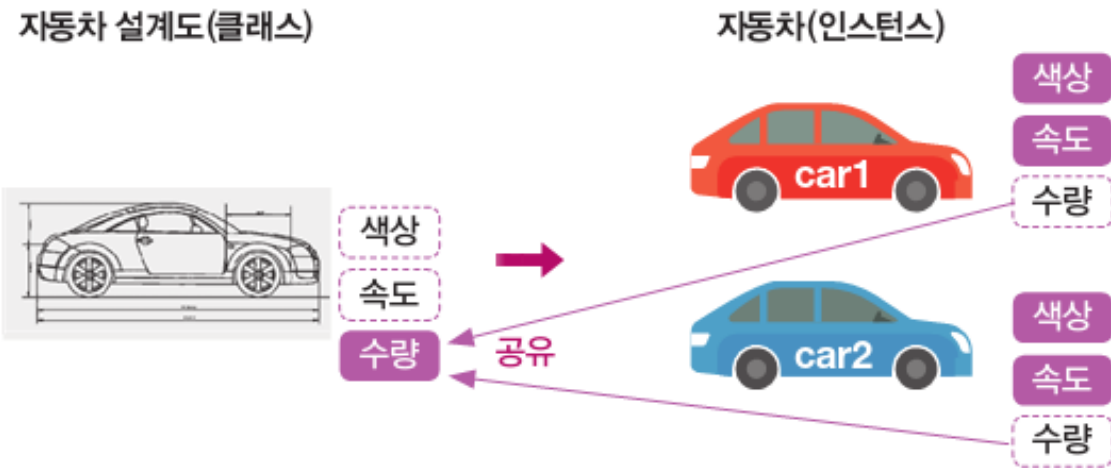
myCar2.color

myCar2.speed

Class Variables

- Class variables

- Car.count



Class Variables

- Exercise

Code12-06.py

```
1  ## 클래스 선언 부분 ##
2  class Car :
3      color = ""    # 인스턴스 변수
4      speed = 0     # 인스턴스 변수
5      count = 0     # 클래스 변수
6
7      def __init__(self) :
8          self.speed = 0
9          Car.count += 1
10
```

Class Variables

- (Cont'd)

```
11  ## 변수 선언 부분 ##
12  myCar1, myCar2 = None, None
13
14  ## 메인 코드 부분 ##
15  myCar1 = Car()
16  myCar1.speed = 30
17  print("자동차1의 현재 속도는 %dkm, 생산된 자동차는 총 %d대입니다." % (myCar1.speed,
    Car.count))
18
19  myCar2 = Car()
20  myCar2.speed = 60
21  print("자동차2의 현재 속도는 %dkm, 생산된 자동차는 총 %d대입니다." % (myCar2.speed,
    myCar2.count))
```

출력 결과

자동차1의 현재 속도는 30km, 생산된 자동차는 총 1대입니다.
자동차2의 현재 속도는 60km, 생산된 자동차는 총 2대입니다.

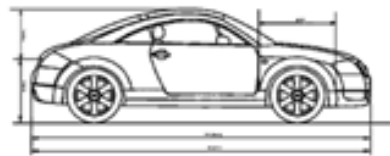
12.5 상속

Inheritance

- Inheritance (상속)

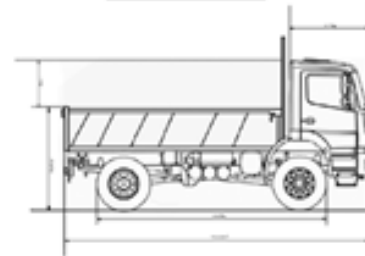
- 기존 클래스에 있는 field와 method를 그대로 물려받는 새로운 클래스를 만드는 것

승용차 클래스



class 승용차:
 필드 - 색상, 속도, 좌석 수
 메서드 - 속도 올리기()
 속도 내리기()
 좌석 수 알아보기()

트럭 클래스



class 트럭:
 필드 - 색상, 속도, 적재량
 메서드 - 속도 올리기()
 속도 내리기()
 적재량 알아보기()

Inheritance

- Examples

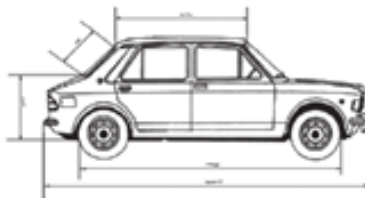
부모 클래스	자식 클래스
Animal(동물)	Lion(사자), Dog(개), Cat(고양이)
Bike(자전거)	MountainBike(산악자전거), RoadBike, TandemBike
Vehicle(탈것)	Car(자동차), Bus(버스), Truck(트럭), Boat(보트), Motorcycle(오토바이), Bicycle(자전거)
Student(학생)	GraduateStudent(대학원생), UnderGraduate(학부생)
Person(사람)	Student(학생), Employee(직원)
Shape(도형)	Rectangle(사각형), Triangle(삼각형), Circle(원)

Inheritance

- Inheritance

class 자동차 :
필드 - 색상, 속도
메서드 - 속도 올리기()
속도 내리기()

자동차 클래스(공통 내용)



Parent class
Super class

```
class 서브_클래스(슈퍼_클래스) :  
    # 이 부분에 서브 클래스의 내용 코딩
```

상속



상속

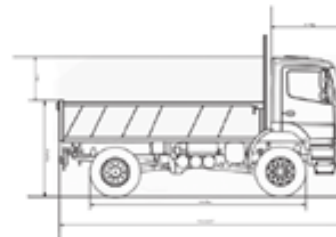


승용차 클래스



class 승용차 : 자동차 상속
필드 - 자동차 필드, 좌석 수
메서드 - 자동차 메서드
좌석 수 알아보기()

트럭 클래스



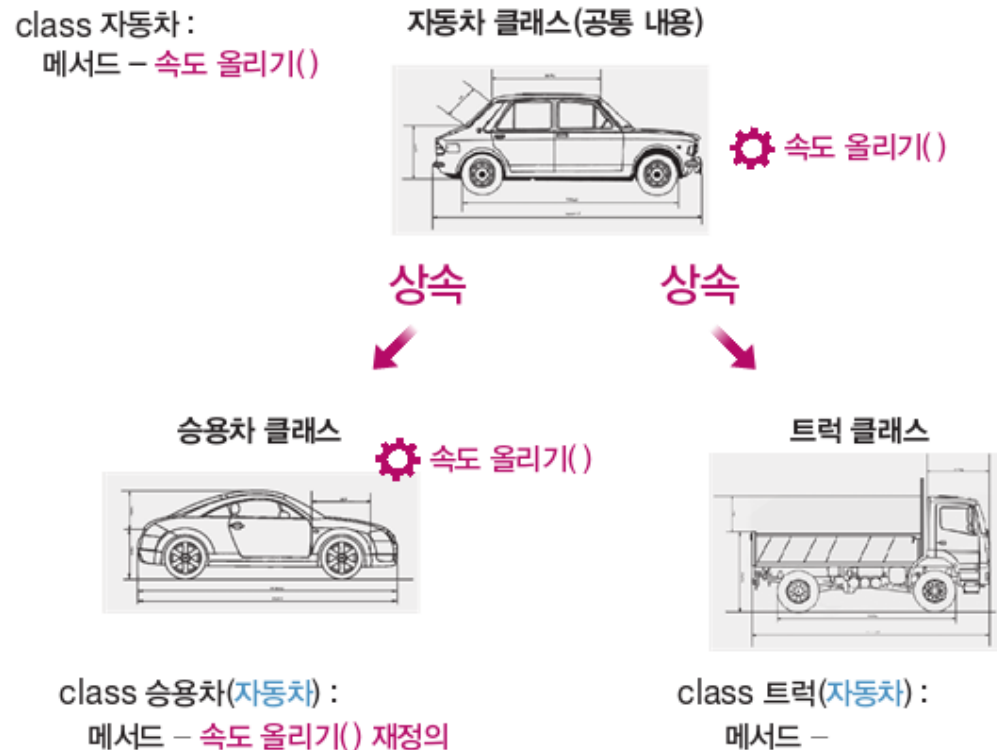
class 트럭 : 자동차 상속
필드 - 자동차 필드, 적재량
메서드 - 자동차 메서드
적재량 알아보기()

Child class
Sub class

Inheritance

- Method overriding

- Re-defining the methods of the parent class within the child class



Inheritance

Code12-07.py

```
1  ## 클래스 선언 부분 ##
2  class Car :
3      speed = 0
4      def upSpeed(self, value) :
5          self.speed += value
6
7          print("현재 속도(슈퍼 클래스) : %d" % self.speed)
8
9  class Sedan( Car ) :
10     def upSpeed(self, value) :
11         self.speed += value
12
13         if self.speed > 150 :
14             self.speed = 150
15
```

```
16         print("현재 속도(서브 클래스) : %d" % self.speed)
17
18 class Truck(Car) :
19     pass
20
21 ## 변수 선언 부분 ##
22 sedan1, truck1 = None, None
23
24 ## 메인 코드 부분 ##
25 truck1 = Truck()
26 sedan1 = Sedan()
27
28 print("트럭 --> ", end = "")
29 truck1.upSpeed(200)
30
31 print("승용차 --> ", end = "")
32 sedan1.upSpeed(200)
```

출력 결과

트럭 --> 현재 속도(슈퍼 클래스) : 200

승용차 --> 현재 속도(서브 클래스) : 150

Inheritance

- 생성자 overriding

```
class Animal:
    def __init__(self, age=0):
        self.age=age

    def eat(self):
        print("동물이 먹고 있습니다. ")

class Dog(Animal):
    def __init__(self, age=0, name=""):
        self.name=name

d = Dog()
print(d.age)
```

AttributeError: 'Dog' object has no attribute 'age'

```
class Animal:
    def __init__(self, age=0):
        self.age=age

    def eat(self):
        print("동물이 먹고 있습니다. ")

class Dog(Animal):
    def __init__(self, age=0, name=""):
        super().__init__(age)
        self.name=name

d = Dog()
print(d.age)
```

Inheritance

- `type()` 함수

```
x = Animal()  
y = Dog()  
print(type(x))  
print(type(y))
```

```
<class '__main__.Animal'>  
<class '__main__.Dog'>
```


Inheritance

- Multiple inheritance

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

    def show(self):
        print(self.name, self.age)
```

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

    def getID(self):
        return self.id
```

```
class CollegeStudent(Person, Student):
    def __init__(self, name, age, id):
        Person.__init__(self, name, age)
        Student.__init__(self, id)
```

```
obj = CollegeStudent('Kim', 22, '100036')
obj.show()
print(obj.getID())
```

12.6 객체지향 프로그래밍의 심화내용

Methods

- **Special methods**

- `__init__()`: constructor (생성자)
- `__del__()`: destructor (소멸자)
- `__repr__()`: called when `print(instance)` appears (`__str__`)
- `__add__()`: called for addition (+) between instances
- `__lt__()`, `__le__()`, `__gt__()`, `__ge__()`, `__eq__()`, `__ne__()`
 - called for `<`, `<=`, `>`, `>=`, `==`, `!=`

Methods

- Exercise

Code12-09.py

```
1  ## 클래스 선언 부분 ##
2  class Line :
3      length = 0
4      def __init__(self, length) :
5          self.length = length
6          print(self.length, '길이의 선이 생성되었습니다.')
7
8      def __del__(self) :
9          print(self.length, '길이의 선이 삭제되었습니다.')
```

```
11     def __repr__(self) :
12         return '선의 길이 : ' + str(self.length)
13
14     def __add__(self, other) :
15         return self.length + other.length
16
17     def __lt__(self, other) :
18         return self.length < other.length
19
20     def __eq__(self, other) :
21         return self.length == other.length
22
23  ## 메인 코드 부분 ##
24  myLine1 = Line(100)
25  myLine2 = Line(200)
```

Methods

- (Cont'd)

```
27 print(myLine1)
28
29 print('두 선의 길이 합 : ', myLine1 + myLine2)
30
31 if myLine1 < myLine2 :
32     print('선분 2가 더 기네요.')
33 elif myLine1 == myLine2 :
34     print('두 선분이 같네요.')
```

```
35 else :
36     print('모르겠네요.')
37
38 del(myLine1)
```

출력 결과

100 길이의 선이 생성되었습니다.
200 길이의 선이 생성되었습니다.
선의 길이 : 100
두 선의 길이 합 : 300
선분 2가 더 기네요.
100 길이의 선이 삭제되었습니다.

Methods

- **Abstract (추상) methods**

- Super class에서는 비어 있음
- Sub class에서 overriding하여 사용
- Overriding하지 않을 경우 예외 발생

```
def method(self) :  
    raise NotImplementedError()
```

출력 결과

NotImplementedError

Code12-10.py

```
1  ## 클래스 선언 부분 ##  
2  class SuperClass :  
3      def method(self) :  
4          pass  
5  
6  class SubClass1 (SuperClass) :  
7      def method(self) :          # 메서드 오버라이딩  
8          print('SubClass1에서 method()를 오버라이딩함')  
9  
10 class SubClass2 (SuperClass) :  
11     pass  
12  
13 ## 메인 코드 부분 ##  
14 sub1 = SubClass1()  
15 sub2 = SubClass2()  
16  
17 sub1.method()  
18 sub2.method()
```

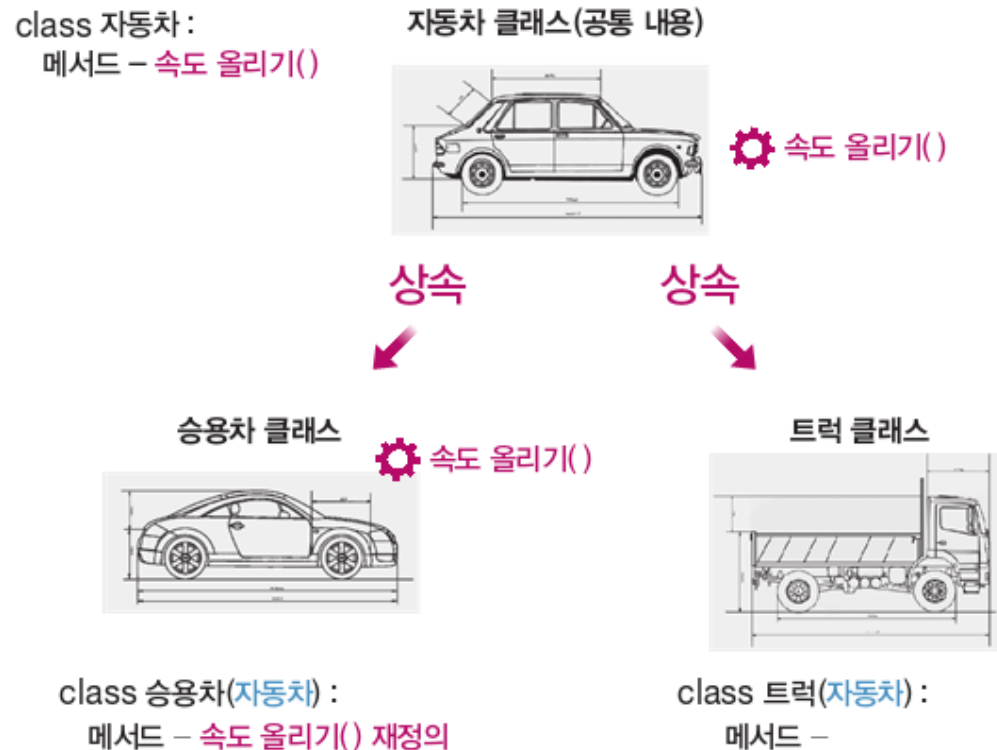
출력 결과

SubClass1에서 method()를 오버라이딩함

Summary

- Method overriding

- Re-defining the methods of the parent class within the child class



Summary

- **Special methods**

- `__init__()`: constructor (생성자)
- `__del__()`: destructor (소멸자)
- `__repr__()`: called when `print(instance)` appears
- `__add__()`: called for addition (+) between instances
- `__lt__()`, `__le__()`, `__gt__()`, `__ge__()`, `__eq__()`, `__ne__()`
 - called for `<`, `<=`, `>`, `>=`, `==`, `!=`

Assignment 15

- **“BankAccount” class**

- class BankAccount
 - 속성변수 (인스턴스): name, balance
 - 클래스 변수: accounts (총 계좌의 수)
- class SavingsAccount(BankAccount)
 - 예금계좌, 입금시 이자율 (interest_rate) 만큼 이자 지급
 - 이자율 0.1 → 5000원 입금시 500원 이자를 balance에 추가
- class CheckingAccount(BankAccount)
 - 당좌수표계좌, 출금 (수표 발행)시 비용 발생 (withdraw_charge)
 - 출금 비용 500원 → 매 출금시 balance에서 500원 추가 지출

```
from bankaccount import BankAccount, SavingsAccount, CheckingAccount
```

```
kim = BankAccount("Kim", 10000)
print(f"총 계좌 수: {BankAccount.accounts}")
kim.deposit(1000)
kim.withdraw(2000)
print(kim)
print("-".center(20, "-"))
```

```
lee = SavingsAccount("Lee", 5000, 0.1)
print(f"총 계좌 수: {BankAccount.accounts}")
lee.deposit(1000)
print(lee)
print("-".center(20, "-"))
```

```
cho = CheckingAccount("Cho", 7000, 500)
print(f"총 계좌 수: {BankAccount.accounts}")
cho.withdraw(5000)
print(cho)
print("-".center(20, "-"))
```

```
del(lee)
print(f"총 계좌 수: {BankAccount.accounts}")
print("-".center(20, "-"))
```

총 계좌 수: 1
Kim 계좌 잔액 9000

총 계좌 수: 2
Lee 계좌 잔액 6100

총 계좌 수: 3
Cho 계좌 잔액 1500

Lee 계좌 해지
총 계좌 수: 2

Assignment 15

- **Methods**

- `class BankAccount`
 - `__init__(self, name, balance)`
 - 클래스 변수 `accounts` 동작 구현
 - `deposit(self, amount)`
 - `withdraw(self, amount)`
 - `__repr__(self)`
 - `__del__(self)`
 - 클래스 변수 `accounts` 동작 구현

Assignment 15

- **Methods**

- `class SavingsAccount`
 - `__init__(self, name, balance, interest_rate)`
 - 부모 클래스의 생성자 활용
 - 속성변수 `self.interest_rate` 추가 정의
 - `deposit(self, amount)`
 - Method overriding
 - 부모 클래스의 `deposit` method 활용

Assignment 15

- **Methods**

- `class CheckingAccount`
 - `__init__(self, name, balance, withdraw_charge)`
 - 부모 클래스의 생성자 활용
 - 속성변수 `self.withdraw_charge` 추가 정의
 - `withdraw(self, amount)`
 - Method overriding
 - 부모 클래스의 `withdraw` method 활용