7. 객체지향 프로그래밍

순천향대학교 컴퓨터공학과
이 상 정

학습 내용

- 클래스 기초 학습
  - 코드와 코드가 처리할 데이터를 함께 정의
- 클래스 메서드 (method)
- 레퍼런스 (reference)
- 생성자 (constructor)
- 상속 (inheritance)
- 정의 및 생성 (creation)

클래스 (Class)

- 함수와 함수가 처리하는 데이터를 결합하면?
  - 코드가 커지더라도 복잡도를 줄일 수 있음
  - 복잡도가 줄면 에러도 줄어듬
  - 에러가 적으면 유지 보수도 쉬워짐

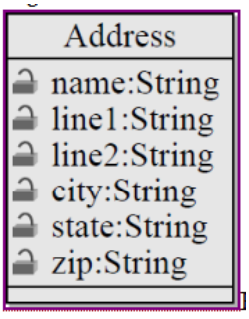
- 클래스 (class)
  - 객체지향 프로그래밍 (object-oriented programming)에서 사용
  - 코드와 코드가 처리할 데이터를 함께 정의
  - 클래스가 정의되면 클래스를 사용하여 데이터 객체(인스턴스)를 생성
  - 객체지향에서는 코드를 메서드(method), 데이터를 속성(attribute), 생성된 객체를 인스턴스 라고 함

간단한 클래스의 정의와 생성

- Address 클래스 예
  - 이름(name), 집 주소(street), 시(city), 주(state), 우편번호(zip)

```
class Address():
    name=""
    line1=""
    line2=""
    city=""
    state=""
    zip=""
```

클래스 다이어그램



  - Address 클래스 이름
  - 클래스 내의 변수는 클래스 속성(attribute) 또는 필드(field)라 함
    - name, line1, line2, city, state

클래스의 생성

- Address 클래스의 인스턴스(instance), 객체(object) 생성
  - homeAddress , vacationHomeAddress

```
8  # Create an address
9  homeAddress=Address()
10 # Set the fields in the address
11 homeAddress.name="John Smith"
12 homeAddress.line1="701 N. C Street"
13 homeAddress.line2="Carver Science Building"
14 homeAddress.city="Indianola"
15 homeAddress.state="IA"
16 homeAddress.zip="50125"
17
```

```
18 # Create another address
19 vacationHomeAddress=Address()
20 # Set the fields in the address
21 vacationHomeAddress.name="John Smith"
22 vacationHomeAddress.line1="1122 Main Street"
23 vacationHomeAddress.line2=""
24 vacationHomeAddress.city="Panama City Beach"
25 vacationHomeAddress.state="FL"
26 vacationHomeAddress.zip="32407"
27 print ("The client's main home is in "+homeAddress.city)
28 print ("His vacation home is in "+vacationHomeAddress.city)
29
30
```


간단한 클래스 생성 코드 (1) [실습 1]

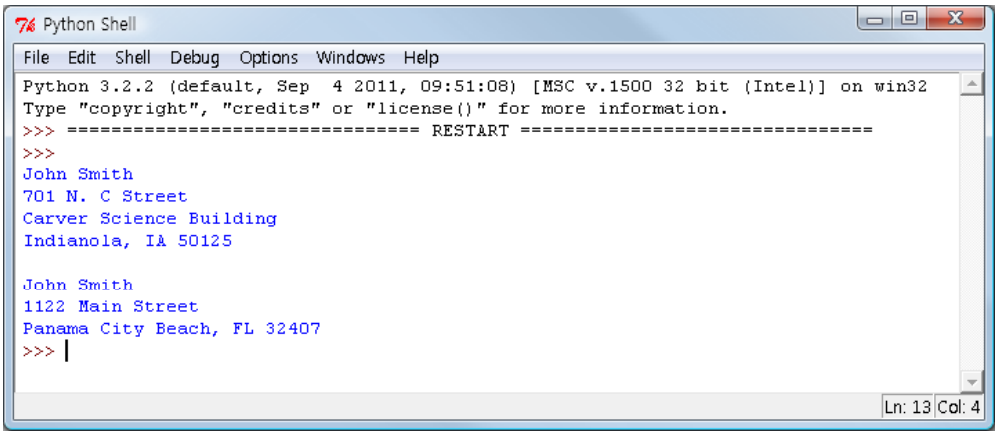
```
class Address():
    name=""
    line1=""
    line2=""
    city=""
    state=""
    zip=""

# Address 인스턴스 생성
homeAddress=Address()

# Address 속성 값 지정
homeAddress.name="John Smith"
homeAddress.line1="701 N. C Street"
homeAddress.line2="Carver Science Building"
homeAddress.city="Indianola"
homeAddress.state="IA"
homeAddress.zip="50125”
```

```
# 또 다른 Address 인스턴스 생성
vacationHomeAddress=Address()

# Address 속성 값 지정
vacationHomeAddress.name="John Smith"
vacationHomeAddress.line1="1122 Main
    Street"
vacationHomeAddress.line2=""
vacationHomeAddress.city="Panama City
    Beach"
vacationHomeAddress.state="FL"
vacationHomeAddress.zip="32407“
```

간단한 클래스 생성 코드 (2)

```
# 주소 출력
def printAddress(address):
    print (address.name)
    # If there is a line1 in the address, print it
    if ( len(address.line1) > 0 ):
        print (address.line1)
    # If there is a line2 in the address, print it
    if ( len(address.line2) > 0 ):
        print( address.line2 )
    print( address.city+", "+address.state+" "+address.zip )

printAddress( homeAddress )
print()
printAddress( vacationHomeAddress )
```


시험주행



메서드 (Method)

- 클래스는 속성 뿐만 아니라 메서드를 가짐
  - 메서드는 클래스 내부의 함수로 속성 다음에 정의
  - 메서드의 첫번째 인수는 self
    - 호출 시에는 전달되지 않고, 객체 자신의 레퍼런스를 의미

- Dog 클래스 예
  - bark() 메서드

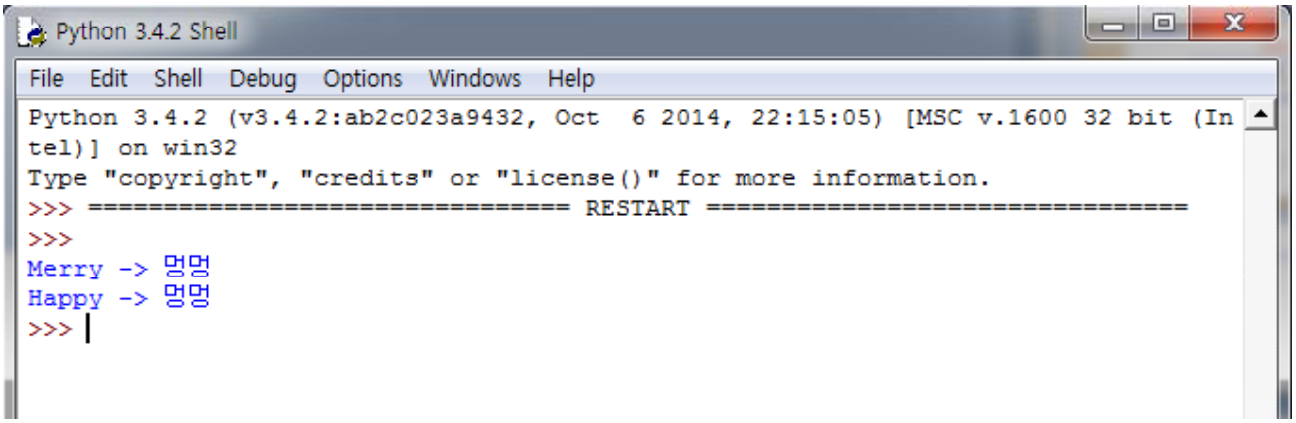
```
class Dog():
    age=0
    name=""
    weight=0

    def bark(self):
        print("%s -> 멍멍" % self.name)
```


메서드 사용 코드 예 [실습 2]

```
class Dog():
    # 속성
    age=0
    name=""
    weight=0
     # 메서드
    def bark(self):
        print("%s -> 멍멍" % self.name)

myDog = Dog()    # 인스턴스 생성
myDog.name = "Merry"
myDog.weight = 20
myDog.age = 3
myDog.bark()      # 메서드 호출
```

```
yourDog = Dog()     # 인스턴스 생성
yourDog.name = "Happy"
yourDog.weight = 30
yourDog.age = 5
yourDog.bark()      # 메서드 호출
```

시험주행



커피바 POS (1) [실습 3]

- 6장의 헬스 클럽 코드를 객체지향 프로그래밍으로 수정
  - Transactions 클래스 정의

- 거래 은행의 레코드 포맷
  - 처음 7글자 가격
  - 다음 16글자 신용카드 번호
  - 다음 16글자는 판매한 물건 설명

```
class Transactions():
    # 속성
    price = 0.0
    credit_card = 0
    item = 0
    # 메서드
    def save(self, cost, card, itm):
        self.price = cost
        self.credit_card = card
        self.item = itm
        self.save_file()
    def save_file(self):
        file = open("transactions.txt", "a")
        file.write("%07d%16s%16s₩n" % (self.price * 100, self.credit_card, self.item))
        file.close()
```

커피바 POS (2)

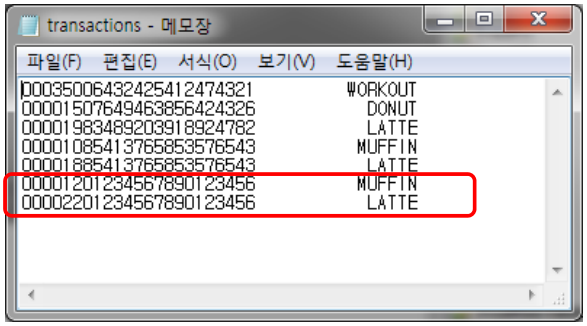
```
class Transactions():
    # 속성
    price = 0.0
    credit_card = 0
    item = 0
    # 메서드
    def save(self, cost, card, itm):
        self.price = cost
        self.credit_card = card
        self.item = itm
        self.save_file()
    def save_file(self):
        file = open("transactions.txt", "a")
        file.write("%07d%16s%16s₩n" % (self.price * 100, self.credit_card, self.item))
        file.close()

items   = ["DONUT", "LATTE", "FILTER", "MUFFIN"]
prices  = [1.50, 2.20, 1.80, 1.20]
running = True
```


커피바 POS (3)

```
while running:
    option = 1
    for choice in items:
        print(str(option) + ". " + choice)
        option = option + 1
    print(str(option) + ". Quit")
    choice = int(input("Choose an option: "))
    if choice == option:
        running = False
    else:
        card = input("Credit card number: ")
        trans = Transactions()
        trans.save(prices[choice-1], card, items[choice - 1])
```

시험주행

```
Python 3.4.2 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2014, 22:15:05)
tel)] on win32
Type "copyright", "credits" or "license()" for more infor
>>> ================================ RESTART ============
>>>
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 4
Credit card number: 1234567890123456
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 2
Credit card number: 1234567890123456
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 5
>>>
```



레퍼런스 (Refernece)

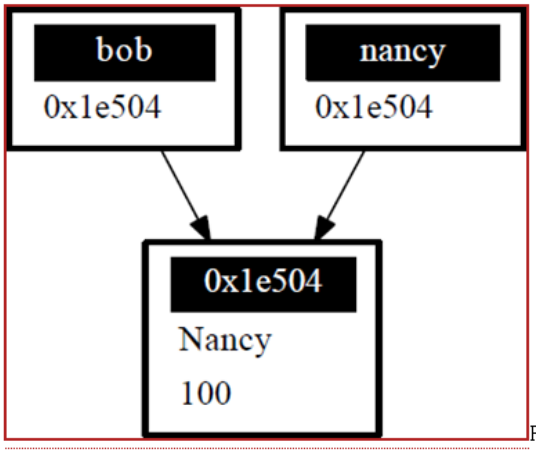
- 레퍼런스(reference, 참조자)
  - 객체를 참조
  - 주소(address), 포인터(pointer), 핸들(handle)
  - 예
    - bob은 Person 객체의 레퍼런스

```
1  class Person:
2      name=""
3      money=0
4
5  bob = Person()
6  bob.name="Bob"
7  bob.money=100
```

레퍼런스 예

```
9  nancy = bob
10 nancy.name="Nancy"
11
12 print(bob.name,"has",bob.money,"dollars.")
13 print(nancy.name,"has",nancy.money,"dollars.")
```

```
Nancy has 100 dollars.
Nancy has 100 dollars.
```



함수와 레퍼런스 [실습 4]

```
1  def giveMoney1(money):
2      money += 100
3
4  class Person:
5      name=""
6      money=0
7
8  bob = Person()
9  bob.name="Bob"
10 bob.money=100
11
12 giveMoney1(bob.money)
13 print(bob.money)
```

=> 100

```
14 def giveMoney2(person):
15     person.money += 100
16
17 giveMoney2(bob)
18 print(bob.money)
```

=> 200

생성자 (Constructor)

❑ 생성자 (constructor)

- 클래스의 인스턴스(instance)가 생성될 때 자동으로 호출
- 객체의 데이터(속성)를 초기화하는데 사용

```
1  class Dog():
2      name=""
3
4      # Constructor
5      # Called when creating an object of this type
6      def __init__(self):
7          print("A new dog is born!")
8
9  # This creates the dog
10 myDog = Dog()
```

```
A new dog is born!
```


생성자 예 [실습 5]

```
1  class Dog():
2      name=""
3
4      # Constructor
5      # Called when creating an object of this type
6      def __init__(self, newName):
7          self.name = newName
8
9  # This creates the dog
10 myDog = Dog("Spot")
11
12 # Print the name to verify it was set
13 print(myDog.name)
14
15 # This line will give an error because
16 # a name is not passed in.
17 herDog = Dog()
```

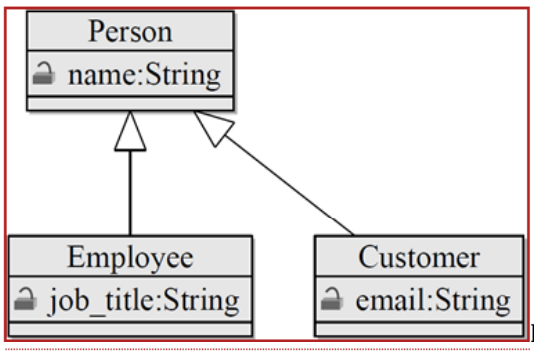
```
1  class Dog():
2      name="Rover"
3
4      # Constructor
5      # Called when creating an object of this type
6      def __init__(self, name):
7          # This will print "Rover"
8          print(self.name)
9          # This will print "Spot"
10         print(name)
11
12 # This creates the dog
13 myDog = Dog("Spot")
```

상속 (Inheritance)

상속 (inheritance)

- 자식 클래스(child class)가 부모 클래스(parent class)의 속성과 메서드를 상속
- 예
  - 부모 클래스 Person의 속성을 자식 클래스 Employee, Customer가 상속
    - 즉 자식 클래스에서 name 속성을 사용

```
1  class Person():
2      name=""
3
4  class Employee(Person):
5      job_title=""
6
7  class Customer(Person):
8      email=""
9
10 johnSmith = Person()
11 johnSmith.name = "John Smith"
12
13 janeEmployee = Employee()
14 janeEmployee.name = "Jane Employee"
15 janeEmployee.job_title = "Web Developer"
16
17 bobCustomer = Customer()
18 bobCustomer.name = "Bob Customer"
19 bobCustomer.email = "send_me@spam.com"
```



메서드의 상속 예 [실습 6]

아래 예를 실행하면 "Person created" 가 3번 출력됨

- 부모의 __init__() 메서드를 자식이 상속

```
class Person():
    name=""

    def __init__(self):
        print("Person created")

class Employee(Person):
    job_title=""

class Customer(Person):
    email=""

johnSmith = Person()
janeEmployee = Employee()
bobCustomer = Customer()
```

오버라이드(override) [실습 7]

❑ 오버라이드 (override)

- 자식 클래스의 메서드가 부모 클래스의 메서드를 대치

```
class Person():
    name=""

    def __init__(self):
        print("Person created")

class Employee(Person):
    job_title=""

    def __init__(self):
        print("Employee created")

class Customer(Person):
    email=""

    def __init__(self):
        print("Customer created")

johnSmith = Person()
janeEmployee = Employee()
bobCustomer = Customer()
```

```
class Person():
    name=""

    def __init__(self):
        print("Person created")

class Employee(Person):
    job_title=""

    def __init__(self):
        Person.__init__(self)
        print("Employee created")

class Customer(Person):
    email=""

    def __init__(self):
        Person.__init__(self)
        print("Customer created")

johnSmith = Person()
janeEmployee = Employee()
bobCustomer = Customer()
```


수정된 커피바 POS [실습 8]

❑ 수정된 커피바 POS 프로그램

- 스타버즈 할인 카드 제시 고객에 추가로 10% 할인
- 부모 클래스 Transactions
  - 앞의 코드와 동일
- 자식 클래스 Discount_trans
  - 부모 클래스 Transactions을 상속
  - 부모 클래스의 생성자 호출
  - 부모의 save() 메서드를 오버라이드

```
# 자식 클래스
class Discount_trans(Transactions):
    # 생성자
    def __init__(self, cost, card, itm):
        Transactions.__init__(self, cost, card, itm)
    # 오버라이드 메서드
    def save(self):
        file = open("transactions.txt", "a")
        file.write("%07d%16s%16s₩n" % (self.price * 100 * 0.9, self.credit_card, self.item))
        file.close()
```

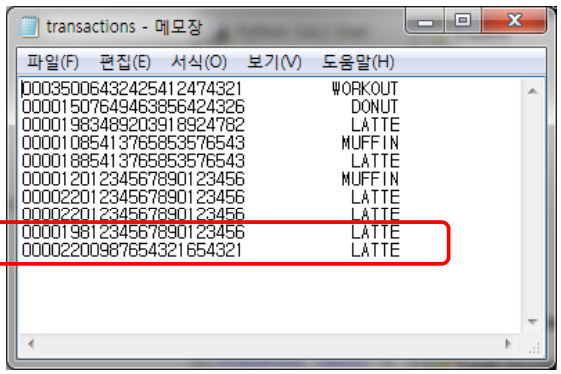
```
# 부모 클래스
class Transactions():
    # 속성
    price = 0.0
    credit_card = 0
    item = 0
    # 생성자
    def __init__(self, cost, card, itm):
        self.price = cost
        self.credit_card = card
        self.item = itm
    # 메서드
    def save(self):
        file = open("transactions.txt", "a")
        file.write("%07d%16s%16s₩n" % (self.price * 100, self.credit_card, self.item))
        file.close()

# 자식 클래스
class Discount_trans(Transactions):
    # 생성자
    def __init__(self, cost, card, itm):
        Transactions.__init__(self, cost, card, itm)
    # 오버라이드 메서드
    def save(self):
        file = open("transactions.txt", "a")
        file.write("%07d%16s%16s₩n" % (self.price * 100 * 0.9, self.credit_card, self.item))
        file.close()
```



```
items   = ["DONUT", "LATTE", "FILTER", "MUFFIN"]
prices  = [1.50, 2.20, 1.80, 1.20]
running = True

while running:
    option = 1
    for choice in items:
        print(str(option) + ". " + choice)
        option = option + 1
    print(str(option) + ". Quit")
    choice = int(input("Choose an option: "))
    if choice == option:
        running = False
    else:
        card = input("Credit card number: ")
        if input("Starbuzz card? ") == "Y":
            trans = Discount_trans(prices[choice-1], card, items[choice - 1])
            trans.save()
        else:
            trans = Transactions(prices[choice-1], card, items[choice - 1])
            trans.save()
```

시험주행

```
Python 3.4.2 Shell
File  Edit  Shell  Debug  Options  Windows  Help
Python 3.4.2 (v3.4.2:ab2c023a9432, Oct  6 2
tel)] on win32
Type "copyright", "credits" or "license()"
>>> ================================ RESTAR
>>>
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 2
Credit card number: 1234567890123456
Starbuzz card? Y
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 2
Credit card number: 0987654321654321
Starbuzz card? N
1. DONUT
2. LATTE
3. FILTER
4. MUFFIN
5. Quit
Choose an option: 5
>>>
```



7장 실습

[실습 1] 간단한 클래스 생성 코드
[실습 2] 메서드 사용 코드 예
[실습 3] 커피바 POS [실습 3]
[실습 4] 함수와 레퍼런스
[실습 5] 생성자 예
[실습 6] 메서드의 상속 예
[실습 7] 오버라이드 예
[실습 8] 수정된 커피바 POS

과제

1. 앞에서 배운 내용을 사용한 임의의 프로그램 작성
   - 프로그램 설명
   - 프로그램 소스
   - 실행 결과