Report

Introduction to Bigdata Programming
STS2011-01

20180594 이정훈
경제학부

이번 과제는 상속 클래스를 이용하여 회사를 만드는 과제였습니다. 최상위 클래스를 Person, 그 다음 층위를 Employee, 최하위 클래스로 Manager와 Satff, Hourly를 지정하여 조직도를 만드는 것이 최종 목적이었습니다.

```
import time
import functools
import re
from datetime import datetime

class Person :
    manager, staff, parttime = 0, 0, 0
    worker = manager + staff + parttime
    td = datetime.today()

    def __init__(self, name, age, gender) :
        self._Name = name
        self._Age = age
        self._Gender = gender

    @property
    def Name(self) :
        return self._Name

    @property
    def Age(self) :
        return self._Age

    @property
    def Gender(self) :
        return self._Gender

    def aboutMe(self) :
        print('이름은 %s 이고, 나이는 %s살 입니다.' %(self._Name, self._Age))

class Employee(Person) :
    def __init__(self, name, age, gender, salary, hiredate) :
        Person.__init__(self, name, age, gender)
        self.Salary = float(re.sub('[,]', '', salary))
        self.Hiredate = hiredate
        Person.worker += 1

    def goWork(self) :
        print('출근했습니다.')
        print('출근시간 [%i-%i-%i %i:%i]' % (Person.td.year, Person.td.month, Person.td.day, Person.td.hour, Person.td

    def doWork(self) :
        t0 = time.time()
        print('>>열심히 일합니다.<<')
        Employee.Working()
        elapsed = (time.time() - t0)
        print('작업시간 : %.5f' % elapsed)

    def Working() :
        print('다음을 계산하시오.')

        import numpy as np
        a = np.random.randint(100)
        b = np.random.randint(100)
        print('%i + %i' %(a,b))
        ans = int(input('Answer : '))

        if ans == (a + b) : print('정답입니다!'); return 0
        else : print('다시 계산하세요.'); return Employee.Working()

    def leaveWork(self) :
        print('퇴근했습니다.')
        print('퇴근시간 [%i-%i-%i %i:%i]' % (Person.td.year, Person.td.month, Person.td.day, Person.td.hour, Person.td

    def aboutMe(self) :
        Person.aboutMe(self)
        print('급여는 %i원이고, 입사일은 %s입니다.' %(self.Salary, self.Hiredate))
```

Person Class

aboutMe( ) : 간단한 자기소개를 하는 함수입니다.

Employee Class

goWork( ) : 객체가 출근합니다. 출근시간이 출력됩니다.
doWork( ) : 객체가 일을 합니다. 실질적인 일처럼 보이기 위해, Working( )함수를 통해 간단한 덧셈문제를 실행자가 직접 풀게 함으로써 연산에 걸리는 시간을 저장해 작업시간을 구현하였습니다.
leaveWork( ) : 객체가 퇴근합니다. 퇴근시간이 출력됩니다.
aboutMe( ) : Person class의 aboutMe를 불러와 출력하고, 하위 항목으로 급여와 입사일을 출력합니다.

```
class Manager(Employee) :
    def __init__(self, name, age, gender, salary, hiredate, department, career) :
        Employee.__init__(self, name, age, gender, salary, hiredate)
        self.department = department
        self.career = int(career)
        if self.career >= 10 : self.Salary *= 1.3
        Person.manager += 1

    def infoEmployees(self) :
        print('총 직원 수 : %i명' % (Person.worker))
        print('매니저 : %i명' % (Person.manager))
        print('스태프 : %i명' % (Person.staff))
        print('파트타임 : %i명' % (Person.parttime))

    def goHome(self) :
        print('직원들에게 퇴근하자고 합니다.')
        print('총 %i명의 직원이 퇴근합니다.' %(Person.worker))
        print('퇴근시간 [%i-%i-%i %i:%i]' % (Person.td.year, Person.td.month, Person.td.day, Person.td.hour, Person.td

    def aboutMe(self) :
        print('*' * 20, 'Manager Information', '*' * 20)
        Employee.aboutMe(self)
        print('부서는 %s이고, 경력은 %s년입니다.' %(self.department, self.career))
        if self.career >= 10 : print('[NOTICE] 경력이 10년 이상이므로 임금이 30% 자동 인상되었습니다');

    def whenStrike(self, nego) :
        print('*' * 20, 'Manager\'s Choice', '*' * 20)
        TF = input('급여를 {}% 인상하시겠습니까? (y/n) : '.format(nego))
        return TF

    def whenComplain(self) :
        print('*' * 20, 'Manager\'s Choice', '*' * 20)
        TF = input('고용노동부에서 민원이 제기되었다고 연락이 왔습니다. \n출석하여 항변하시겠습니까? (y/n) : ')
        return TF
```

최하위 클래스 (Manager, Staff, Hourly)

최하위 클래스에는 각 직무별 능력을 별도로 부여하였습니다. 개성 넘치는 직원들과 함께 즐거운 회사 생활을 합시다!

Manager Class

infoEmployees( ) : 매니저는 직원의 정보를 확인할 수 있습니다. 총 직원, 매니저, 스태프, 파트타임의 인원수를 확인할 수 있습니다.
goHome( ) : 매니저의 권한으로 직원을 모두 퇴근시킬 수 있습니다! 퇴근시간이 출력됩니다.
aboutMe( ) : Person과 Employee의 소개를 이어받아 부서와 경력을 알 수 있습니다. 10년 이상이 넘은 경력자는 급여가 30% 자동 인상됩니다.
whenStrike( ) : 파트타임직의 직원이 파업을 할 경우 급여 인상 결정권이 있습니다.

whenComplaint( ) : 파트타임직의 직원이 고용노동부에 민원을 제기할 경우 항변 여부를 결정할 수 있습니다.

```
class Staff(Employee) :
    def __init__(self, name, age, gender, salary, hiredate, department, career) :
        Employee.__init__(self, name, age, gender, salary, hiredate)
        self.department = department
        self.career = career
        Person.staff += 1

    def leaveCorp(self) :
        if Person.staff < 1 :
            Person.staff = 0
            print('회사에 남아있는 스태프가 없습니다.')
        else :
            print('회사 일에 지쳐 사표를 내려고 합니다.')
            TF = input('매니저에게 사표를 던지시겠습니까?(y/n) : ')
            if TF.lower() == 'y' : print('사표에 맞은 매니저가 통증을 호소합니다. 사표가 수리됩니다.'); Person.staff -= 1; print('
            elif TF.lower() == 'n' : print('무단결근으로 처리되다 해고당했습니다.'); Person.staff -= 1; print('회사를 퇴사합니다')
            else :
                print('잘못된 입력입니다. 다시 시도하세요')
                Staff.leaveCorp(self)
        Person.worker = Person.manager + Person.staff + Person.parttime

    def aboutMe(self) :
        print('*' * 20, 'Staff Information', '*' * 20)
        Employee.aboutMe(self)
        print('부서는 %s이고, 경력은 %s입니다.' %(self.department, self.career))
```

Staff Class

leaveCorp( ) : 직원 중 유일하게 회사를 퇴사할 수 있습니다. 매니저에게 사표를 던져봅시다! (주의 : 사직 의사 번복을 할 수 없습니다.)
aboutMe( ) : aboutMe( ) : Person과 Employee의 소개를 이어받아 부서와 경력을 알 수 있습니다.

```
class Hourly(Employee) :
    def __init__(self, name, age, gender, salary, hiredate, department, period) :
        Employee.__init__(self, name, age, gender, salary, hiredate)
        self.department = department
        self.period = period
        Person.parttime += 1

    def aboutMe(self) :
        print('*' * 20, 'Part time Employee Information', '*' * 20)
        Employee.aboutMe(self)
        print('부서는 %s이고, 계약기간은 %s입니다.' %(self.department, self.period))

    def Strike(self) :
        print('급여가 너무 적어서 파업에 들어갑니다')
        print('*' * 20, 'Employee\'s Negotiation', '*' * 20)
        print('매니저와 급여 협상에 들어갑니다.')
        nego = int(input('임금 인상률을 제시하십시오(%) : '))
        TF = Manager.whenStrike(self, nego)
        if TF == 'y' : print('급여 인상이 완료되었습니다.'); self.Salary *= (1 + nego * 0.01)
        elif TF == 'n' : print('급여 협상이 결렬되었습니다. 괘씸죄로 급여가 {}% 삭감됩니다.'.format(nego)); self.Salary *= (1 - ne
        else : print('잘못된 입력입니다. 다시 시도하세요'); Hourly.Strike(self)

    def Complaints(self) :
        print('근무환경에 불만이 있어 고용 노동청에 민원을 넣습니다.')
        TF = Manager.whenComplain(self)
        if TF == 'y' : print('매니저가 항변에 성공했습니다.\n민원을 넣은 사실이 적발되어 급여가 10% 삭감됩니다.'); self.Salary *= 0.9
        elif TF == 'n' : print('매니저가 항변을 포기했습니다. \n급여가 10% 인상됩니다'); self.Salary *= 1.1
        else : print('잘못된 입력입니다. 다시 시도하세요'); Hourly.Complaints()
        Person.worker = Person.manager + Person.staff + Person.parttime
```

Hourly Class

aboutMe( ) : aboutMe( ) : Person과 Employee의 소개를 이어받아 부서와 계약기간을 출력합니다.
Strike( ) : 급여가 적으면 파업을 할 수도 있습니다. 주의 ! 인상폭을 높게 요구할 경우 급여가 되려 깎일 수 있습니다.
Complaints( ) : 부당 노동 행위로 인해 고용노동부에 객체가 신고합니다. 성공하면 급여가 10% 상승, 실패하면 급여가 10% 깎입니다.

활용예

```
manager1 = Manager('김철수','45','남','4,500,000','2010년 1월 16일','경리부','12')
```

```
manager1.infoEmployees()
```

```
총 직원 수 : 1명
매니저 : 1명
스태프 : 0명
파트타임 : 0명
```

```
manager1.aboutMe()
```

```
******************** Manager Information ********************
이름은 김철수 이고, 나이는 45살 입니다.
급여는 5850000원이고, 입사일은 2010년 1월 16일입니다.
부서는 경리부이고, 경력은 12년입니다.
[NOTICE] 경력이 10년 이상이므로 임금이 30% 자동 인상되었습니다
```

```
manager1.doWork()
```

```
>>열심히 일합니다.<<
다음을 계산하시오.
7 + 77
Answer : 84
정답입니다!
작업시간 : 2.37315
```

```
manager1.goHome()
```

```
직원들에게 퇴근하자고 합니다.
총 1명의 직원이 퇴근합니다.
퇴근시간 [2022-10-11 22:37]
```

```
manager1.goWork()
```

```
출근했습니다.
출근시간 [2022-10-11 22:37]
```

```
manager1.leaveWork()
```

```
퇴근했습니다.
퇴근시간 [2022-10-11 22:37]
```

```
staff1 = Staff('이준성','35','남','3,500,000','2020년 10월 8일','회계부','5년')
```

```
staff1.doWork()
```

```
>>열심히 일합니다.<<
다음을 계산하시오.
77 + 32
Answer : 109
정답입니다!
작업시간 : 2.98555
```

```
part1 = Hourly('이성만','23','남','860,000','2022년 2월 22일','경리부','12개월')
```

```
part1.aboutMe()
```

```
******************** Part time Employee Information ********************
이름은 이성만 이고, 나이는 23살 입니다.
급여는 860000원이고, 입사일은 2022년 2월 22일입니다.
부서는 경리부이고, 계약기간은 12개월입니다.
```

```
part1.Strike()
```

```
급여가 너무 적어서 파업에 들어갑니다
******************** Employee's Negotiation ********************
매니저와 급여 협상에 들어갑니다.
임금 인상률을 제시하십시오(%) : 20
```

```
part1.aboutMe()
```

```
********** Part time Employee Information **********
이름은 이성만 이고, 나이는 23살 입니다.
급여는 860000원이고, 입사일은 2022년 2월 22일입니다.
부서는 경리부이고, 계약기간은 12개월입니다.
```

```
part1.Strike()
```

```
급여가 너무 적어서 파업에 들어갑니다
********** Employee's Negotiation **********
매니저와 급여 협상에 들어갑니다.
임금 인상률을 제시하십시오(%) : 20
********** Manager's Choice **********
급여를 20% 인상하시겠습니까? (y/n) : y
급여 인상이 완료되었습니다.
```

```
part1.aboutMe()
```

```
********** Part time Employee Information **********
이름은 이성만 이고, 나이는 23살 입니다.
급여는 1032000원이고, 입사일은 2022년 2월 22일입니다.
부서는 경리부이고, 계약기간은 12개월입니다.
```

```
part1.Complaints()
```

```
근무환경에 불만이 있어 고용 노동청에 민원을 넣습니다.
********** Manager's Choice **********
고용노동부에서 민원이 제기되었다고 연락이 왔습니다.
출석하여 항변하시겠습니까? (y/n) : y
매니저가 항변에 성공했습니다.
민원을 넣은 사실이 적발되어 급여가 10% 삭감됩니다.
```

```
part1.Salary
```

```
928800.0
```

```
part2 = Hourly('김준호','21','남','750,000','2021년 9월 16일','경리부','12개월')
```

```
manager1.infoEmployees()
```

```
총 직원 수 : 4명
매니저 : 1명
스태프 : 1명
파트타임 : 2명
```

After work

- **Class 상속**
  - 상속을 직접 이용해 이해가 빠르게 되도록 도왔습니다.
  - 헷갈리던 상속의 매커니즘을 이해했습니다.

- **@property**
  - property가 어떠한 기능을 하는지 다시한 번 확인할 수 있었습니다.