ECE12: Introduction to Programming
Lecture 19

Rainer Dömer

doemer@uci.edu

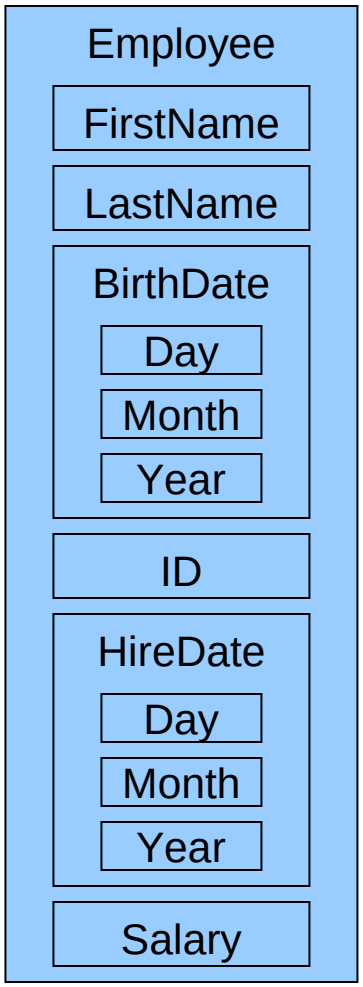
The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 19: Overview

- Object-oriented Programming
  - Membership composition
    - "Has a" relationship
  - Inheritance
    - "Is a" relationship
  - Concepts and terminology
    - Base class vs. derived class
    - Single vs. multiple inheritance
    - Polymorphism
    - Overriding methods
  - Example: Employee

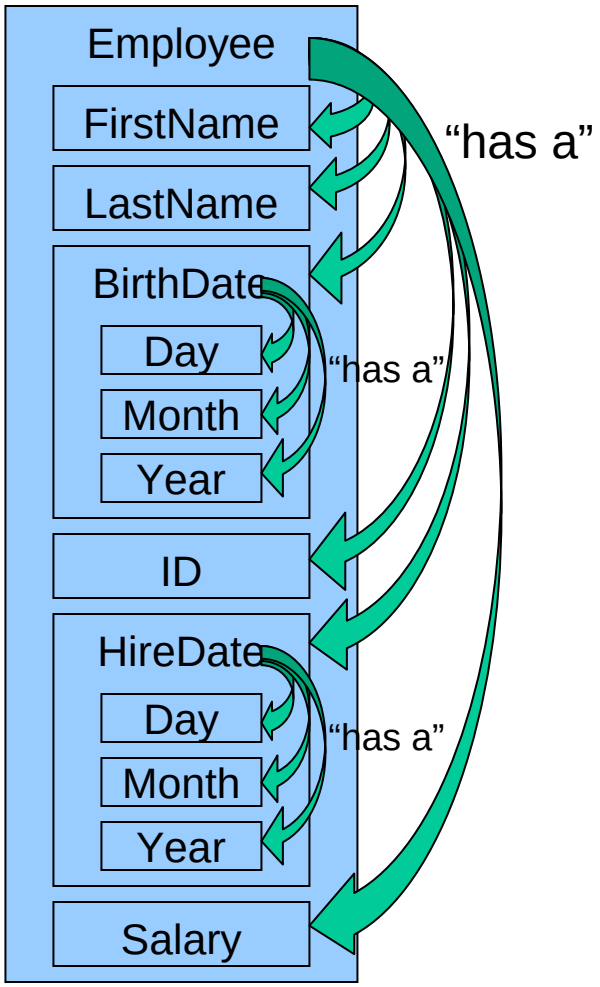
Object-Oriented Programming

- Membership composition
  - Example: Employee
    - class Employee:
      - String: FirstName
      - String: LastName
      - Date: BirthDate
        - » Integer: Year
        - » Integer: Month
        - » Integer: Day
      - Integer: ID
      - Date: HireDate
        - » Integer: Year
        - » Integer: Month
        - » Integer: Day
      - Floating point: Salary



Object-Oriented Programming

- Membership composition
  - Example: Employee
    - class Employee:
      - String: FirstName
      - String: LastName
      - Date: BirthDate
        - » Integer: Year
        - » Integer: Month
        - » Integer: Day
      - Integer: ID
      - Date: HireDate
        - » Integer: Year
        - » Integer: Month
        - » Integer: Day
      - Floating point: Salary
  - "Has a" relationship



Object-Oriented Programming

- Example: **`employee.py`** (part 1/3)

```
# employee.py: example for class composition
#
# author: Rainer Doemer
# 03/04/04 RD    initial version

# class definitions

class Date:
    """class for representation of a date"""
    def __init__(self, day, month, year):
        """creates a date object"""
        self.Day = day
        self.Month = month
        self.Year = year

    def __str__(self):
        """returns the date as printable string"""
        return "%d/%d/%d" % (self.Month, self.Day, self.Year)

...
```

Object-Oriented Programming

- Example: **`employee.py`** (part 2/3)

```
...

class Employee:
    """class for representation of an employee"""
    def __init__(self, FirstName, LastName, BirthDate,
                         ID, HireDate, Salary):
        """creates an employee object"""“
        self.FirstName = FirstName
        self.LastName = LastName
        self.BirthDate = BirthDate
        self.ID = ID
        self.HireDate = HireDate
        self.Salary = Salary

    def __str__(self):
        """returns the employee data in printable format"""
        return "%s %s, born %s,\nID %d, hired %s, making $%d" % \
                (self.FirstName,self.LastName,str(self.BirthDate),
                        self.ID,str(self.HireDate),self.Salary)
...
```

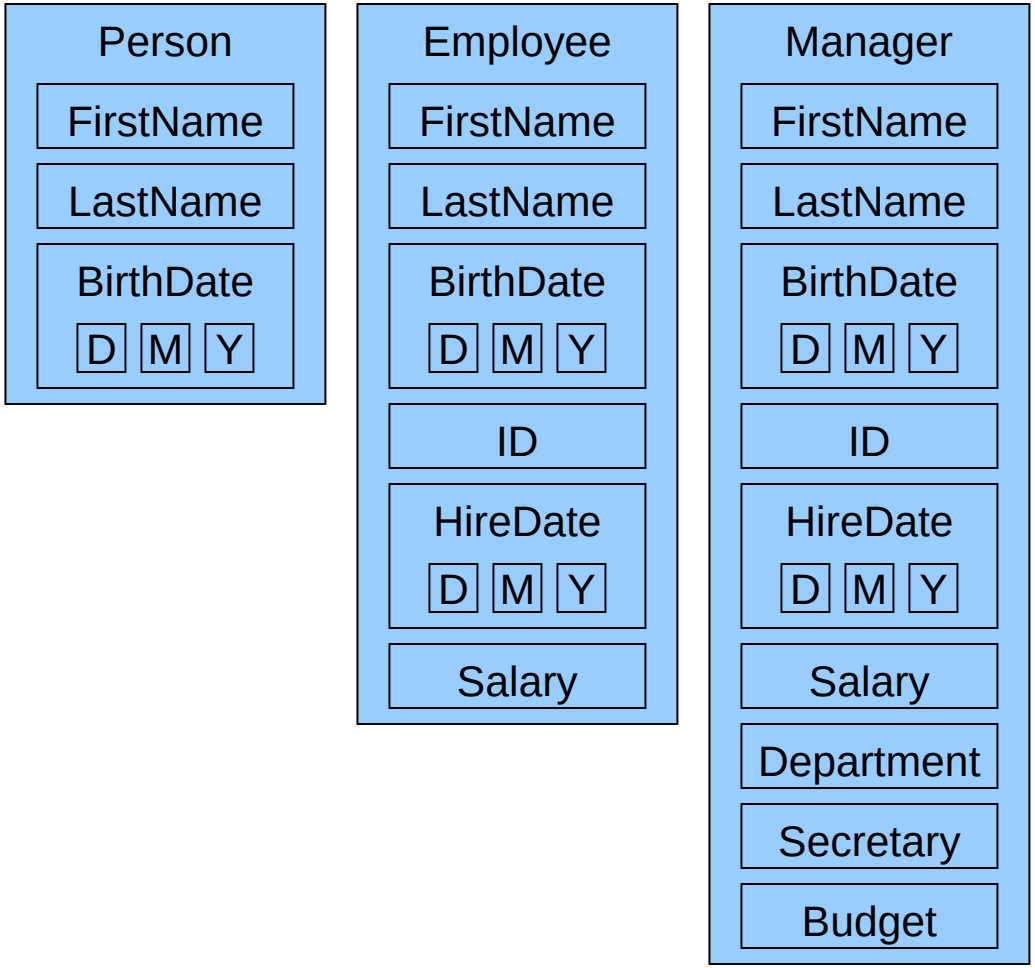
Object-Oriented Programming

- Example: **`employee.py`** (part 3/3)

```
...

# driver program

if __name__ == "__main__":
    e1 = Employee("John", "Smith", Date(23, 2, 1968),
                          1001, Date(1, 1, 2003), 40000)
    print e1
    e2 = Employee("Jeff", "Somebody", Date(31, 12, 1969),
                          1002, Date(1, 2, 2003), 35000)
    print e2
```

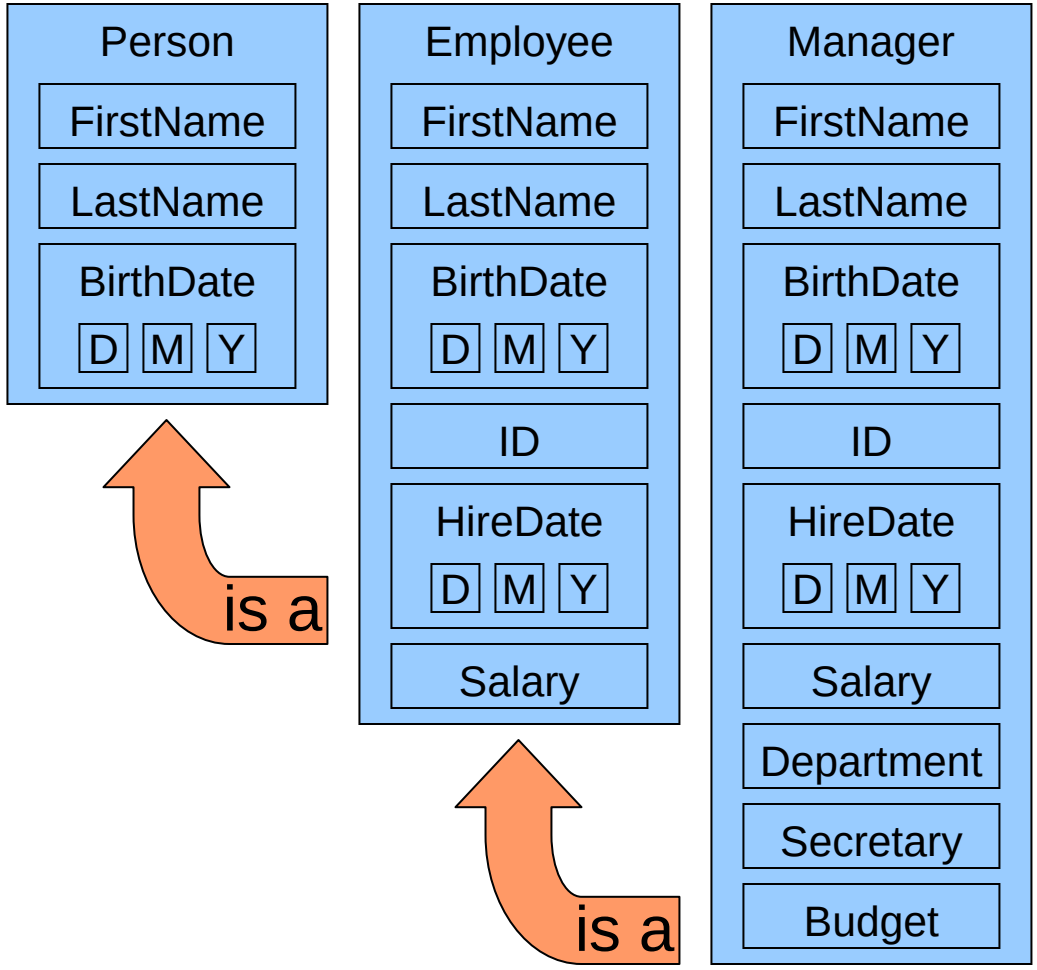
Object-Oriented Programming

- Inheritance
  - Example:
    - Person
    - Employee
    - Manager



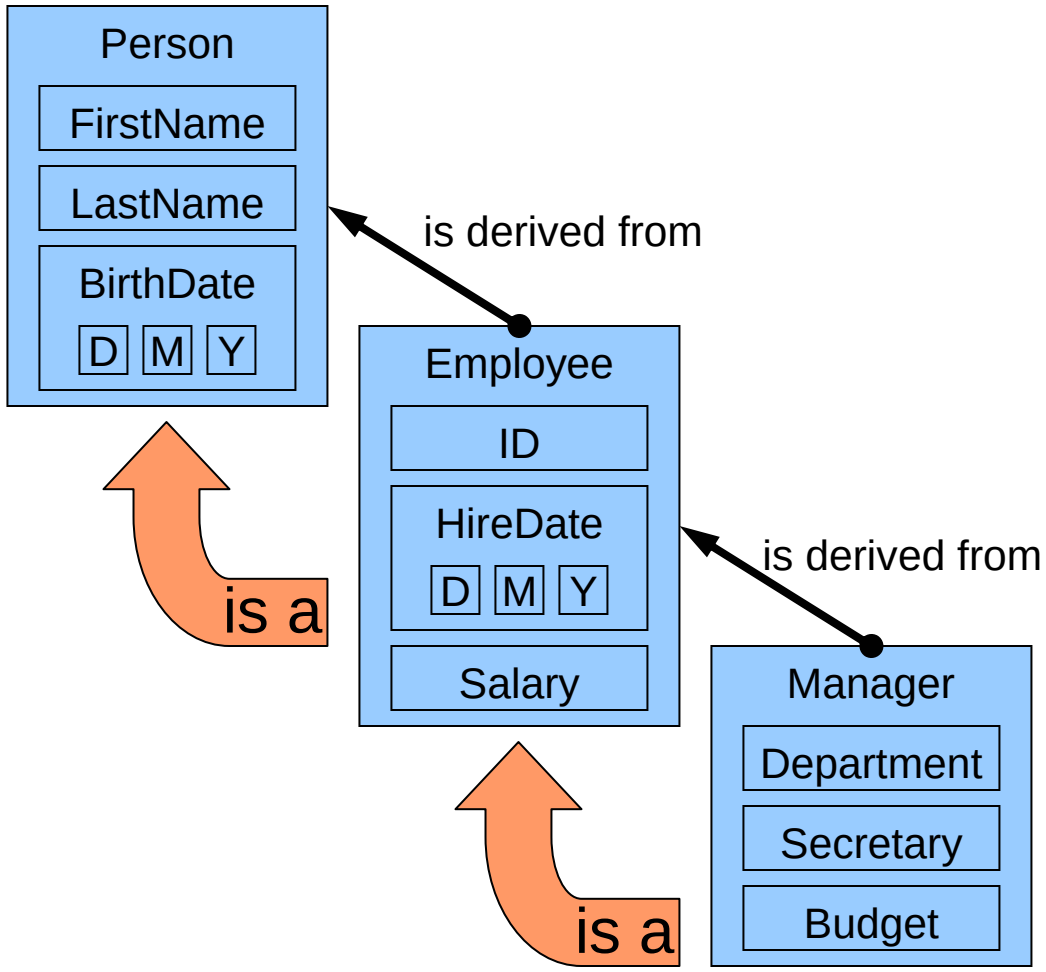
Object-Oriented Programming

- Inheritance
  - Example:
    - Person
    - Employee
    - Manager
  - "Is a" relationship



Object-Oriented Programming

- Inheritance
  - Example:
    - Person
    - Employee
    - Manager
  - "Is a" relationship
  - Base class
    - Person
  - Derived classes
    - Employee
    - Manager



Object-Oriented Programming

- Example: **`manager.py`** (part 1/5)

```
# manager.py: example for class inheritance
# author: Rainer Doemer
#
# modifications:
# 03/04/04 RD    initial version (based on employee.py)

# class definitions

class Date:
    """class for representation of a date"""
    def __init__(self, day, month, year):
        """creates a date object"""
        self.Day = day
        self.Month = month
        self.Year = year

    def __str__(self):
        """returns the date as printable string"""
        return "%d/%d/%d" % (self.Month, self.Day, self.Year)
...
```

Object-Oriented Programming

- Example: **`manager.py`** (part 2/5)

```
...

class Person:
    """class for representation of a person"""

    def __init__(self, FirstName, LastName, BirthDate):
        """creates a person object"""
        self.FirstName = FirstName
         self.LastName = LastName
         self.BirthDate = BirthDate

    def __str__(self):
        """returns the person data in printable format"""
        return "%s %s, born %s" % (self.FirstName, self.LastName,
                                   str(self.BirthDate))
...
```

Object-Oriented Programming

- Example: **`manager.py`** (part 3/5)

```
...

class Employee(Person):
    """class for representation of an employee"""

    def __init__(self, FirstName, LastName, BirthDate,
                       ID, HireDate, Salary):
        """creates an employee object"""
        Person.__init__(self, FirstName, LastName, BirthDate)
        self.ID = ID
        self.HireDate = HireDate
        self.Salary = Salary

    def __str__(self):
        """returns the employee data in printable format"""
        return "%s,\nID %d, hired %s, making $%d" %         \
                (Person.__str__(self), self.ID,
                       str(self.HireDate), self.Salary)

...
```

Object-Oriented Programming

- Example: **`manager.py`** (part 4/5)

```
...

class Manager(Employee):
    """class for representation of a manager"""

    def __init__(self, FirstName, LastName, BirthDate,
                       ID, HireDate, Salary,
                       Dept, Secretary, Budget):
        """creates a manager object"""
        Employee.__init__(self, FirstName, LastName, BirthDate,
                                ID, HireDate, Salary)
        self.Dept = Dept
        self.Secretary = Secretary
        self.Budget = Budget

    def __str__(self):
        """returns the manager data in printable format"""
        return "%s,\nDept. %s, Secretary %s, Budget $%d" %  \
                       (Employee.__str__(self), self.Dept,
                               self.Secretary, self.Budget)
...
```

Object-Oriented Programming

- Example: **`manager.py`** (part 5/5)

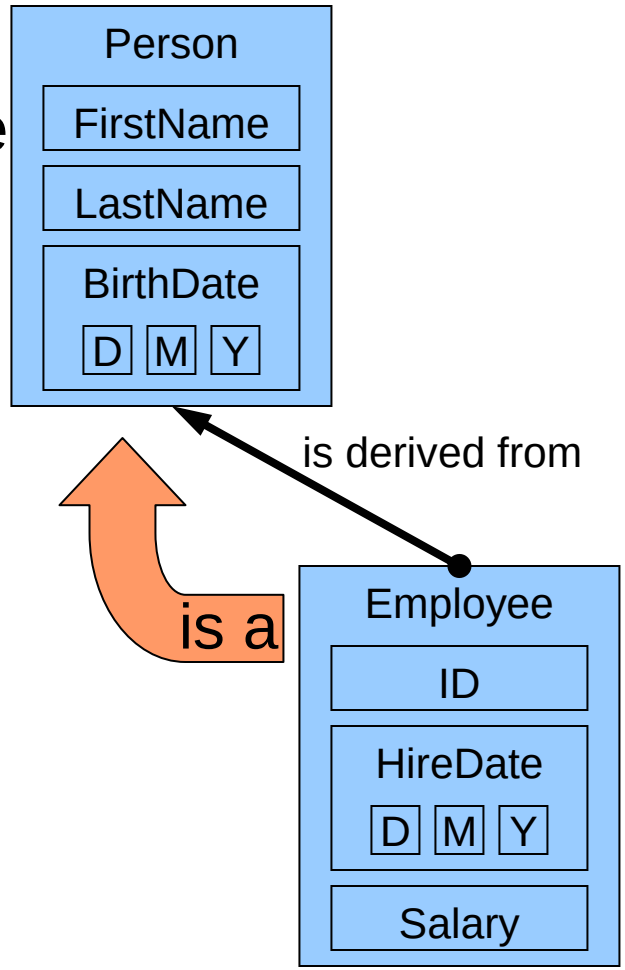
```
...

# driver program

if __name__ == "__main__":
    e1 = Employee("John", "Smith", Date(23, 2, 1968),
                  1001, Date(1, 1, 2003), 40000)
    e2 = Employee("Jeff", "Somebody", Date(31, 12, 1969),
                  1002, Date(1, 2, 2003), 35000)
    p1 = Person("Jane", "Doe", Date(30, 6, 1971))
    p2 = Person("Joe", "Average", Date(30, 6, 1970))
    m1 = Manager("Jenny", "X", Date(15, 1, 1959),
                  1003, Date(1, 1, 2000), 55000,
                  "Imports", "Jeff", 1000000)
    m2 = Manager("Jonathan", "Y", Date(16, 2, 1958),
                  1004, Date(1, 1, 2000), 55000,
                  "Exports", "Jane", 1500000)
    for p in [e1, e2, p1, p2, m1, m2]:
        print p
```

Object-Oriented Programming

- Inheritance
  - Overriding methods
    - Method **`__init__`** of class **`Person`** is overwritten by method **`__init__`** of class **`Employee`**
    - Method **`__str__`** of class **`Person`** is overwritten by method **`__str__`** of class **`Employee`**
    - etc.
  - Polymorphism
    - An object behaves differently depending on its type
      - A person is printed with name and birthday
      - An employee is printed as person, plus ID, hire date, and salary
      - A manager is printed as an employee, plus department, secretary and budget

Object-Oriented Programming

- Inheritance
  - Single inheritance
    - Employee is a Person



Object-Oriented Programming

- Inheritance
  - Single inheritance
    - Employee is a Person
  - Multiple inheritance
    - Employee is a Person and a Spouse