دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

وراثت و چندریختی — ۲

رابطه وراثت

برنامه‌سازی پیشرفته — رامتین خسروی

بهار ۹۹

inheritance

وراثت

Image taken from "Object-Oriented Analysis and Design", 3rd ed., by Grady Booch et al.

سیستم اطلاعاتی دانشگاه

Student
name nin student_id
get_name() get_nin() get_id()

Employee
name nin emp_code
get_name() get_nin() get_code() calc_salary()

سیستم اطلاعاتی دانشگاه

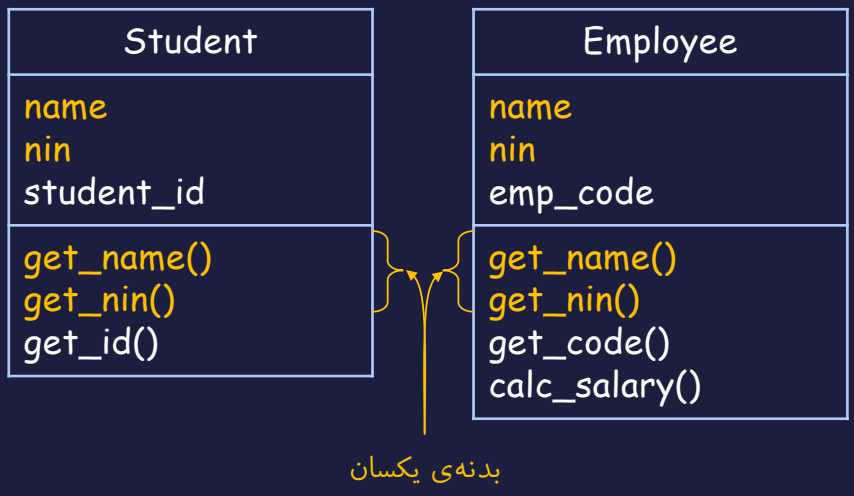
اطلاعات مشترک بین دو کلاس

Student
name nin student_id
get_name() get_nin() get_id()

Employee
name nin emp_code
get_name() get_nin() get_code() calc_salary()

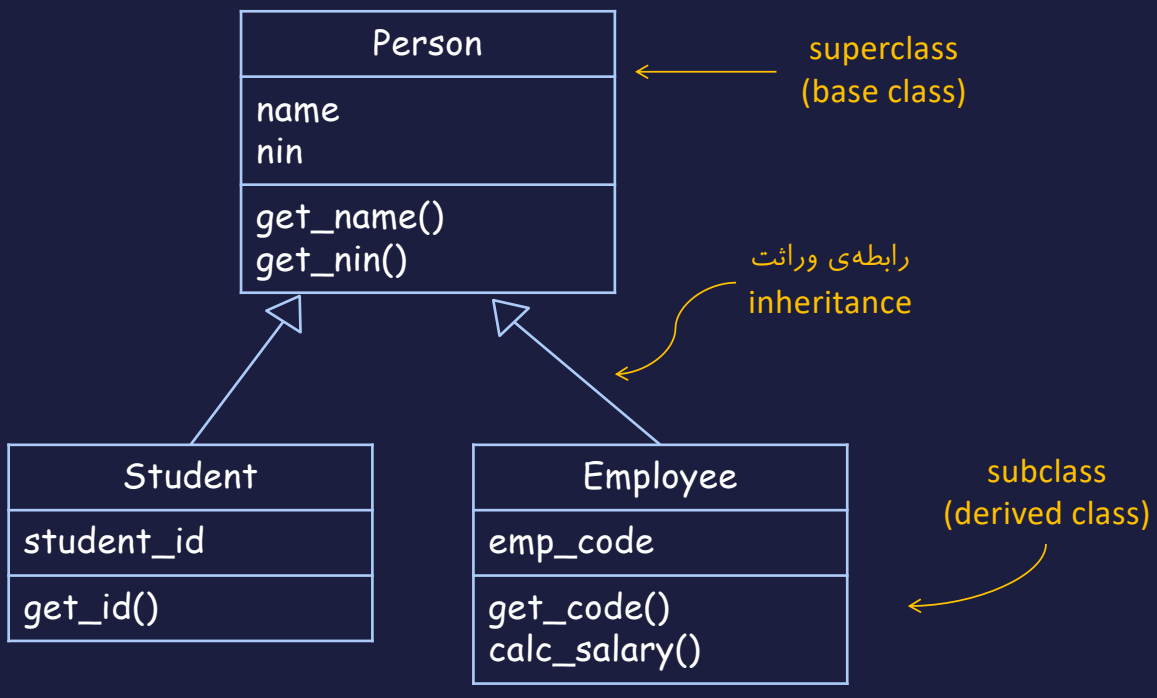
سیستم اطلاعاتی دانشگاه

اطلاعات مشترک بین دو کلاس



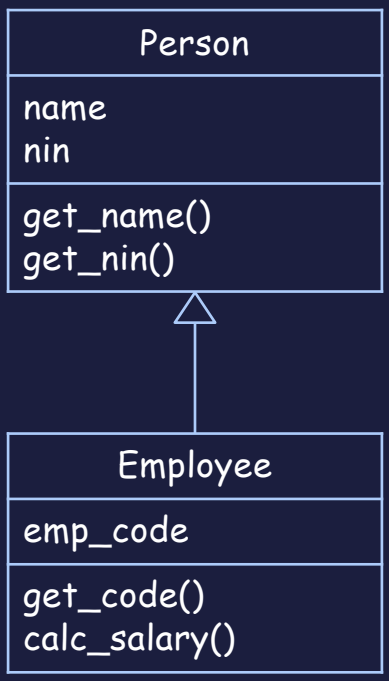
5

فاکتورگیری اطلاعات مشترک

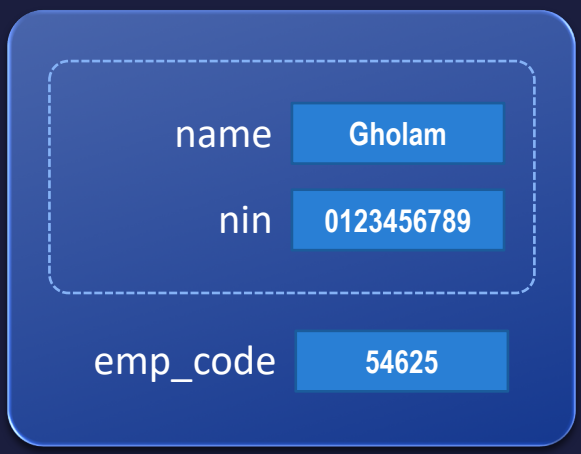


6

اشیاء از نوع زیرکلاس‌ها



یک شیء از نوع Employee



7

متن این مثال را ببینید

ramtung / apnotes

Code | Issues 0 | Pull requests 0 | Actions | Projects 0 | Wiki

Branch: master ▾ apnotes / 11_Inheritance / 01_inheritance.cpp

ramtung fixed compile errors and filenames

1 contributor

54 lines (45 sloc) | 1.05 KB

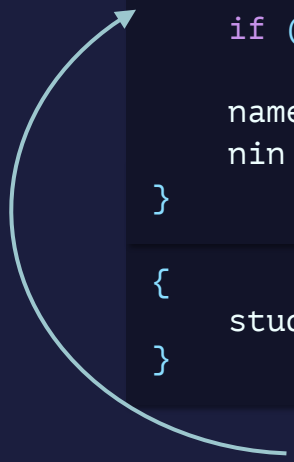
```
1  #include <iostream>
2  #include <string>
3  using namespace std;
4
5  class Person {
6  public:
7          Person(string n, string c)
8                  : name(n), nat_code(c) {}
```

8

```cpp
class Person {
public:
    Person(string name_, string nin_);
    string get_name() { return name; }
    string get_nin() { return nin; }
private:
    string name;
    string nin;
};

class Student : public Person {
public:
    Student(string n, string c, string sid);
    string get_id() { return student_id; }
private:
    string student_id;
};
```

9



```cpp
Person::Person(string name_, string nin_) {
    if (name_ == "" || nin_ == "")
        throw invalid_argument("...");
    name = name_;
    nin = nin_;
}

. Person(n, c)
{
    student_id = sid;
}
```

10

Using <regex> functions for working with Regular Expressions

```
Student::Student(string n, string c, string sid)
        : Person(n, c)
{
    if (regex_match(sid, regex("\\d{8}")))
        throw invalid_argument("...");
    student_id = sid;
}
```

11

```
class Employee : public Person {
public:
    Employee(string name_, string nin_,
             string emp_code_, int base_salary_);

    string get_emp_code() { return emp_code; }
    int calc_salary(int hours_worked);
private:
    string emp_code;
    int base_salary;
};
```

12

```cpp
Employee::Employee(string name_, string nin_,
                   string emp_code_, int base_salary_)
        : Person(name_, nin_)
{
    emp_code = emp_code_;
    base_salary = base_salary_;
}

int Employee::calc_salary(int hours_worked)
{
    int hourly_pay = base_salary / 240;
    return base_salary +
           (hours_worked - 240) * hourly_pay * 1.4;
}
```

13

```cpp
int main()
{
    Student s("gholam", "0123456789", "810188123");
    cout << s.get_name() << endl;

    Employee e("ghamar", "1234567890", "1234", 2000000);
    cout << e.calc_salary(263) << endl;
}
```

14

تمرین عملی

به هدف آشنایی با تعریف زیرکلاس‌ها، یک زیرکلاس از Person به نام Researcher تعریف کنید که علاوه بر نام و شماره ملی دارای یک فیلد از نوع رشته است که نام آزمایشگاهی که در آن کار می‌کند را نگه می‌دارد. برای این کلاس سازنده مناسب تعریف کنید و در تابع main یک نمونه از آن بسازید.

برای کلاس Researcher یک متد تعریف کنید به نام print_info() که نتیجه فراخوانی آن چاپ خروجی مانند زیر است:

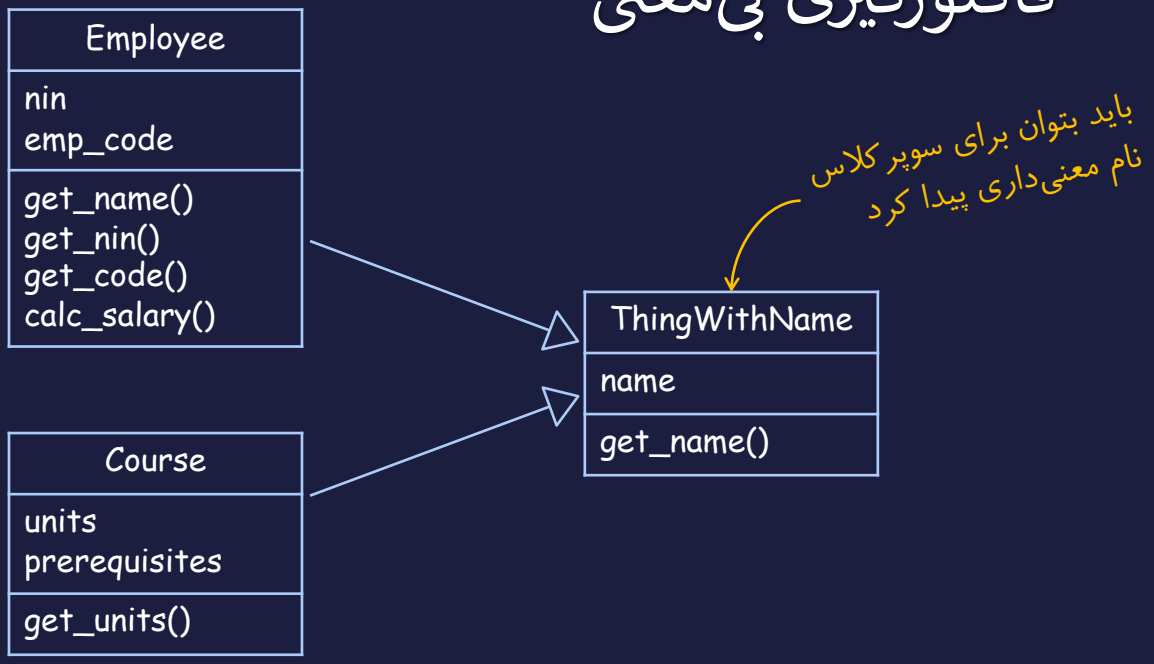
```
Name: Edmond Rockey
NIN: 012345689
Lab: Formal Methods
```


فاکتورگیری بی‌معنی

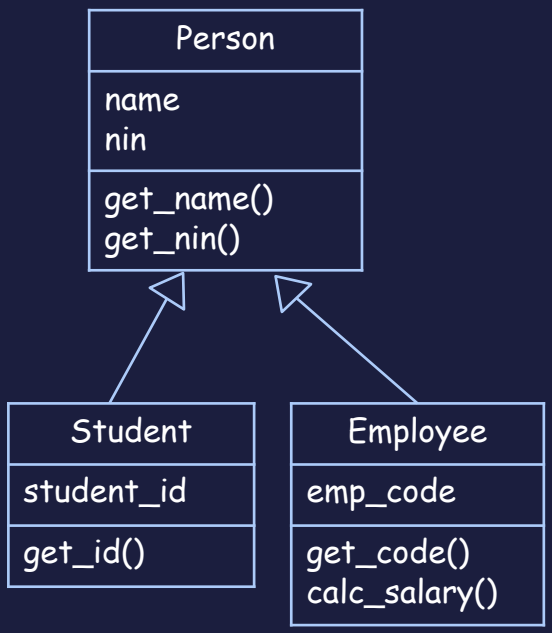
Employee
name nin emp_code
get_name() get_nin() get_code() calc_salary()

Course
name units prerequisites
get_name() get_units()

فاکتورگیری بی‌معنی



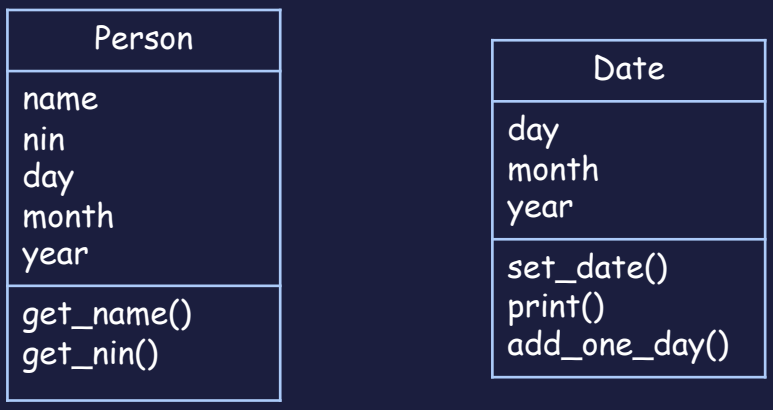
استفاده‌ی مجدد از کد



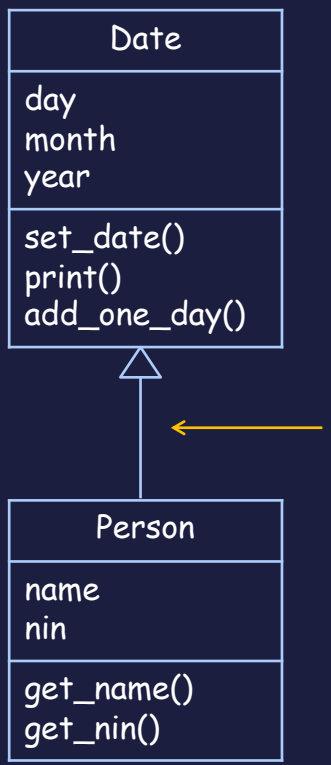
بدنه‌ی متدهای get_name() و get_nin() یک‌بار نوشته می‌شوند و در تمام زیرکلاس‌ها استفاده می‌شوند.

reuse = استفاده‌ی مجدد

تاریخ تولد شخص



استفاده‌ی مجدد نابه‌جا!



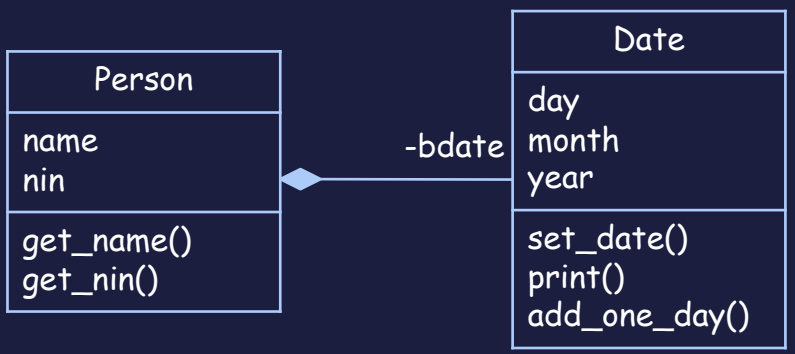
به هدف استفاده مجدد از کد کلاس تاریخ، با این استدلال که هر فرد تاریخ تولد دارد!

```
Person gholam();
gholam.add_one_day();
gholam.print();
```

آزمون IS-A

- پیش از به‌کارگیری رابطه‌ی وراثت آن را با آزمون IS-A محک بزنید:
  - Student IS-A Person
  - Professor IS-A(n) Employee
  - ~~Person IS-A Date~~

استفاده‌ی مجدد با رابطه‌ی HAS-A




```
class Person{
  ...
private:
  Date bdate;
};
```

Person HAS-A Date ✓

فایده‌های وراثت

- جلوگیری از نوشتن کد تکراری
- یک قرارداد مشترک برای گروهی از کلاس‌ها
  (بعداً بیشتر خواهیم دید)

تمرین

کدام‌یک درست است؟