دانشگاه تهران
دانشکده مهندسی برق و کامپیوتر

وراثت و چندریختی — ۳

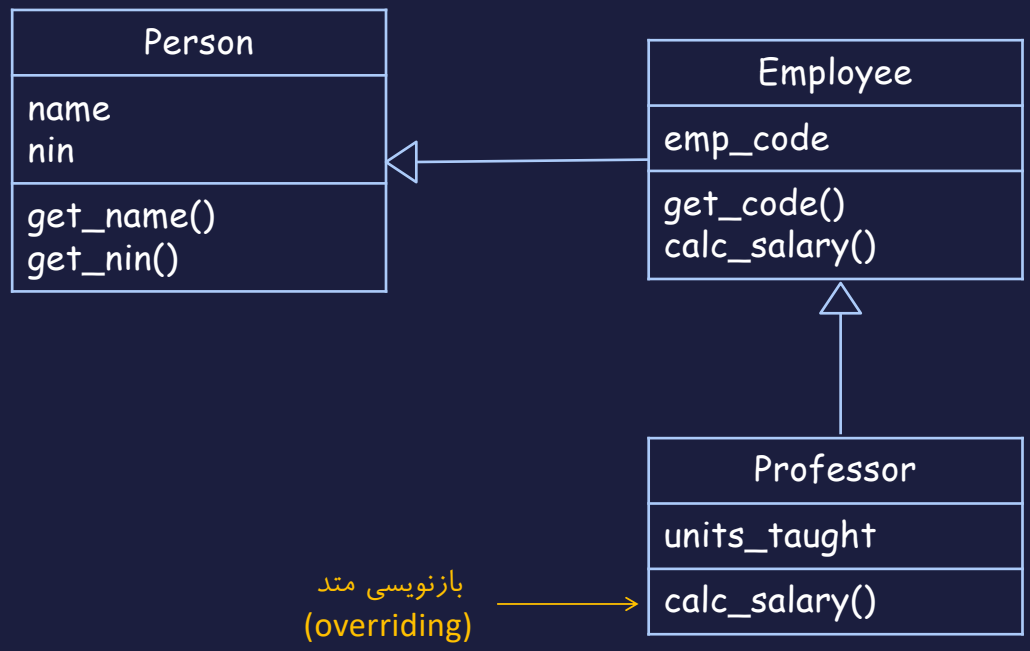
بازنویسی متدها

برنامه‌سازی پیشرفته — رامتین خسروی

بهار ۹۹

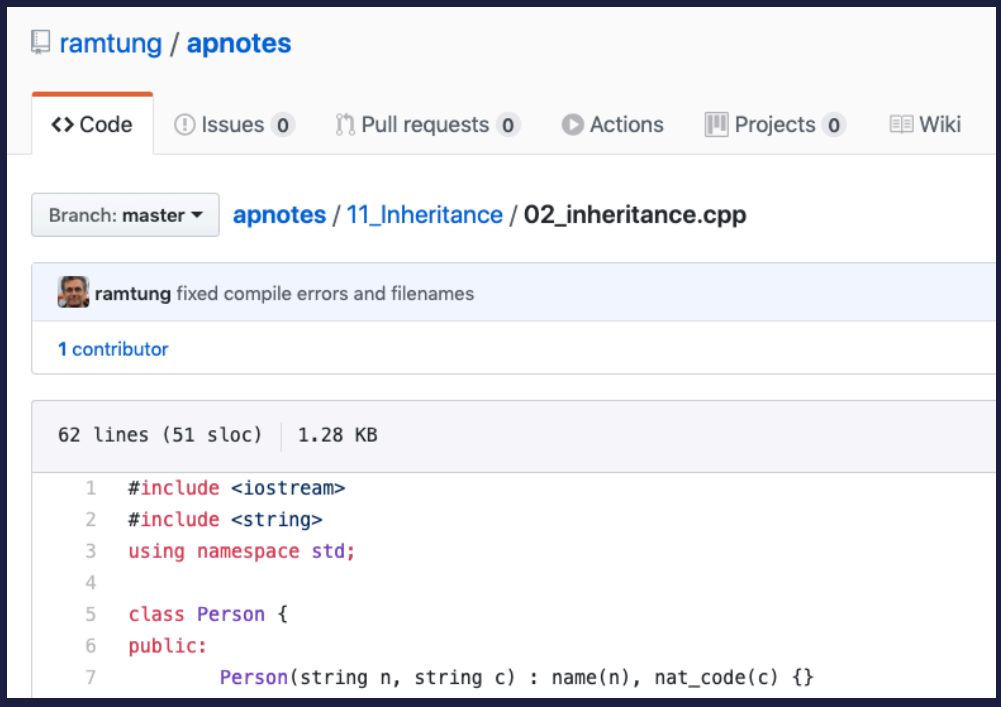
1

اضافه کردن کلاس استاد



2

متن این مثال را ببینید

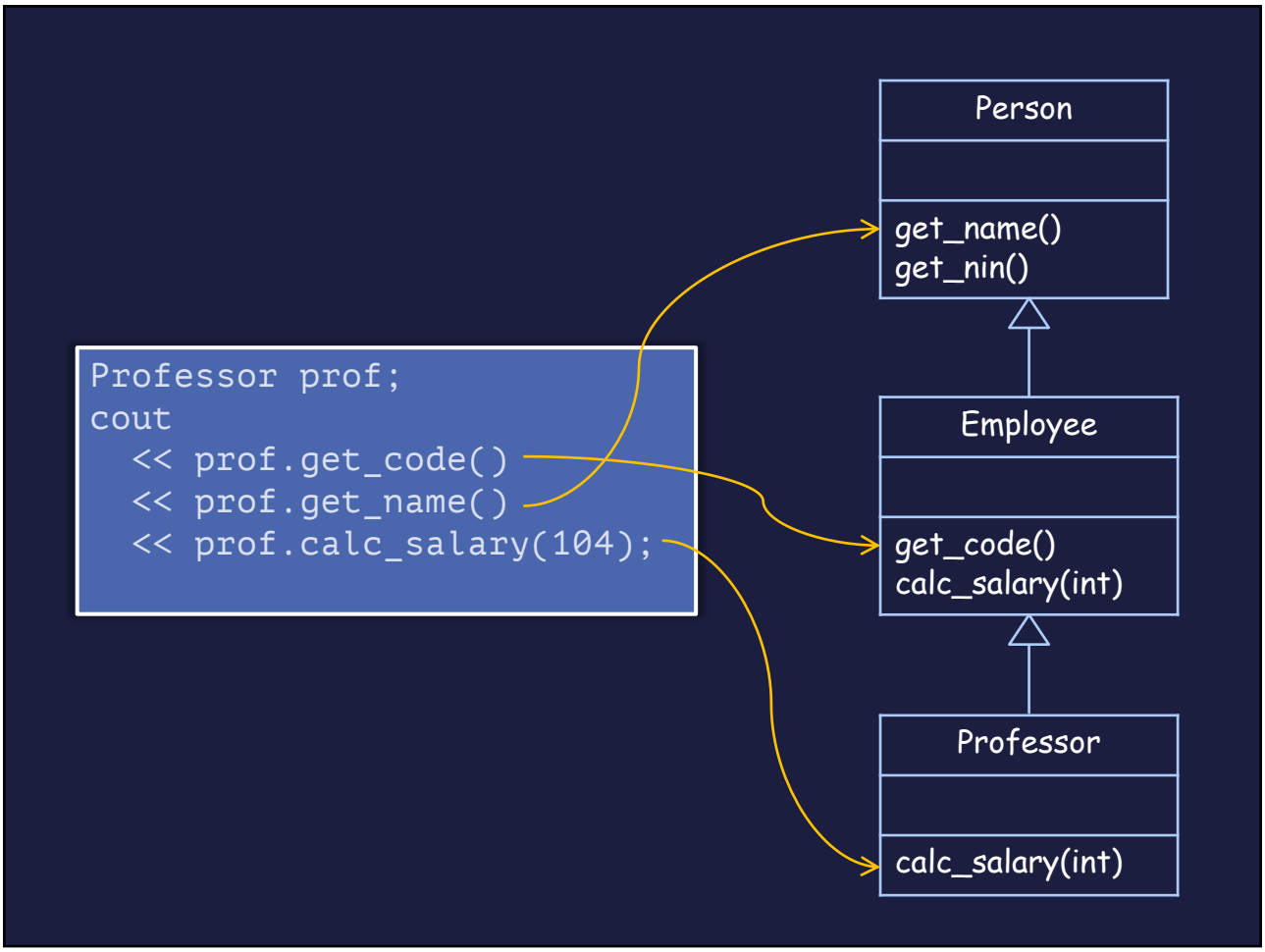



```
class Professor : public Employee {
public:
    Professor(string name_, string nin_,
              string emp_code_, int base_salary_,
              int teaching_units_);
    int calc_salary(int hours_worked);
private:
    int teaching_units;
};
```

```cpp
Professor::Professor(string name_, string nin_,
                     string emp_code_, int base_salary_,
                     int teaching_units_)
        : Employee(name_, nin_, emp_code_, base_salary_)
{
    if (teaching_units < 0)
        throw invalid_argument("...");
    teaching_units = teaching_units_;
}
```

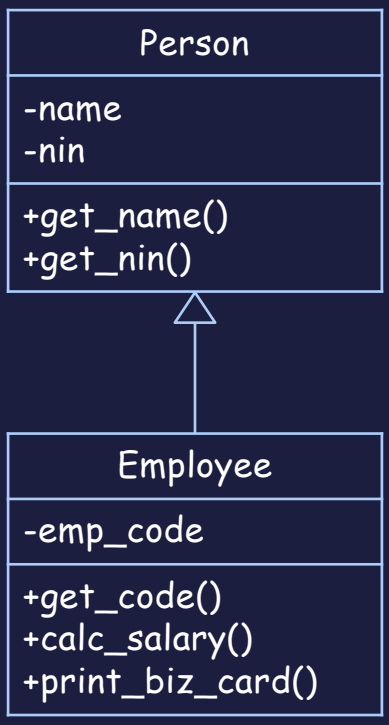
```cpp
int Professor::calc_salary(int hours_worked)  {
    int extra_units = teaching_units - 10;
    return Employee::calc_salary(hours_worked) +
           extra_units * 100000;
}
```

5



6

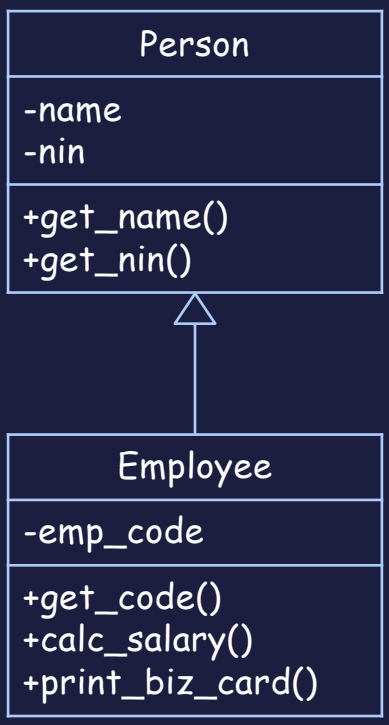
کنترل دسترسی



اعضای خصوصی در کلاس‌های
دیگر قابل دسترسی نیستند،
حتی برای شما زیرکلاس عزیز!

```
void Employee::print_biz_card(){
    // print title
    cout << name << endl;
    cout << emp_code << endl;
    // print contact info.
}
```

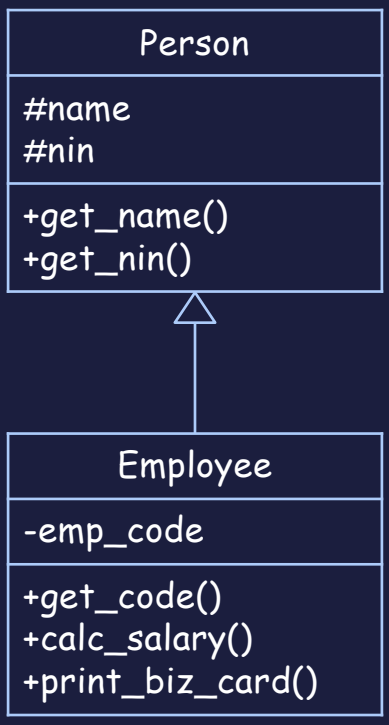

راه‌حل اول



استفاده از واسط عمومی سوپرکلاس

```
void Employee::print_biz_card(){
    // print title
    cout << get_name() << endl;
    cout << emp_code << endl;
    // print contact info.
}
```

دسترسی محافظت‌شده

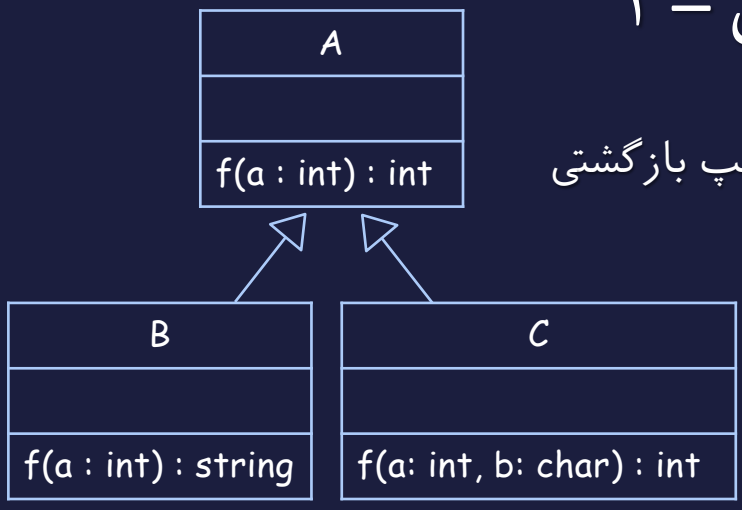



```
class Person {
protected:
  string name;
  string nin;
public:
  // as before
};
```

```
void Employee::print_biz_card(){
    // print title
    cout << name << endl;
    cout << emp_code << endl;
    // print contact info.
}
```


قواعد بازنویسی – ۱

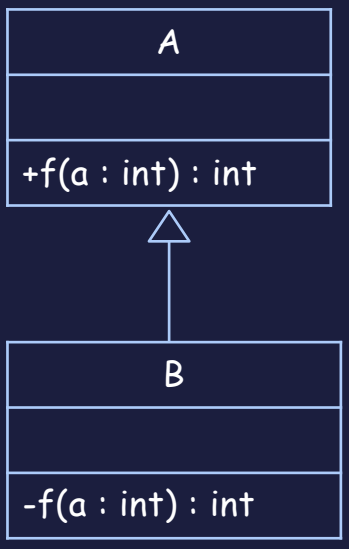
- تایپ پارامترها و تایپ بازگشتی متدها یکسان باشند



Compile error!

This is an overLOAD, not overRIDE!

قواعد بازنویسی – ۲



Compile error!

- سطح دسترسی متد زیرکلاس نباید ضعیف‌تر باشد.

```
B b;
A* ap = &b;
ap->f(2);
```

قرار بوده متد B::f را کسی از بیرون صدا نکند!

این اسلاید در C++ غلط و در جاوا درست است!

رابطه‌ی وراثت

- فاکتورگیری کدهای مشترک
- زیرکلاس اعضای ابرکلاس را به ارث می‌برد
- بازنویسی متدهای به ارث رسیده ممکن است

تمرین

فرض کنید کلاسی به نام Point وجود دارد که ما از نحوه‌ی پیاده سازی آن اطلاعی نداریم و به متن آن نیز دسترسی نداریم، اما از وجود سازنده و متدهای زیر آگاهیم:

`Point(double x, double y)`	سازنده کلاس – مختصات x و y نقطه را دریافت می کند
`void setLoc(double x, double y)`	تعیین مکان نقطه با دادن مختصات
`void setLoc(Point p)`	تعیین مکان نقطه از روی مکان یک نقطه‌ی دیگر
`double getX()`	مختصه‌ی x نقطه
`double getY()`	مختصه‌ی y نقطه

فرض کنید بخواهیم این کلاس را طوری تعمیم دهیم که بتوان با آن در مختصات قطبی نیز کار کرد. به این منظور زیرکلاسی از آن به نام NewPoint تعریف می کنیم که مختصات نقطه را در آن واحد در هر دو مختصات دکارتی و قطبی داشته باشد.

```
class NewPoint : public Point {
private:
    double r;
    double theta;
public:
    NewPoint(double r, double theta) { 1 }
    void setPolarLoc(double r, double theta) { 2 }
    double getR() { return r; }
    double getTheta() { return theta; }
    3
}
```

این کلاس باید طوری تعریف شود که ترکیب دو نوع دسترسی به آن موجب دریافت نتایج نادرست نشود. مثلاً اگر متد setLoc(3,4) از یک شیء را صدا کنیم، فراخوانی getR() مقدار 5 را برگرداند. در تعریف زیر برای این کلاس، بخش‌های 1 تا 3 را بنویسید. بخش 3 شامل متدها یا فیلدهای دیگری است که لازم است برای عملکرد صحیح این کلاس تعریف شوند.