



اعضای ایستای کلاس

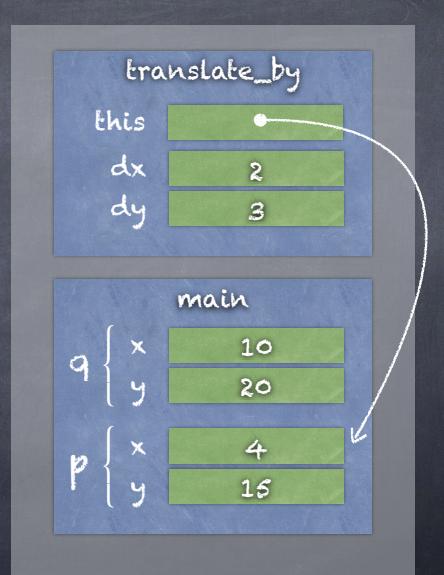
۹۹ بهار

برنامه‌سازی پیشرفته — رامتین خسروی

1

یادآوری: اشاره‌گر this

```
class Point {  
public:  
    Point(double _x, double _y) : x(_x), y(_y) {}  
    double get_x() { return x; }  
    double get_y() { return y; }  
    double len() { return sqrt(x*x+y*y); }  
    void translate_by(double dx, double dy) {  
        x += dx;  
        y += dy;  
    }  
private:  
    double x;  
    double y;  
};  
  
int main()  
{  
    Point p(10, 20);  
    Point q(4, 15);  
    p.translate_by(2, 3);  
}
```



2

```
class Date {  
public:  
    Date(int d, int m, int y);  
    void set_date(int d, int m, int y);  
    void print_date();  
  
    int get_day() { return day; }  
    int get_month() { return month; }  
    int get_year() { return year; }  
private:  
    int day;  
    int month;  
    int year;  
};  
  
bool is_leap_year(int year) {  
    int r = year % 33;  
    return r==1 || r==5 || r==9 || r==13 || ...;  
}
```

3

```
class Date {  
public:  
    Date(int d, int m, int y);  
    void set_date(int d, int m, int y);  
    void print_date();  
  
    int get_day() { return day; }  
    int get_month() { return month; }  
    int get_year() { return year; }  
  
    static bool is_leap_year(int year); ←  
private:  
    int day;  
    int month;  
    int year;  
};  
  
bool Date::is_leap_year(int year) {  
    int r = year % 33;  
    return r==1 || r==5 || r==9 || r==13 || ...;  
}
```

4

```

class Date {
public:
    ...
    static bool is_leap_year(int year);
private:
    int day;
    int month;
    int year;
};

bool Date::is_leap_year(int year) {
    int r = year % 33;
    return r==1 || r==5 || r==9 || r==13 || ...;
}

int main() {
    Date::is_leap_year(1399); ←
    Date bd(31, 6, 1398);
    bd.is_leap_year(1399);
}

```

5

```

class Date {
public:
    ...
    static bool is_leap_year(int year);
private:
    int day;
    int month;
    int year; ←
};

bool Date::is_leap_year() {
    int r = year % 33;
    return r==1 || r==5 || r==9 || r==13 || ...;
}

int main() {
    Date::is_leap_year(1399); ←
    Date bd(31, 6, 1398);
    bd.is_leap_year(1399);
}

```

Compile Error: Invalid use of member 'year'
in static member function

6

فیلد های ایستا

```
class Test {  
private:  
    static int aStaticField;  
public:  
    static void aStaticMethod() {  
        aStaticField++;  
    }  
};  
  
int Test::aStaticField = 0;
```

یک متغیر به ازای کلاس - قابل دسترسی توسط تمام اشیاء

7

مثال فیلد های ایستا

```
class Employee {  
public:  
    Employee(string name_, string p_code_, double bs);  
    double calc_salary(double hours_worked) const;  
    string to_string() const;  
  
private:  
    string name;  
    string p_code;  
    double base_salary;  
    int uid;  
  
    static int nextId;  
    static int getNextId() {  
        return nextId++;  
    }  
};
```

8

```

class Employee {
public:
    Employee(string name_, string p_code_, double bs);
    ...
private:
    string name;
    string p_code;
    double base_salary;
    int uid;

    static int nextId;
    static int getNextId() { return nextId++; }
};

int Employee::nextId = 0;

Employee::Employee(string name_, string p_code_, double bs)
{
    name = name_;
    p_code = p_code_;
    base_salary = bs;
    uid = getNextId();
}

```

9

تمرین کوتاه

- کلاسی به نام ResourceController بنویسید که اجازه ندهد بیش از ۳ شیء از کلاس ساخته شود.
- سؤال: برای چهارمین نمونه چه کنیم؟

10

کلاس‌هایی فقط با اعضای ایستا

```
class EmployeeRepository {  
private:  
    static DBConnection con;  
    // ...  
public:  
    static void save_employee(Employee* e) {  
        // save employee in the database  
    }  
  
    static Employee* find_by_id(int emp_id) {  
        // load and return the employee with ID emp_id  
    }  
    //...  
};
```

11

کلاس‌هایی فقط با اعضای ایستا

```
class EmployeeRepository {  
private:  
    static DBConnection con;  
    // ...  
public:  
    static void save_employee(Employee* e) {  
        // save employee in the database  
    }  
  
    static Employee* find_by_id(int emp_id) {  
        // load and return the employee with ID emp_id  
    }  
    //...  
};  
auto e = EmployeeRepository::find_by_id(120482);
```

12

```
class EmployeeRepository {  
private:  
  
    DBConnection con;  
    // ...  
public:  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
};
```

13

```
class EmployeeRepository {  
private:  
    static EmployeeRepository *instance;  
  
    DBConnection con;  
    // ...  
public:  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
};
```

14

```
class EmployeeRepository {  
private:  
    static EmployeeRepository *instance;  
  
    DBConnection con;  
    // ...  
public:  
    static EmployeeRepository* get_instance();  
  
  
  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
};
```

15

```
class EmployeeRepository {  
private:  
    static EmployeeRepository *instance;  
  
    DBConnection con;  
    // ...  
public:  
    static EmployeeRepository* get_instance() {  
        if (!instance)  
            instance = new EmployeeRepository();  
        return instance;  
    }  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
};
```

16

```
class EmployeeRepository {  
private:  
    static EmployeeRepository *instance;  
    EmployeeRepository() { ... }  
  
    DBConnection con;  
    // ...  
public:  
    static EmployeeRepository* get_instance()  
    {  
        if (!instance)  
            instance = new EmployeeRepository();  
        return instance;  
    }  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
};
```

17

```
class EmployeeRepository {  
private:  
    static EmployeeRepository *instance;  
    EmployeeRepository() { ... }  
  
    DBConnection con;  
    // ...  
public:  
    static EmployeeRepository* get_instance()  
    {  
        if (!instance)  
            instance = new EmployeeRepository();  
        return instance;  
    }  
  
    void save_employee(Employee* e) { ... }  
    Employee* find_by_id(int emp_id) { ... }  
    // ...  
}; auto e = EmployeeRepository::get_instance().find_by_id("120482");
```

18

الگوی طراحی Singleton

- یک الگوی طراحی که تعداد اشیاء یک کلاس را به یک محدود می‌کند.
- یکی از معروف‌ترین الگوهای طراحی
- مورد انتقادهای زیاد
- Singleton Considered Stupid

19

خواناتر از سازنده

سازنده زیر شیء d را به چه تاریخی مقداردهی اولیه می‌کند؟

```
Date d();
```

20

خواناتر از سازنده

سازنده زیر شیء `d` را به چه تاریخی مقداردهی اولیه می‌کند؟

```
Date d();
```

با نمونه‌های زیر مقایسه کنید:

```
Date d = Date::today();  
Date d = Date::unix_epoch(); // 01-01-1970  
Date d = Date::parse_date("01/01/1399");
```