面向对象：使用工具（即类）的人可以不用了解“工具是如何创建的（即类的实现）”。
面向对象的特点：代码重用（面向过程也有）、实现隐藏、继承、多态性。

1、类与对象

类：既能描述数据的属性（数据成员），也能描述对属性的操作（成员函数）。

“类”和“对象”是两个不同的概念。例如 Rational r1; 中的 Rational 是类，r1 是对象。

定义一个类主要是声明了它的数据成员（也称成员变量）和成员函数。
在类的定义中对数据成员赋初值，称为类内初始化，在创建该类的对象时执行。
如果一个数据成员在类内和构造函数的初始化列表中都进行了初始化，则以后者为准。
静态数据成员一般不可以在类内初始化（因为静态成员的初始化不能依赖于对象的创建），但整型、字符型和布尔型的静态常量数据成员可以在类内初始化。

私有成员(private)：只能由该类的成员函数访问。
【该私有成员所属对象的成员函数，或该类其它对象的成员函数，都可以访问该私有成员】
保护成员(protected)：只能由该类及其派生类的成员函数访问。

类指针不是对象，例如：Rational *p; 没有定义对象，所以也不调用构造函数。
创建动态对象的表达式是 new 类名(参数)。如 new Rational(1, 2) 这个表达式定义了一个动态对象，并提供该对象在内存中的地址。这个地址可以赋值给一个 Rational 类的指针。

2、访问对象

当一个类指针 p 储存了一个对象 x 的地址，可以用 p->来访问 x 的成员 k。
【虽然写作 p->k，但 p 指向的是 x，而不是 k】

用 cout << 不能直接输出自定义的类的对象。用 cin >> 不能直接输入自定义的类的对象。
同类型的对象之间可以相互赋值，即复制所有非静态的数据成员。

非静态的成员函数中有隐藏的 this 指针，指向调用该成员函数的当前对象（即*this）。
例如当前对象的类型为 IntArray，那么 this 指针的类型就是 IntArray *。
静态成员函数中没有 this 指针，所以即使通过某个对象来调用静态成员函数，也无法在函数中访问当前对象和它的非静态成员。

3、构造与析构

构造函数：初始化该对象的数据成员。在创建对象时自动调用。
析构函数：在对象消亡时自动调用。

构造函数的函数名与类名相同，没有返回类型（不是 void 而是没有），可以有各种参数。
析构函数的函数名是类名前加波浪线~，也没有返回类型，且参数表为空。

参数表为空的构造函数统称为缺省（或默认）的构造函数，不一定是系统自动生成的。
缺省的构造函数也可以将数据成员初始化为特定值（比如用常量赋初值）。
只有当程序员没有为一个类定义构造函数时，编译系统才会自动生成一个缺省的构造函数，这个缺省的构造函数会将所有非静态数据成员初始化为随机值。

构造函数初始化列表位于函数头和函数体之间。它以一个冒号开头，接着是一个以逗号分隔的数据成员列表（结尾处没有任何符号）。
对象成员和常量数据成员必须在初始化列表中进行初始化。
静态数据成员不在构造函数中初始化（因为它不属于对象）。

如果程序员定义类时没有定义析构函数，编译系统会自动生成一个缺省的空析构函数。

拷贝构造函数的原型：类名 (const <类名> &);
定义一个新对象并用已经存在的同类对象给它初始化时，会调用拷贝构造函数。
“定义一个新对象”包括定义一个局部对象、动态对象、作为形参的对象（值传递）和函数返回时创建的临时对象等等。
如果程序员没有定义拷贝构造函数，系统会定义一个缺省的拷贝构造函数。该函数将已存在的同类对象原式原样地复制给新对象。
当数据成员中包含指针，且指针指向动态变量或动态数组时，很可能需要程序员自定义拷贝构造函数，以复制该动态变量或数组（而不是两个对象的成员指针指向相同的空间）。

4、常量对象、常量数据成员与静态数据成员

常量对象一定要通过程序员自定义的构造函数来初始化。
常量对象**不能调用非常量成员函数**。
表示常量成员函数的 const 写在函数头的**后面**。
常量成员函数不能修改数据成员，也**不能调用非常量成员函数**。
【但常量成员函数可以修改非当前对象的同类对象的数据成员】
定义一个非常量的引用时，不能用常量对象给它初始化。

```
const Rational a(1, 2);
Rational &b = a; // 编译报错
函数原型：void fun (Rational &b);
函数调用：fun (a); // 编译报错
```

常量数据成员必须在构造函数的初始化列表中初始化，不能在构造函数的函数体中赋值。

静态成员要在类的定义中声明，声明时在前面加上 static。
静态成员函数在类的定义之外定义时，函数头前面不加 static。
静态数据成员必须在类的定义之外写定义语句（同时可以初始化），前面不加 static。
【在类的定义之外定义成员时，别忘了加上 **类名::** 】

静态数据成员不属于对象，所以 sizeof(对象)不计入静态数据成员的大小。
【但可以通过对象来访问该类的静态成员】

静态成员函数没有 this 指针，所以不能访问非静态数据成员，不能调用非静态成员函数。

5、友元

友元（friend）关系是授予的而不是索取的，既不对称，也不可传递。
如果在类的定义中声明或定义了一个函数且在函数头前有 friend，那么这个函数不是这个类的成员函数，而是这个类的友元函数（没有 this 指针）。
类 A 的成员函数作为类 B 的友元函数时，必须先**声明**类 B，再**定义**类 A，再**定义**类 B。