2.1 对象交互

Clock 案例

class Display {

private int value = 0;

private int limit = 0;

public Display(int limit) {

this.limit = limit;

}

public boolean increase() {

boolean flopped = false;

value++;

if (value == limit) {

value = 0;

flopped = true;

}

return flopped;

}

public int getValue() {

return value;

}

}

class Clock {

private Display hour = null;

private Display minute = null;

private Display second = null;

public Clock() {

hour = new Display(24);

minute = new Display(60);

second = new Display(60);

}

public void start() {

boolean increaseMinute = second.increase();

if (increaseMinute) {

boolean increaseHour = minute.increase();

if (increaseHour) {

hour.increase();

}

}

}

2.2 访问属性

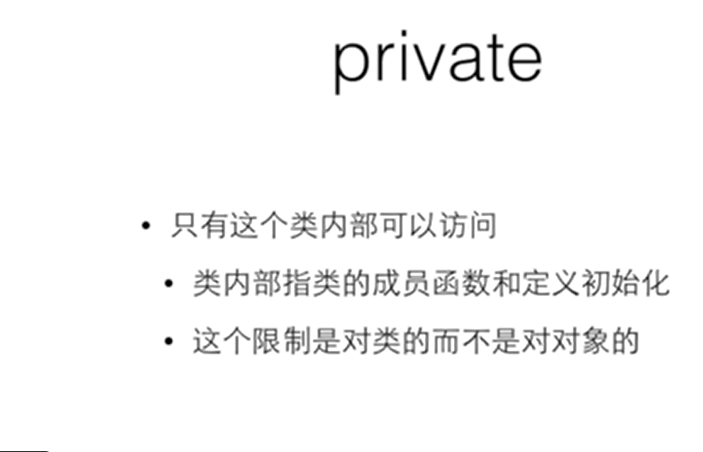
封装，就是把数据和对这些数据的操作放在一起，并且用这些操作把数据掩盖起来，是面向对象的基本概念之一，也是最核心的概念。

我们有一个非常直截了当的手段来保证在类的设计的时候做到封装：

1. 所有的成员变量必须是private的，这样就避免别人任意使用你的内部数据；
2. 所有public的函数，只是用来实现这个类的对象或类自己要提供的服务的，而不是用来直接访问数据的。除非对数据的访问就是这个类及对象的服务。简单地说，给每个成员变量提供一对用于读写的get/set函数也是不合适的设计。

成员变量应设置为private;

Private的成员变量和成员函数可在类内部访问（代码层面，而不是运行层面，参见第一周fraction编程题实现multiply和plus功能）



2.3 包

不加public称为 friendly

只能在同一个package中访问

2.4 类变量

类是描述，对象是实体。在类里所描述的成员变量，是位于这个类的每一个对象中的。

而如果某个成员有static关键字做修饰，它就不再属于每一个对象，而是属于整个类的了。

通过每个对象都可以访问到这些类变量和类函数，但是也可以通过类的名字来访问它们。类函数由于不属于任何对象，因此也没有办法建立与调用它们的对象的关系，就不能访问任何非static的成员变量和成员函数了。

Static 变量/函数，即为类变量/函数，对它的操作会影响到所有对象，即它是属于类而不是属于对象。

另：static函数只能对static变量进行操作，不能对非static的成员函数和变量操作