# 复用类

## 一、两种代码复用的方法

### 1、组合

在一个类中初始化另一个类，new方法.

### 2、继承

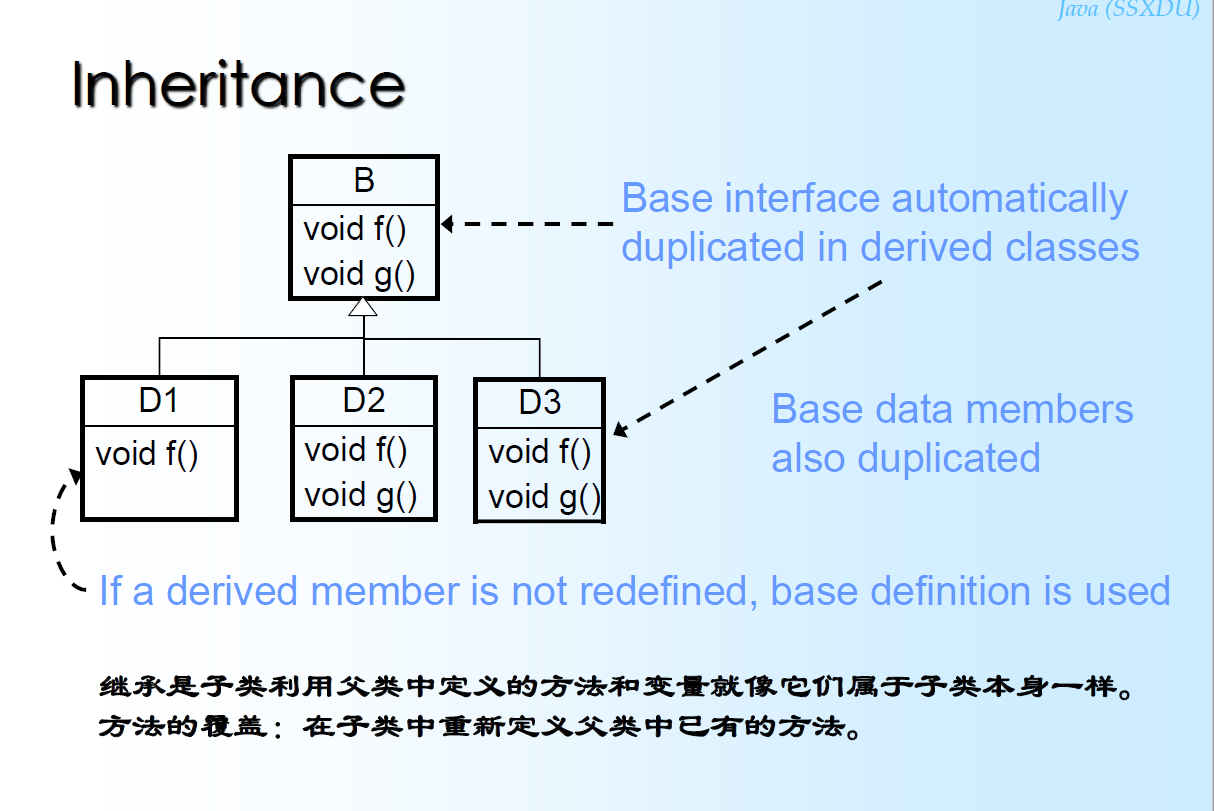
public class 导出类的名字 extends 基类的名字{}.

导出类的包含了所有的基类的方法与属性。

在导出类被实例化时，最先初始化的是基类的构造器，然后才是导出类的构造器。

使用super关键字调用基类版本的域或方法。

Java会调用的是基类的默认构造器，如果基类没有，需要用super显式地调用基类的带参构造器。

调用基类构造器必须是你在导出类构造器中要做的第一件事。

继承时，若是没有对基类的方法进行覆盖，就说明用的是原方法。

所有类都是从Object中继承出来的，如果缺省extends子句，则该类为java.lang.Object的子类。

## 二、toString

toString源于object类。每个类都有toString方法，默认输出是确定的，但是一般都会重写toString方法后覆盖默认输出。

例子：

class person{}

class sunhui{

person p = new person();

System.out.println(p.toString());

System.out.println(p);//这两种调用toString的方法都对。前者是显式调用，后者是隐式调用  
}

## 3、类初始化时代码段的初始化顺序

我们在初始化一个类时总是会有这样的疑问：我究竟是先初始化的什么，这样的疑问固然可以通过一个简单的程序解决，但是总结下来记住也会节省不少的时间。

根据以往博客的介绍，前人总结的是每当一个类初始化时，可以尝试运行program中给的[Explore.java](program/Explore.java)。

运行结果如下：

1. Fsx无参构造器初始化了
2. 静态代码块被执行了
3. Fsx有参构造器初始化了
4. 普通代码块执行了
5. 构造器执行了
6. Fsx有参构造器初始化了
7. 普通代码块执行了
8. 构造器执行了

所以，当类被第一次因实例化而加载时，初始化顺序是：静态属性，静态代码块，普通属性，普通代码块，构造器。当再次加载该类时，由于静态属性和静态代码段是所有类共有的，所以不再初始化，从第三步开始。

这里还有一个例外情况，当初始化的是有父类extends出来的类时，首先实例化基类，再实例化导出类。