Java中方法重载和重写

注意：重载（Overloading）和重写（覆盖，Override）都是指的是**方法的重载和重写**，特别是当说覆盖时，指的也只是方法，成员变量没有覆盖这么一说，因为子类和父类中有重名的变量时，子类在实例化时会把这两个变量都进行封装，用特定的标号加以区分。

1. **方法重载**：
2. 方法重载（overload）：概念：在同一个类中，允许存在两个或者多个同名方法，但只要参数个数或者参数类型不同即可。注意：返回值对于方法重载没有任何影响，也就是说，如果参数个数和类型都相同时，只是返回类型不同，这时候算是错误。
3. 重载特点：方法重载只看**参数个数和类型**，与返回值类型无关。方法返回类型不同，不能区别出不同的方法。无法以返回类型作为重载函数的区分标准。
4. **构造方法重载**：与方法重载一样的概念，因为构造方法也是一个方法，所以构造方法重载与方法重载一样。正好构造方法没有返回参数。this是一个关键字，可以实现在一个构造方法中调用另外一个构造方法，this（。。。。。）小括号中表示的就是所要调用的构造方法的参数排列。this（）必须放在构造方法的第一行，即this（）前面不能有任何可以执行的代码。
5. **方法重载的作用**：提高程序的可读性。由于方法名字就代表了此方法的功能，但是由于很多方法完成近似的功能，所以需要方法名字相同。
6. **方法重载Overloading**是一个**类中多态性**的一种表现。调用方法时通过传递给它们的不同参数个数和参数类型来决定具体使用哪个方法, 这就是多态性。
7. **方法重写**
8. 重写注意事项：

（1） **父类与子类之间的多态性，对父类的函数进行重新定义**。如果在子类中定义某方法与其父类有相同的名称和参数，我们说该方法被重写 (Overriding)。在[Java](http://java.chinaitlab.com/)中，子类可继承父类中的方法，而不需要重新编写相同的方法。

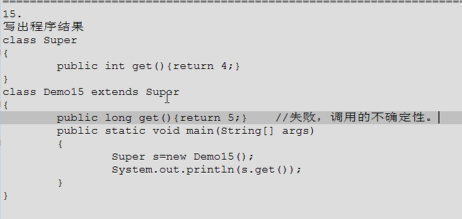
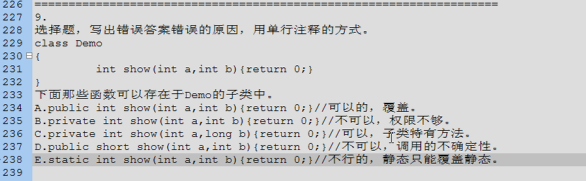
但有时子类并不想原封不动地继承父类的方法，而是想作一定的修改，这就需要采用方法的重写。

方法重写又称方法覆盖。  
    （2）若子类中的方法与父类中的某一方法具有相同的方法名、返回类型和参数表，则新方法将覆盖原有的方法。

如需父类中原有的方法，可使用super关键字，该关键字引用了当前类的父类。  
 （3）子**类函数的访问修饰权限不能少于父类的**；

1. **子类方法覆盖父类方法时，子类权限必须要大于等于父类的权限。**

但是当父类中的方法权限为private的，则此时不允许重写，因为父类的private方法，子类中本来就不可以使用，对子类来说相当于把此方法隐藏掉了。重写真正意义是：子类和父类中如果有**声明相同**的方法，则执行子类中的方法，但是如果子类中没有此方法，父类中的方法也必须能够调用才可以。上述那种情况，子类把此方法删去，父类中的此声明相同的方法不能使用。

1. **静态的方法不能重写非静态的方法，同样，非静态的方法也不能重写静态的方法。**非静态的方法可以重写非静态的方法，静态的方法可以重写静态的方法。静态只能重写静态的，或被静态的重写。（此时注意静态方法比较特殊，因为它不再是依赖于对象了，而是直接依赖于类）
2. **如果子类重写父类方法时，方法名、参数列表都一样，权限也符合，但是返回类型不同，这是非法的，编译失败，重写失败，要重写，方法声明中除了权限可以大一些，其他的必须都相同，特别注意是返回类型必须相同。**
3. 
4. 什么时候使用重写操作？

当对一个类进行子类拓展时，子类需要保留父类的功能声明，但是要定义子类中该功能的特有内容时，就使用重写操作来完成。

1. **方法重载与重写的区别**：
2. 重载（Overload）：指同一类中允许存在多个同名方法，但是这些方法的参数列表不同（类型或个数不同），依靠此特征判断调用不同的方法。包括构造方法的重载和成员方法的重载。
3. 重写（Override）：又叫**覆写，覆盖**；指的是子类重新定义父类中的方法。方法名和参数列表必须相同，执行体不同。
4. 重写是在两个类中且是父类和子类，重载是在一个类中。
5. 重载只要求是方法名相同和小括号里的参数列表完全相同，但对方法的返回值类型没有要求一致；而重写要求方法的声明完全相同，即方法名、方法返回值类型、参数列表完全相同。
6. 构造方法的作用：为新建的对象赋予初始值。
7. 如果一个类中没有一个构造方法，则编译器会自动生成一个无参数的构造方法，一旦类中存在任何一个或多个构造方法，编译器就不会再自动添加无参数的构造方法。
8. 构造方法的名与类名相同，没有返回值，当然不需要修饰符，在new一个新的对象时调用相应的对象。
9. 方法重载是一个类中多态性的表现，方法重写是父类与子类之间多态性的表现。