1. 直接赋值、浅拷贝和深拷贝的区别

对于基本数据类型，使用=做赋值操作的时候，实际上是拷贝它的值，但是对于对象而言，实际赋值的只是这个对象的引用，将原对象的引用传递过去，他们实际上还是指向的同一个对象。浅拷贝和深拷贝就是在这个基础上做的区分，如果在拷贝这个对象的时候，只对基本数据类型进行了拷贝，而对引用数据类型只是进行了引用的传递，而没有真实的创建一个新的对象，则认为是浅拷贝。反之，在对引用数据类型进行拷贝的时候，创建了一个新的对象，并且复制其内的成员变量，则认为是深拷贝。

直接赋值：其实就是对象的引用（别名）

浅拷贝：拷贝父对象，不会拷贝对象的内部的子对象。浅拷贝仅仅是指向被复制的内存地址，如果原地址中对象被改变了，那么复制出来的对象也会相应改变。

深拷贝：copy模块的deepcopy方法，完全拷贝了父对象及其子对象。在计算机中开辟了一块新的内存地址用于存放复制的对象。

1. java中的引用

先说C++中的引用是同一块内存的不同名字，而Java的引用是指向一个对象，引用本身也占用了内存

1. java的值传递和引用传递

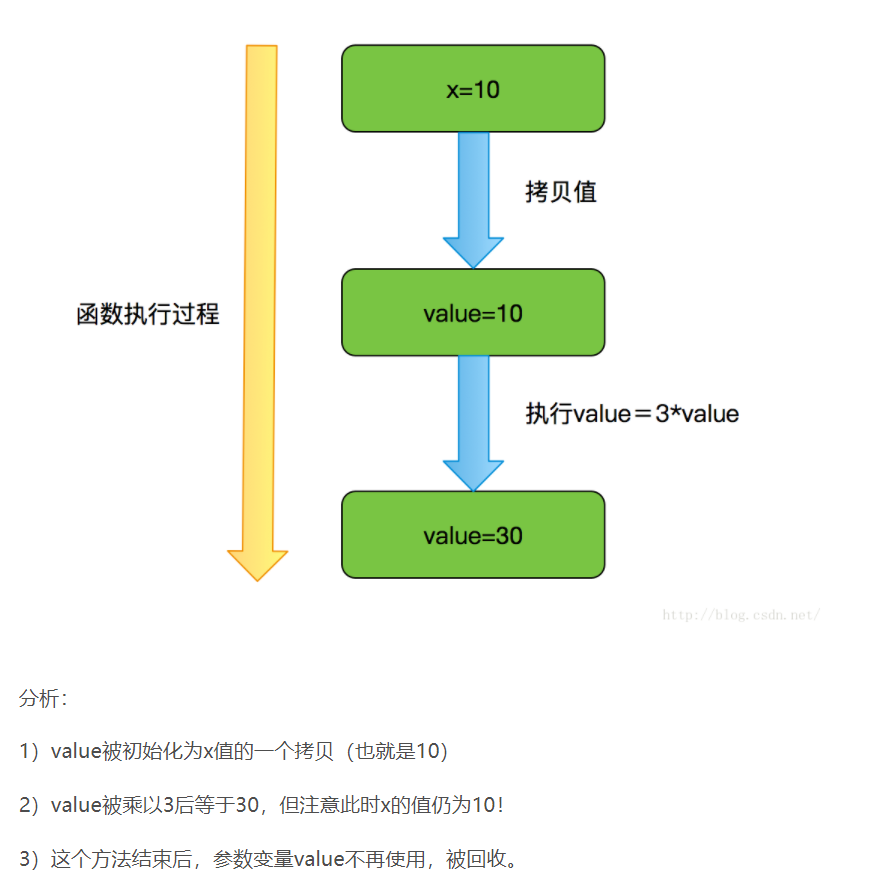
参数传递给方法的两种方式：  
（1）按值调用

（2）按引用调用

·按值调用表示方法接收的是调用者提供的值，而按引用调用则表示方法接收的是调用者提供的变量地址（在C语言中就是指针，但Java中没有指针的概念）。

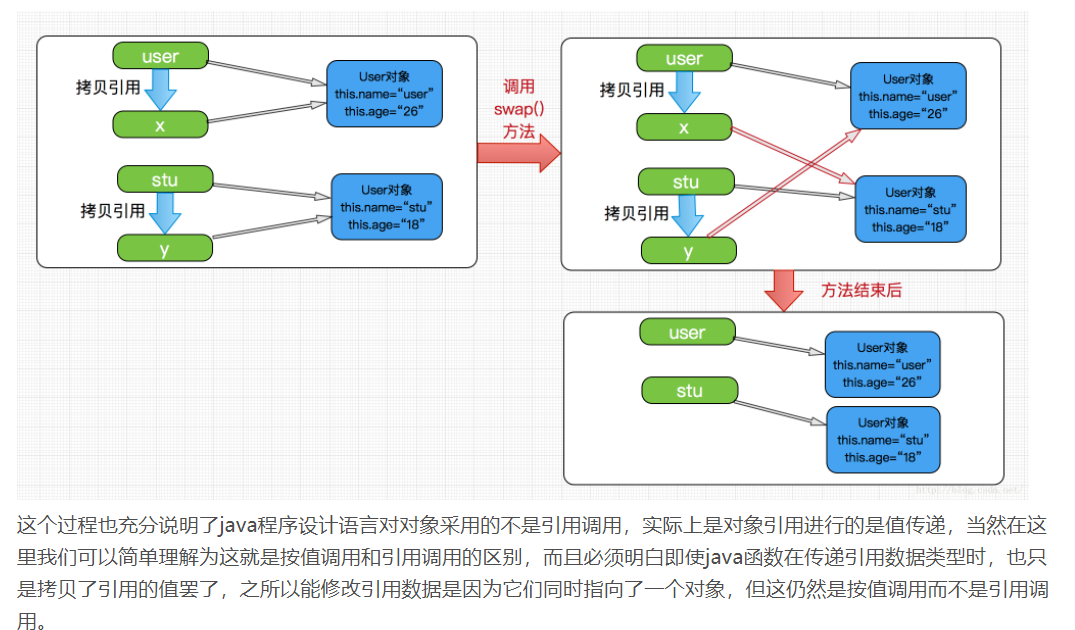
所以在Java中采用了按值调用，也就是说方法得到的是所有参数值的一个拷贝，方法并不能修改传递给它的任何参数变量的内容。

当传递方法参数类型为基本数据类型时，一个方法是不可能修改一个基本数据类型的参数。



当传递方法参数类型为引用数据类型时，一个方法将修改一个引用数据类型的参数所指向对象的值

user和stu的值并没有发生变化，也就是方法并没有改变存储在变量user和stu中的对象引用。swap方法的参数x和y被初始化为两个对象引用的拷贝，这个方法交换的是这两个拷贝的值而已，最终，所做的事都是白费力气罢了。在方法结束后x，y将被丢弃，而原来的变量user和stu仍然引用这个方法调用之前所引用的对象。



1. 多态，重写和重载

多态是指允许不同类的对象对同一消息做出响应，统一操作作用于不同的对象，可以有不同的解释，产生不同的执行结果。多态存在的三个必要条件：继承、重写、父类引用指向子类对象（子类到父类的类型转换。在多态中需要将子类的引用赋给父类对象，只有这样该引用才既能可以调用父类的方法，又能调用子类的方法）

Override重写一个方法，以实现不同的功能。一般是用于子类在继承父类时，重写父类中的方法。

重写的规则：

1. 重写方法的参数列表必须完全与被重写的方法的相同，否则不能称其为重写而是重载
2. 重写方法的访问修饰符一定要大于被重写方法的访问修饰符（public>protected>default>private）
3. 重写的方法的返回值必须和被重写的方法的返回值一致
4. 重写的方法所抛出的异常必须和被重写方法所抛出的异常一致，或者是其子类
5. 被重写的方法不能为private,否则在其子类中只是新定义了一个方法，并没有对其进行重写
6. 静态方法不能被重写，可以被重载

Overload是重载，一般是用于在一个类内实现若干重载的方法，这些方法的名称相同而参数形式不同。

重载的规则：

1. 在使用重载时只能通过相同的方法名、不同的参数形式实现。不同的参数类型可以是不同的参数类型、不同的参数个数，不同的参数顺序
2. 不能通过访问权限、返回类型、抛出的异常进行重载
3. 方法的异常类型和数目不会对重载造成影响