赋值、浅复制和深复制解析

1. 以下统一将单个复制单位，list的每一个泛型，map的每一个value 称为单位X
2. 一些基本知识

* 基本类型是按值传递的，每个基本类型都是独立的
* 不可变量虽然是按照引用传递的，但是在改变值时，就会在内存中创建新值，并指向新值
* 所以从应用角度来说，可以把基本类型和不可变量看成独立的个体
* 引用类型（必须用new创建）是按照地址传递的，且修改是在原基础上修改，不会创建新值
* X = new X(abc) 是在内存中重新开辟新的空间abc，然后X重新指向该内存，和不可变量的改变值的方式同理。
* get（list、map、实体等）一般是用来创建新值的，与某个元素相等的新值，而不是用来修改值的，因为即使修改成功了，浅复制中，拷贝的另一方也会被修改掉
* Get（list、map、实体等）得到的是引用（除基本变量）
* 从应用角度来说，get（list、map、实体）得到的基本类型和不可变量，然后再赋予新值，是不可能改变源数据的值的，这是一种错误的修改源数据方式，同理get（list、map、实体）得到引用类型，然后用new去赋予新值，同样不可能改变源数据，get（list、map、实体）修改源数据的方式只能修改非final引用类型（必须用new创建的类型），得到的是引用，且不能用等于new的方式修改，要用封装好的修改方法，例如实体的set，StringBuilder的append等，且浅拷贝会影响拷贝另一方的值。
* Set（list,实体）/put（map）一般是用来修改值的，用来将某个元素指向新的地址，所以set（new()）/set（1，new()）/put(new())都会给对应的元素分配新的内存，不会影响在浅拷贝下，拷贝的另一方的值，但是set（x.getUser）那么X在改变user的时候，由于set也指向了该地址，所以set也就改变了。
* List，map的浅复制，表现为list,map的浅复制，x的赋值，也就是说对于X对于基本类型和不可变量，list和map表现为深复制，但是当X为可变类（非final的引用类型），则表现为（list,map）的浅复制，X的赋值，list的set，map的put，是将该部分指向了另一部分内存，不会影响在浅复制下另一方的数据，但是当set（X.getUser），put(X.getUser)这样传引用的话，那么当X的user对象改变的情况下，被传的对象也会受影响。list.get和map.get可以看作是单个被复制的元素（或实体）Y，Y的get/set表现同单个赋值一样，即如果不是基本类型和不可变量，一方以任何形式改变，另一方必改变。

1. 数据的拷贝操作一般有三种方式：赋值操作（等号拷贝）、浅复制操作（常见工具类拷贝）和深复制操作（自写工具类拷贝），然后对于不同操作下，不同数据的表现为：赋值，浅复制和深复制。具体关系如下
2. 最浅层的拷贝→赋值

* 实现方式：指的是相同类型（实体、map、list等）之间用＝号进行拷贝；如：User userA = UserB Map mapA = mapB List listA = listB
* 适用对象说明：适用于一切类型，对于基本类型和不可变量是深复制
* Get/Set/ append等封装的修改函数：除基本类型和不可变量外，以任何形式改变拷贝的一方的值，另一方同样会被改变

1. 再深一层的拷贝→浅复制

* 实现方式：常见于99.99%拷贝工具类,基本不存在深复制工具类：如实体的beanUtils、mapStruct、 clone的初级实现;列表list的addAll, System.arraycopy() ;Map 的 putAll等
* 适用对象说明：如果X都是基本类型和不可变量，以及只含有基本类型和不可变量的实体，已经算是深复制了.如果X实体，且实体中存在的非final引用类型（内嵌的实体，list，可变类等必须用new创建的属性），对于该部分就是浅复制
* Get/Set/ append等封装的修改函数：对于单个复制单位 ，如果是基本类型和不可变量，以及只含有基本类型和不可变量的实体，已经算是深复制了,所以以任何方式改变一方的值另一方都不会收到影响；如果是实体，且实体中存在的非final引用类型（内嵌的实体，list，可变类等必须用new创建的属性），对于该部分就是浅复制；set是将该部分指向了另一部分内存，不会影响在浅复制下另一方的数据，但是当set（X.getUser）这样传引用的话，那么当X的user对象改变的情况下，被传的对象也会受影响；get是获得引用，和前面说的一样，对于基本类型和不可变量，不可以用get来修改，即使get出来了，赋予新值也是产生了新的变量，对于可变类属性，get也不能用new的办法来修改，否则产生的也是新值，要用封装好的修改方法（entity.set ;StringBuilder.append; list.set()）来修改List，map的浅复制，表现为list,map的浅复制，x的赋值，也就是说对于X对于基本类型和不可变量，list和map表现为深复制.但是当X为可变类（非final的引用类型），则表现为（list,map）的浅复制，X的赋值.list的set.map的put，是将该部分指向了另一部分内存，不会影响在浅复制下另一方的数据，但是当set（X.getUser），put(X.getUser)这样传引用的话，那么当X的user对象改变的情况下，被传的对象也会受影响。list.get和map.get可以看作是单个被复制的元素（或实体）Y，Y的get/set表现同单个赋值一样，即如果不是基本类型和不可变量，一方以任何形式改变，另一方必改变。

1. 最深层的拷贝→深复制，

* 实现的方式一般是构造函数，clone的深层实现，以及序列化，没有工具类，需要自己编写，
* 适用对象：单个复制单位，list的每一个泛型，map的每一个value 称为单位X，如果X是实体，存在引用类型（内嵌的实体，list，可变类等必须用new创建的属性），那么就需要深复制来阻断引用的传递。
* Get/Set/ append等封装的修改函数：拷贝的一方无论怎么改变，另一方都不会受到影响