再了解深拷贝和浅拷贝之前，先了解两个概念，引用拷贝和对象拷贝

1.引用拷贝：

是指创建一个指向对象的引用变量的拷贝，例如：

```
Employee emp1 = new Employee("Taylor", 26);
Employee emp2 = emp1;
System.out.println(emp1); // Employee@355da254
System.out.println(emp2); // Employee@355da254
```

即emp1和emp2指向堆空间中的同一个对象，这就叫引用拷贝。

2.对象拷贝是指创建对象本身的一个副本，例如：

```
Employee emp1 = new Employee("Swift", 26);
Employee emp2 = (Employee) emp1.clone();
System.out.println(emp1); // Employee@7852e922
System.out.println(emp2); // Employee@4e25154f
```

即emp1和emp2分别指向堆空间中的不同对象，这就叫对象拷贝，但需要注意的是，使用clone()方法进行对象拷贝时，必须要求Employee类实现Clonebale接口并且重写clone方法，且上述代码段所在的方法还需要处理CloneNotSupportException异常。

其中，浅拷贝和深拷贝都属于对象拷贝。

浅拷贝：1.对于数据类型是基本数据类型的成员变量，浅拷贝会直接进行值传递，也就是将该属性值复制一份给新的对象。因为是两份不同的数据，所以对其中一个对象的该成员变量值进行修改，不会影响到另一个对象拷贝得到的数据。2.对于数据类型是引用数据类型的成员变量，比如说成员变量是某个数组、某个类的对象等，那么浅拷贝会进行引用传递，也就是只是将该成员变量的引用值复制一份给新的对象。因为实际上两个对象的该成员变量都指向同一个实例。在这种情况下，在一个对象中修改改成员变量会影响到另一个对象的该成员变量值。

实现浅拷贝的方式有两种：（代码参见 https://www.cnblogs.com/shakinghead/p/7651502.html）

1.通过拷贝构造方法实现浅拷贝：

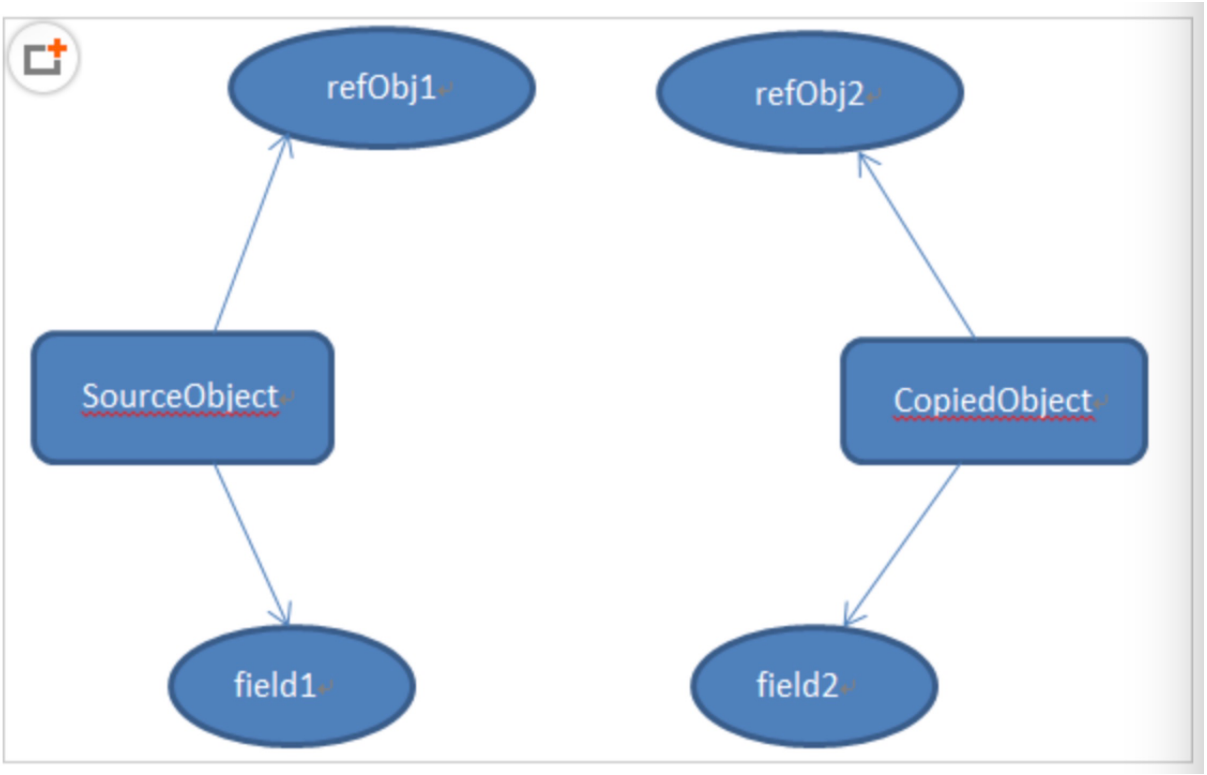
拷贝构造方法指的是该类的构造方法参数为该类的对象。使用拷贝构造方法可以很好的完成浅拷贝，直接通过一个现有的对象创建出与该对象属性相同的新的对象。

2.通过重写clone()方法进行浅拷贝

Object类是类结构的根类，其中有一个方法为protected Object clone() throws CloneNotSupportException，这个方法就是进行的浅拷贝。有了这个浅拷贝模板，我们可以通过调用clone()方法来实现对象的浅拷贝。但是需要注意的是，1.Object类虽然有这个方法，但是这个方法是受保护的(被protected修饰)，所以我们无法直接使用。2.使用clone方

法的类必须实现Cloneable接口，否则会抛出异常CloneNotSupportedException。对于这两点，我们的解决方法是，在要使用clone方法的类中重写clone()方法，通过super.clone()调用object类中的原clone方法。

深拷贝：一个类有一个对象，其成员变量中又有一个对象，该对象指向另一个对象，另一个对象又指向另一个对象，直到一个确定的实例。这就形成对象图，那么，对于深拷贝来说，不仅要复制对象的所有基本数据类型



因为创建内存空间和拷贝整个对象图，所以深拷贝相比于浅拷贝速度较慢并且花销较大。
深拷贝的实现方法主要有两种：
一、通过重写clone方法来实现深拷贝
与通过重写clone方法实现浅拷贝的思路一样，只需要为对象图的每一层的每一个对象都实现Cloneable接口并重写clone方法，最后在最顶层的类的重写clone方法中调用所有的clone方法即可实现深拷贝。简单的说就是：每一层的每个对象都进行浅拷贝=深拷贝。

二、通过对象序列化实现深拷贝
虽然层次调用clone方法可以实现深拷贝，但是显然代码量实在太大。特别对于属性数量比较多、层次比较深的类而言，每个类都要重写clone方法太过繁琐。
将对象序列化为字节序列后，默认会将该对象的整个对象图进行序列化，再通过反序列即可完美地实现深拷贝。

可以通过很简洁的代码即可完美实现深拷贝。不过要注意的是，如果某个属性被transient修饰，那么该属性就无法被拷贝了。