可变的字符串对象），那么可能不需要生成这个对象内容的新副本。仅通过保持该对象来生成它的新引用，可能就足够了。例如，为 AddressCard 类实现 copy 方法时，这个类包含 name 和 email 成员，下面实现的 copyWithZone:方法就足够了。

```
-(AddresssCard *) copyWithZone: (NSZone *) zone
{
    AddressCard *newCard = [[AddressCard allocWithZone: zone] init];

    [newCard assignName: name andEmail: email];
    return newCard;
}

-(void) assignName: (NSString *) theName andEmail: (NSString *) theEmail
{
    name = theName;
    email = theEmail;
}
```

这里没有使用 setName:andEmail:方法来设置实例变量，因为该方法会生成参数的副本，但这个练习的目的不是这样的。所以使用新的方法 assignName:andEmail:为两个变量赋值。

在这里可以不给实例变量赋值（而是对它们进行完全复制），因为复制的卡片并不会影响到原始的卡片对象（包含不可变的字符串对象）中的成员变量 name 和 email。因为默认的两个实例变量均是强引用（strong），简单的赋值操作就会创建其他的引用到这些对象，在第 17 章“内存管理和自动引用计数”中有对这个问题的详细解释。

18.5 练习

1. 根据 NSCopying 协议为 AddressBook 类实现一个 copy 方法。也实现

mutableCopy 方法是否可行？为什么？思考一下，如果使用了 AddressBook 类 book 属性的设值方法会怎样。如果将地址簿作为设置函数的参数传递，谁会拥有这个地址簿？如何修正这个问题？

2. 修改第 8 章中定义的 Rectangle 类和 XYPoint 类，使其符合<NSCopying>协议的要求。然后为两个类添加 copyWithZone:方法，确保 Rectangle 使用 XYPoint 的 copy 方法复制它的 XYPoint 成员 origin。为这些类实现可变和不可变副本是否行得通？解释原因。
3. 创建一个 NSDictionary 对象，并使用键/对象对来填充它，然后产生可变和不可变副本。这些复制是深复制还是浅复制？验证你的答案。