7.7 别返回 null 值

我认为，要讨论错误处理，就一定要提及那些容易引发错误的做法。第一项就是返回 null 值。我不想去计算曾经见过多少个几乎每行代码都在检查 null 值的应用程序。下面就是其中的一个例子：

```
public void registerItem(Item item) {
  if (item != null) {
    ItemRegistry registry = peristentStore.getItemRegistry();
    if (registry != null) {
      Item existing = registry.getItem(item.getID());
      if (existing.getBillingPeriod().hasRetailOwner()) {
        existing.register(item);
      }
    }
  }
}
```

这种代码看似不坏，其实糟透了！返回 null 值，基本上是在给自己增加工作量，也是在给调用者添乱。只要有一处没检查 null 值，应用程序就会失控。

你有没有注意到，嵌套 if 语句的第二行没有检查 null 值？如果在运行时 persistentStore 为 null 会发生什么事？我们会在运行时得到一个 NullPointer-Exception 异常，也许有人在代码顶端捕获这个异常，也可能没有捕获。两种情况都很糟糕。对于从应用程序深处抛出的 NullPointerException 异常，你到底该作何反应呢？

可以敷衍说上列代码的问题是少做了一次 null 值检查，其实问题多多。如果你打算在方法中返回 null 值，不如抛出异常，或是返回特例对象。如果你在调用某个第三方 API 中可能返回 null 值的方法，可以考虑用新方法打包这个方法，在新方法中抛出异常或返回特例对象。

在许多情况下，特例对象都是爽口良药。设想有以下一段代码：

```
List<Employee> employees = getEmployees();
if (employees != null) {
  for(Employee e : employees) {
    totalPay += e.getPay();
  }
}
```

现在，getExployees 可能返回 null 值，但是否一定要这么做呢？如果修改 getEmployees，返回空列表，就能使代码整洁起来：

```
List<Employee> employees = getEmployees();
for(Employee e : employees) {
  totalPay += e.getPay();
}
```