空对象模式

题目标签

- 题目难度：一般
- 知识点标签：设计模式，空对象设计模式
- 课程时长：20分钟

题目描述

面试中的问题：如何做到null值的优雅规避，项目中如何更好避免空指针异常（NullPointerException）

案情回顾

我们在写Java Web项目时有个方法叫getBookInfoByIndex，目的是返回Book对象之后再调用book.show方法来获取某个书籍的基本信息。如果我们输入1,2,3等等可以查询到的数字，程序运行正常，但是一旦我们输入-1，或者超过书本数量的index，那不就会报错么。此时我们比较常规的做法就是在客户端加一个判断，判断是否为null。如果为null的话，就不再调用show()方法。但是，你有没有考虑过？这样做，确实消除了报错，但是这样做真的好吗？你想如果在一段程序中有很多处调用getBookInfoByIndex()方法，岂不是很多处都要判断book对象是否为null？这还不算坏，如果哪一处没有判断，然后报错了，很有可能导致程序没法继续运行甚至崩溃。

代码实现

我们将创建一个定义操作（在这里，是客户的名称）的 AbstractCustomer 抽象类，和扩展了 AbstractCustomer 类的实体类。工厂类 CustomerFactory 基于客户传递的名字来返回 RealCustomer 或 NullCustomer 对象。
NullPatternDemo，我们的演示类使用 CustomerFactory 来演示空对象模式的用法。

步骤 1

创建一个抽象类。
AbstractCustomer.java

```
public abstract class AbstractCustomer {
   protected String name;
   public abstract boolean isNil();
   public abstract String getName();
}
```

步骤 2

创建扩展了上述类的实体类。
RealCustomer.java

```
public class RealCustomer extends AbstractCustomer {

   public RealCustomer(String name) {
```

```
      this.name = name;
   }

   @Override
   public String getName() {
      return name;
   }

   @Override
   public boolean isNil() {
      return false;
   }
}
```

NullCustomer.java

```
public class NullCustomer extends AbstractCustomer {

   @Override
   public String getName() {
      return "Not Available in Customer Database";
   }

   @Override
   public boolean isNil() {
      return true;
   }
}
```

步骤 3

创建 CustomerFactory 类。
CustomerFactory.java

```
public class CustomerFactory {

   public static final String[] names = {"Rob", "Joe", "Julie"};

   public static AbstractCustomer getCustomer(String name){
      for (int i = 0; i < names.length; i++) {
         if (names[i].equalsIgnoreCase(name)){
            return new RealCustomer(name);
         }
      }
      return new NullCustomer();
   }
}
```

步骤 4

使用 CustomerFactory，基于客户传递的名字，来获取 RealCustomer 或 NullCustomer 对象。
NullPatternDemo.java

```
public class NullPatternDemo {
   public static void main(String[] args) {
```

```
      AbstractCustomer customer1 = CustomerFactory.getCustomer("Rob");
      AbstractCustomer customer2 = CustomerFactory.getCustomer("Bob");
      AbstractCustomer customer3 = CustomerFactory.getCustomer("Julie");
      AbstractCustomer customer4 = CustomerFactory.getCustomer("Laura");

      System.out.println("Customers");
      System.out.println(customer1.getName());
      System.out.println(customer2.getName());
      System.out.println(customer3.getName());
      System.out.println(customer4.getName());
   }
}
```

步骤 5

验证输出。

```
Customers
Rob
Not Available in Customer Database
Julie
Not Available in Customer Database
```

总结

空对象就是一个正常的对象，只不过你用这个对象来替代null。作用就是 当使用null表示缺少对象时，在每次引用前都要测试其是否为null，因此需要在代码中加入判断语句，当判断语句变多时，代码就变得杂乱，使用空对象可以减少判断的语句。