



# 软件工程

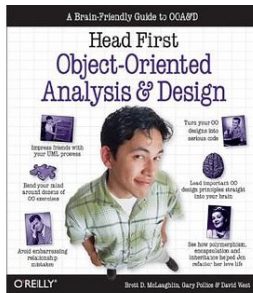
## 面向对象编程三部曲

Spring 2022, SWUFE

复习

OOA&D

## Head First Object-Oriented Analysis and Design



作者: Brett D. McLaughlin / Gary Pollice / Dave West

出版社: O'Reilly Media

副标题: A Brain Friendly Guide to OOA&D

出版年: 2006-12-4

页数: 636






定价: USD 49.99

装帧: Paperback

ISBN: 9780596008673

豆瓣评分

8.6  125人评价

5星		49.6%
4星		38.4%
3星		11.2%
2星		0.8%
1星		0.0%

掌握面向对象的语法不意味着能写出好的软件

设计模式的灵活应用往往需要大量经验

## 1. OO编程三部曲



## 1.1 背景



Rick开了一个卖吉他的店铺。

由于吉他的种类和数量繁多，他需要有一个**搜索系统**帮助他根据用户的需求找到匹配的吉他。

## 1.1 第一版系统

Inventory
guitars: List
addGuitar(String, double, String, String, String, String, String) getGuitar(String): Guitar search(Guitar): Guitar

Guitar
serialNumber: String price: double builder: String model: String type: String backWood: String topWood: String
getSerialNumber(): String getPrice(): double setPrice(float) getBuilder(): String getModel(): String getType(): String getBackWood(): String getTopWood(): String

有个Bug!

```
public Guitar search(Guitar searchGuitar) {  
    for (Guitar guitar : guitars) {  
        // ignore serial number since it is unique  
        if (guitar.getBuilder().equals(searchGuitar.getBuilder()) &&  
            guitar.getModel().equals(searchGuitar.getModel()) &&  
            guitar.getBackWood().equals(searchGuitar.getBackWood()) &&  
            guitar.getTopWood().equals(searchGuitar.getModel()) &&  
            guitar.getPrice() <= searchGuitar.getPrice()) {  
            return guitar;  
        }  
    }  
    return null;  
}
```

## 单元测试

```
inventory.addGuitar( serialNumber: "011", builder: "USA", model: "K110", type: "acoustic",  
    backWood: "Awood", topWood: "Awood", price: 180);  
inventory.addGuitar( serialNumber: "020", builder: "UK", model: "S112-s", type: "electric",  
    backWood: "Bwood", topWood: "Awood", price: 160);  
inventory.addGuitar( serialNumber: "022", builder: "UK", model: "S330", type: "electric",  
    backWood: "Bwood", topWood: "Awood", price: 180);  
inventory.addGuitar( serialNumber: "023", builder: "UK", model: "S330-s", type: "acoustic",  
    backWood: "Bwood", topWood: "Awood", price: 200);
```

```
Guitar guitar = new Guitar(null, "UK", "S330", "electric", "Bwood",  
    "Bwood", 200);
```

```
assertEquals(inventory.search(guitar).getSerialNumber(), "022");
```

## 解决Bug

```
guitar.getTopWood().equals(searchGuitar.getModel())
```

```
Guitar guitar = new Guitar( serialNumber: null, builder: "UK", model: "S330", type: "electric",  
                             backWood: "Bwood", topWood: "Awood", price: 150);  
  
assertEquals(inventory.search(guitar).getSerialNumber(), actual: "022");
```

UK, uk, GB, gb, GBR , 英国 , 英



## 使用枚举类型

```
public enum Builder {  
    CHINA, USA, UK, JP  
}
```

```
public class Guitar {  
    private String serialNumber;  
    private Builder builder;  
    private String model;  
    private Type type;  
    private Wood backWood;  
    private Wood topWood;  
    private double price;
```

```
if (guitar.getBuilder() == searchGuitar.getBuilder() &&  
    guitar.getModel().equalsIgnoreCase(searchGuitar.getModel()) &&  
    guitar.getBackWood() == searchGuitar.getBackWood() &&  
    guitar.getTopWood() == searchGuitar.getTopWood() &&  
    guitar.getPrice() <= searchGuitar.getPrice()) {  
    return guitar;  
}
```



## 小任务

- 第一步：git clone [git@github.com:ChenZhongPu/swufe-se.git](https://github.com:ChenZhongPu/swufe-se.git)
- 第二步：导入 week9/guitar\_v1 工程 ( **Gradle** )
- 第三步：阅读代码并进行单元测试

## 回顾



Make sure your software does  
what the customer wants it to do

```
public Guitar search(Guitar searchGuitar) {  
    for (Guitar guitar : guitars) {  
        // ignore serial number since it is unique  
        if (guitar.getBuilder() == searchGuitar.getBuilder() &&  
            guitar.getModel().equalsIgnoreCase(searchGuitar.getModel()) &&  
            guitar.getBackWood() == searchGuitar.getBackWood() &&  
            guitar.getTopWood() == searchGuitar.getTopWood() &&  
            guitar.getPrice() <= searchGuitar.getPrice()) {  
            return guitar;  
        }  
    }  
    return null;  
}
```



## OO设计原则

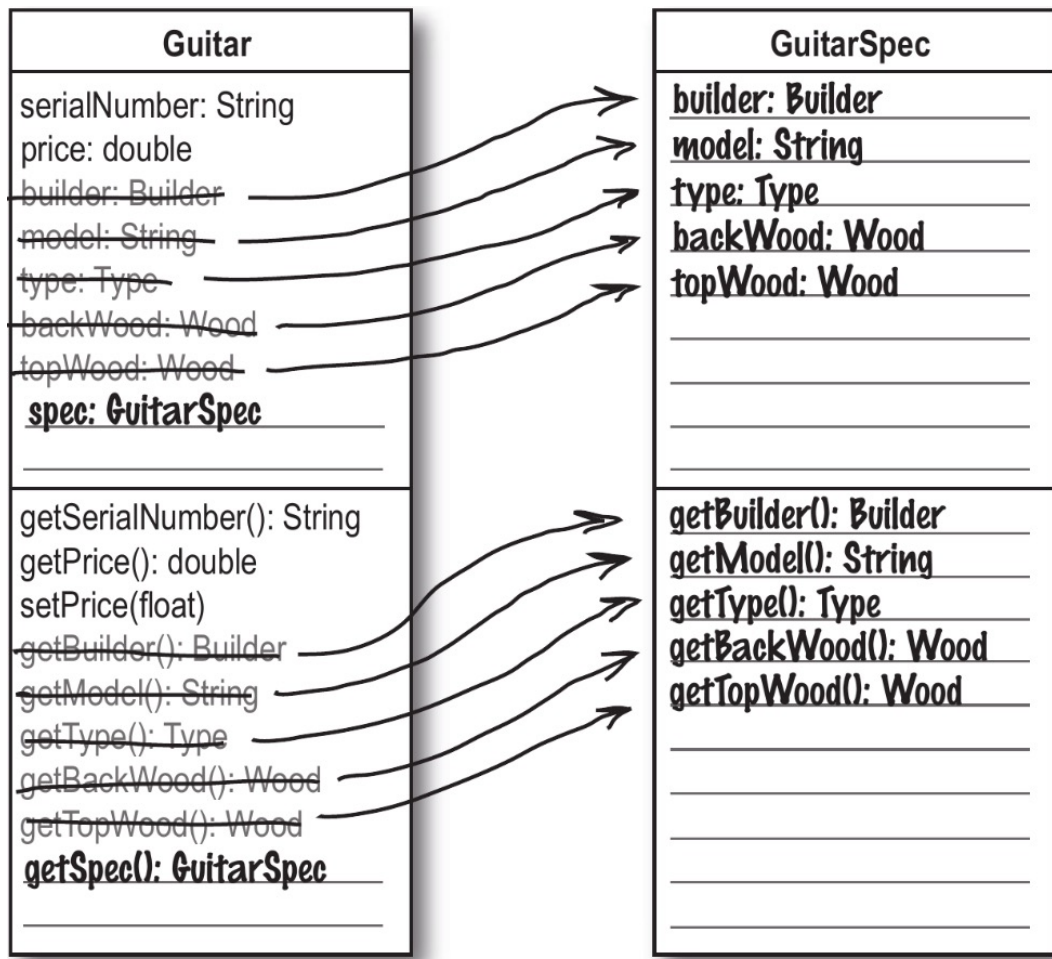
```
Guitar guitar = new Guitar(null, Builder.UK, "S330", Type.ELECTIRC,  
    Wood.AWOOD, Wood.AWOOD, 200);  
assertEquals(inventory.search(guitar).getSerialNumber(), "022");
```

- Objects should do what their names indicate.
- Each object should represent a single concept.
- Unused properties are a dead giveaway.

如果一个对象的部分属性总是空值，  
这往往意味着它违背了单一责任原则

## 1.2 利用OO原则改进

- 不同概念应该使用不同类进行抽象和封装
- 如果有代码重复问题，就使用 has-a 的原则封装。



## 回顾

实现用户需求

利用OO原则  
实现灵活性

实现易维护  
和代码复用

重点关注类的设计和代码重复问题

## 1.3 代码复用和可维护问题

改变是软件开发过程中永远不变的事情！

GuitarSpec
<u>builder: Builder</u>
<u>model: String</u>
<u>type: Type</u>
<u>backWood: Wood</u>
<u>topWood: Wood</u>



Rick计划在吉他的描述中添加弦的数量

錦瑟無端五十弦  
一弦一柱思華年



```
public class GuitarSpec {  
    private Builder builder;  
    private String model;  
    private Type type;  
    private int numStrings;  
    private Wood backWood;  
    private Wood topWood;
```

Inventory的search()方法和GuitarSpec耦合。

这意味着：每当GuitarSpec发生改变，  
Inventory还是需要修改。

```
public Guitar getGuitar(GuitarSpec guitarSpec, double price) {  
    for (Guitar guitar : guitars) {  
        if (guitar.getGuitarSpec().getBuilder() == guitarSpec.getBuilder() &&  
            guitar.getGuitarSpec().getType() == guitarSpec.getType() &&  
            guitar.getGuitarSpec().getModel().equalsIgnoreCase(guitarSpec.getModel()) &&  
            guitar.getGuitarSpec().getBackWood() == guitarSpec.getBackWood() &&  
            guitar.getGuitarSpec().getTopWood() == guitarSpec.getTopWood() &&  
            guitar.getGuitarSpec().getNumStrings() == guitarSpec.getNumStrings() &&  
            guitar.getPrice() <= price) {  
            return guitar;  
        }  
    }  
    return null;
```

```
public Guitar search(GuitarSpec guitarSpec, double price) {  
    for (Guitar guitar : guitars) {  
        if (guitar.getGuitarSpec().matches(guitarSpec) &&  
            guitar.getPrice() <= price) {  
            return guitar;  
        }  
    }  
    return null;  
}
```

把可能变化的部分  
移出去。

代码的低耦合！

## 回顾

实现用户需求

利用OO原则  
实现灵活性

实现易维护  
和代码复用

设计模式、低耦合、  
开闭原则...

```
public Guitar search(GuitarSpec guitarSpec, double price) {  
    for (Guitar guitar : guitars) {  
        if (guitar.getGuitarSpec().matches(guitarSpec) &&  
            guitar.getPrice() <= price) {  
            return guitar;  
        }  
    }  
    return null;  
}
```

如果需要在运行时实现高度自定义的搜索，还可以使用策略模式，将搜索算法封装起来！



## 小任务

- 为上述的吉他搜索系统使用**策略模式**。

## 2. 好的设计=灵活的软件

Nothing ever stays the same





## 小结



实现用户需求



利用OO原则  
实现灵活性



实现易维护和  
代码复用

OOA&D is about writing great  
software!