代码的重构

Refactoring of Code

for GreenTeaJUG

结合《重构-改善既有代码设计》的实战

吴璞渊 wupuyuan@gmail.com

1 现有可用的代码需要修改吗？

- 金科玉律？ -- 如果功能可用，就别改代码！
- 代码的“变质”是量变的累积
  - 系统发布赶时间点，部分功能的实现会敷衍。
  - 业务变化在最初设计之外，相关的代码实现会敷衍。
  - 团队的管理。
  - 代码的坏影响会扩散。
- 现有代码是需要被**修改**的

2 什么是重构

- 重构是代码的整理，但不仅仅是整理。
- 重构是代码的优化，但不是什么都优化。
- 重构是用新(更合适)的代码结构，取代旧的。
- 重构的重心是改善代码的可读性。

3 重构怎么安排？

- 功能独立的代码模块，在同一时间由一个人重构，但不应只安排一个人。
- 不应该为重构单独安排时间，除非很闲。
- 重构包含code review。

4 重构需要的技能

- 软件工程
- 设计模式
- 单元测试(习惯)

5 重构的原则-降低风险

- 小步迭代

  重构是增量式的行为，每次只触及部分代码，不是全部。

- 单元测试

  对重构代码的检验，应在“单元测试”可验证的范围内。

6 重构的第一步-方法细分

- 暂时抛弃面向对象的思想，换成以方法(函数)为主体的函数编程。
- 方法细分：将看不懂的代码块、表达式都封装成小方法。
- 取个有助于理解的方法名。
- 适合用单元测试验证。
- 小方法更适合做内联优化。

```
public void method(){
  if( (exp1 || exp2) && exp3){
    if(exp4 && exp5){
        {Code block 1}
    }else{
        while(exp6){
           {Code block 2}
        }
    }
  } else {
    try{
        {Code block 3}
    }catch(){}
  }else{
    String str
    try{
     if(exp7){
        str =  "str1" ;
     }else{
        str = statement1;
     }
    }catch(){}
  }
}
```

7 重构的第一步-方法细分

```
public void method(){

  if(isMetch1()){

    if(isMetch2()){

        subMethod1();

    }else{

        loopByKey__();

    }
  } else {

        subMethod3();

  }else{

        createName();

  }
}
```

```
public boolean isMetch1(){
  return (exp1 || exp2) && exp3;
}


public boolean isMetch2(){
  return exp4 && exp5;
}


public void subMethod1(){
  {Code block 1}
}


public void loopByKey__(){
  while(exp6){
    {Code block 2}
  }
}
```

```
public void subMethod3(){
  try{
    {Code block 3}
  }catch(){
  }
}



public String createName(){
  String str
  try{
   if(exp7){
      str =  "str1" ;
   }else{
      str = statement1;
   }
  }catch(){}
  return str;
}
```

8 重构-code review

- 反推条件表达式。
- 寻找“功能”相同的方法。
- 寻找无用的方法。

9 重构-代码的提炼

- 重拾面向对象的模式
- 封装新的方法
- 梳理继承关系
- 创建公共方法类
- 移动方法

10 重构-合适的注释

- 包含业务和场景的说明。
- 包含数据模型的说明。
- 包含代码修改的版本信息、修改时间。

重构的实战 系统创建之初

- 系统只售卖两个产品。
- 每个产品对应的处理类有个execute方法。
- 两个execute方法，90%相同。

重构的实战 运行两年之后

- 售卖系统销售50多个产品。
- 产品的数据模型新增类型(type)和付费模式(payment)字段。
- type分8种：基础、测试、性能、分析等
- payment 分4种：预付、后附、按量付、公测。
- 每个产品的处理类中的execute方法中，70%“看起来”差不多的。

重构的实战 从新任务为切入点开始

- 接到新的产品ProductZ的开发任务，和现有的ProductX、ProductY类似。
- 原有的方法：copy一份execute方法改改就OK。
- 重构，从ProductX、ProductY的方法分解开始。

重构的实战 方法的分解

```
Public void execute(){
  // 校验
  verify();
  query1();
  if(exp1 || exp2 ...){
    action1();
  }else{
    action2();
  }

  method1();
}
```

```
Public void execute(){
  // 校验
  verify();
  query2();
  if(exp1 || exp2 ...){
    action1();
  }else{
    action3();
  }

  method2();
}
```

重构的实战 方法的分解

```
Public void execute(){
  // 校验
  verify();
  // 查询指定产品的信息，对应表
XXX_PRODUCT
  queryByProductId(X);
  if( exp1 || exp2 ... ){
    // 查询入口免费的信息
    queryFree()
  }else{
    queryExclusive1();
  }

  loopByKey___();
  // 组织X的展示信息
  dispX();
}
```

```
Public void execute(){
  // 校验
  verify();
  // 查询指定产品的信息,对应表
XXX_PRODUCT
  queryByProductId(Y);
  if( exp1 || exp2 ... ){
    // 查询入口免费的信息
    queryFree()
  }else{
    queryExclusive2();
  }

  loopByKey___();
  // 组织X的展示信息
  dispY();
}
```

重构的实战 方法的分解

```
Public void execute(){
  // 校验
  verify();
  // 查询指定产品的信息，对应表XXX_PRODUCT
  queryByProductId(X);
  // 是不是公测产品
  if( isOBT()){
    // 查询入口免费的信息
    queryFree()
  }else{
    // 查询专属信息
    queryExclusive(X);
  }

  // 过滤
  filtration();
  // 组织X的展示信息
  dispX();
}
```

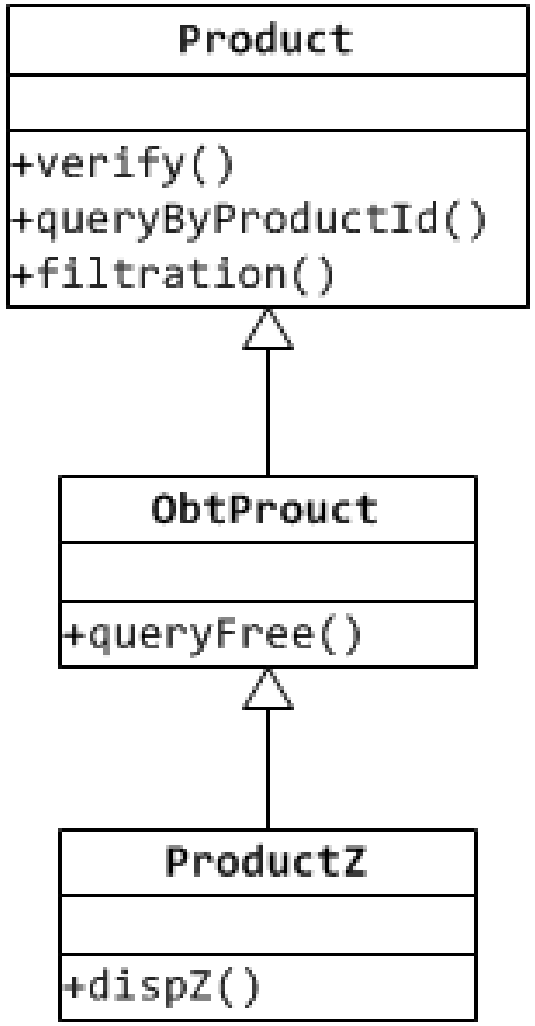
```
Public void execute(){
  // 校验
  verify();
  // 查询指定产品的信息,对应表XXX_PRODUCT
  queryByProductId(Y);
  // 是不是公测产品
  if( isOBT()){
    // 查询入口免费的信息
    queryFree()
  }else{
    // 查询专属信息
    queryExclusive(Y);
  }

  // 过滤
  filtration();
  // 组织X的展示信息
  dispY();
}
```

重构的实战 代码提炼

针对ProductX和ProductY的execute方法的分解基本完成。

原来所有的product处理类都是平级关系。

对于要新发布的ProductZ完全可以采用新的代码结构。

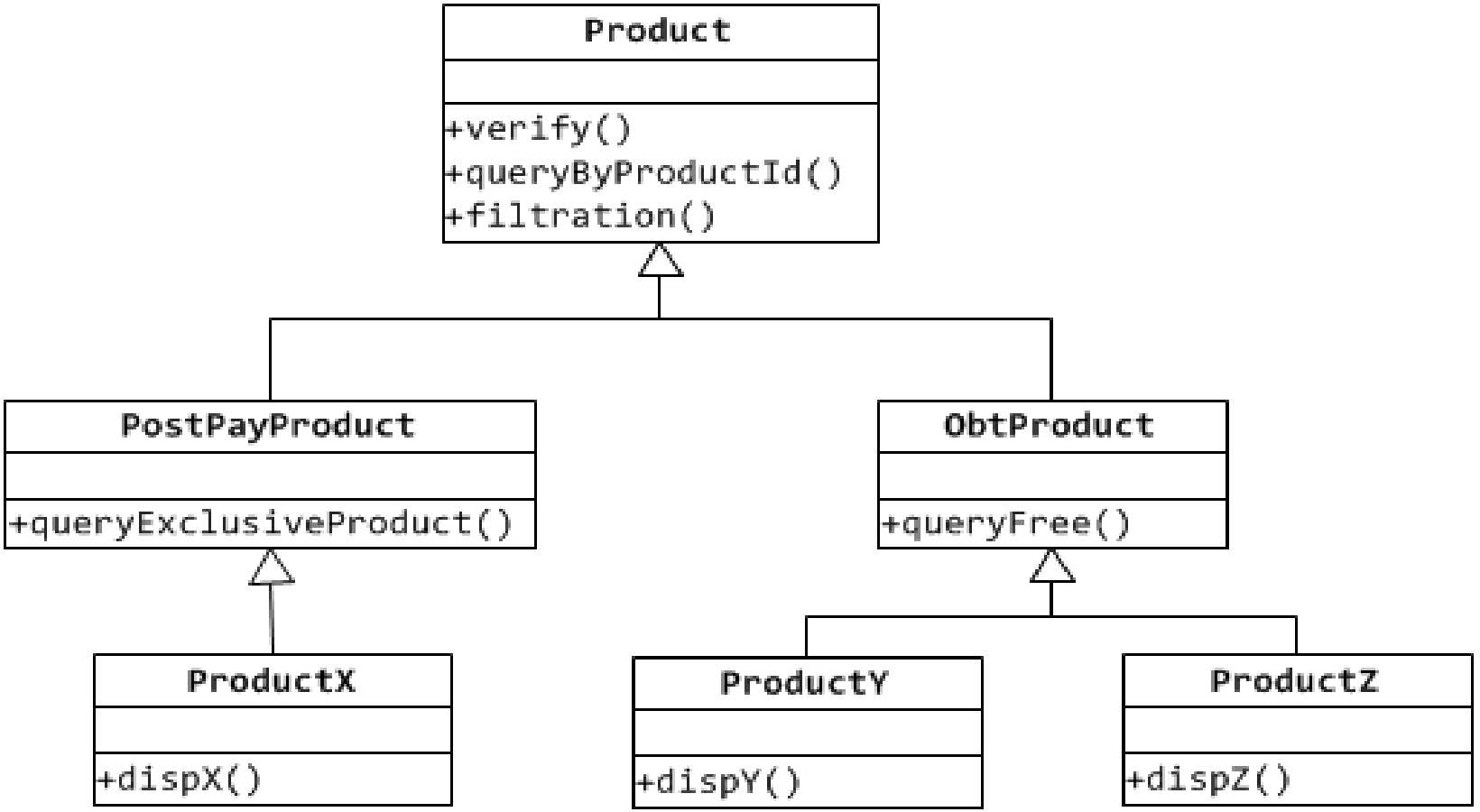


重构的实战 用新需求的发布验证重构正的确性

- 先发布ProductZ；已有的代码ProductX和ProductY，暂时先不发布，虽然它们完成了方法的分解。

- 经过ProductZ的发布流程，验证新模型/方法的正确性以后，再发布ProductX和ProductY。这样只需要做代码删除或移动工作和少量的单元测试即可。

重构的实战 调整已有的类



重构的实战

重构之后

- 在经历了代码的重构后，别忘记对相关的配置项做清理。
- 如果发现，架构、数据模型或通信接口有优化点，提出来。这是重构的另一个核心价值。

谢谢聆听