## 一、重复的代码提炼成函数：

1. 同一个class内的两个函数含有相同的表达式（expression）
2. 两个subclasses有相同的表达式，或相似的表达式。抽取相同的表达式（属性和方法），放在父类里，两个子类再去继承。

## 二、把过长的函数变小

1、找出只被赋值一次的临时变量。

2、**将该临时变量声明为final**

3、编译：这可确保该临时变量的确只被赋值一次。

4、将临时变量**等号右侧部分提炼到一个独立函数中**；

5、首先将函数声明为private。日后你可能会发现有更多class需要使用 它，彼时你可再放松对它的保护。

6、编译，测试：确保提炼出来的函数无任何连带影响（副作用），结果不变；

7、**把临时变量全替换成独立出来的函数**；

例如：

double getPrice() {

int basePrice = \_quantity \* \_itemPrice;

double discountFactor;

if (basePrice > 1000) discountFactor = 0.95;

else discountFactor = 0.98;

return basePrice \* discountFactor;

}

优化后：

double getPrice() {

return basePrice() \* discountFactor();

}

private double discountFactor() {

if (basePrice() > 1000) return 0.95;

else return 0.98;

}

private int basePrice() {

return \_quantity \* \_itemPrice;

}

通过以上的优化，一个大函数，已经变成了多个小函数，重点是代码的可读性提高了，顺带的代码量变少

## 三、参数对象化

当你看到一个函数的入参有四，五个，甚至更多时，且好几个函数都使用这组入参，这时就要用参数对象化来优化代码。这些函数可能隶属同一个class，也可能隶属不同的classes 。这样一组参数就是所谓的Date Clump （数据泥团）」。这时用一个对象封装这些参数，再用对象取代它们

## 四、提炼类和提炼子类

如果想利用单一class做太多事情，其内往往就会出现太多instance变量。一旦如此，Duplicated Code也就接踵而至了。Extract Class 是Extract Subclass 之外的另一种选择，两者之间的抉择其实就是委托（delegation）和继承（inheritance）之间的抉择。

情况一：某个class做了应该由两个classes做的事。（Extract Class）

情况二：class 中的某些特性（features）只被某些（而非全部）实体（instances）用到。Extract Subclass（提炼子类）