局部变量是阻碍函数重构的最大问题，为什么？因为局部变量需要在方法中传来传去

1. 提炼函数

将一段代码从过长的函数中提炼为一个函数，前提是这点代码有意义

如果函数需要返回多个值：

推荐拆分成多个方法，每个方法只返回一个值

示例：

// 重构前

public void 转账(){

    // 假设这里有 100 多行代码

    // 但理想情况是每个方法 30 行左右

}

// 重构后

public void 转账(){

    if(!验证密码()){

        throw new Error();

}

    if(!验证账号金额()){

        throw new Error();

    }

}

private bool 验证密码(){

    ...

}

private bool 验证账号金额(){

    ...

}

1. 内联函数

一个函数的代码够简单，一眼可以看懂，则移除该函数，将代码移到调用者

如上示例：

如果“验证密码”“验证账号金额”的逻辑很简单，那么我们就不需要提炼该函数了

1. 以查询取代临时变量

如果临时变量被赋值后没有修改，则以一个查询方法取代这个临时变量（函数中应该少用临时变量）

// 重构前

public void 转账(){

    // 经过一系列复杂的技术，我们得到了sum

    ...

    var sum = ...;

    方法1(sum);

}

void 方法1(){ ... }

// 重构后

public void 转账(){

    方法1();

}

void 方法1(){ getSum(); ... }

private int getSum(){

    // 经过一系列复杂的技术，我们得到了sum

    ...

    return sum;

}

1. 分解临时变量

一个临时变量只能承担一个职责，如果承担了多个职责，那应该新起变量

// 重构前

public void 转账(){

    var sum = getASum();

    ...

    sum = getBSum()

}

// 重构后

public void 转账(){

    var aAum = getASum();

    ...

    var bBum = getBSum()

}

1. 以函数对象取代函数

如果一个函数过于复杂，则应该使用新起一个对象用于完成这个函数的功能，将函数用到的局部变量和函数所在类的引用（如果新类没有引用到这个类，则不传）通过构造函数传到新类

1. 引入注释型变量

将一个复杂的表达式的结果赋予一个变量，变量名解释其用途

// 重构前

public void 转账(){

    方法1(a\*c\*d-l-i+q);

}

// 重构后

public void 转账(){

    var sum = a\*c\*d-l-i+q;

    方法1(sum);

}