****

**软件测试技术第一次实验报告**



**学 院 智能与计算学部**

**专 业 3016218077**

**年 级 2016**

**姓 名 张凯炫**

**2019年 3 月 12 日**

# 软件测试技术第一次实验报告

1. 需求分析（描述具体需求）

问题如下，现在钱包里面有一张50元，一张20元，两张5元，三张1元，需要判断能否找零。

详细需求如下：

* 输入需要找零的数目
* 根据钱包里面的钱的数量，判断能否找零
* 输出判断结果

1. 概要设计（简单描述设计思路，配合UML图）

类Calculator有一个成员变量looseChange代表可找的零钱的面额，还有一个公有静态函数looseOk(int testMoney)，对于测试的金额testMoney，函数looseOk计算后返回对应的结果。

在分析数据的特征后，算法设计采用了贪心的思想，先用大的面额去比较，然后用小的面额去比较，直到比较完毕。

1. 详细设计（详细描述具体如何实现，附代码及说明）

1.程序设计

程序代码如下

在类Calculator 中先定义了静态全局变量looseChange，用于存储找零的面额值，静态公有函数looseOk参数testMoney是测试的金额，如果可以找零返回true，如果不可以找零返回false，

对于looseOk函数，先判断大的面额是否可以找零，如果可以找零， number数组对应的值减一，反之，i增加，判断下一个面额的是否可以找零，当所有面额都判断后，依然无法完全找零，则返回false ，反之是true

package experiment1;

public class Calculator {

public static int [] looseChange = { 50 , 20 , 5 , 1};

public static boolean looseOk(int testMoney)

{

int number [] = {1 , 1 , 2 , 3};//store money's number

int i = 0;

for( i = 0 ; i < looseChange.length && testMoney != 0 ; )

{

if (number[i] == 0)

{//money has used

i++;

continue;

}

if (testMoney >= looseChange[i])

{//use money

testMoney -= looseChange[i];

number[i] -= 1;

}

else

{//next money

i++;

}

}

if (testMoney == 0)

{ return true; }

else

{

return false;

}

}

public static void main(String s[])

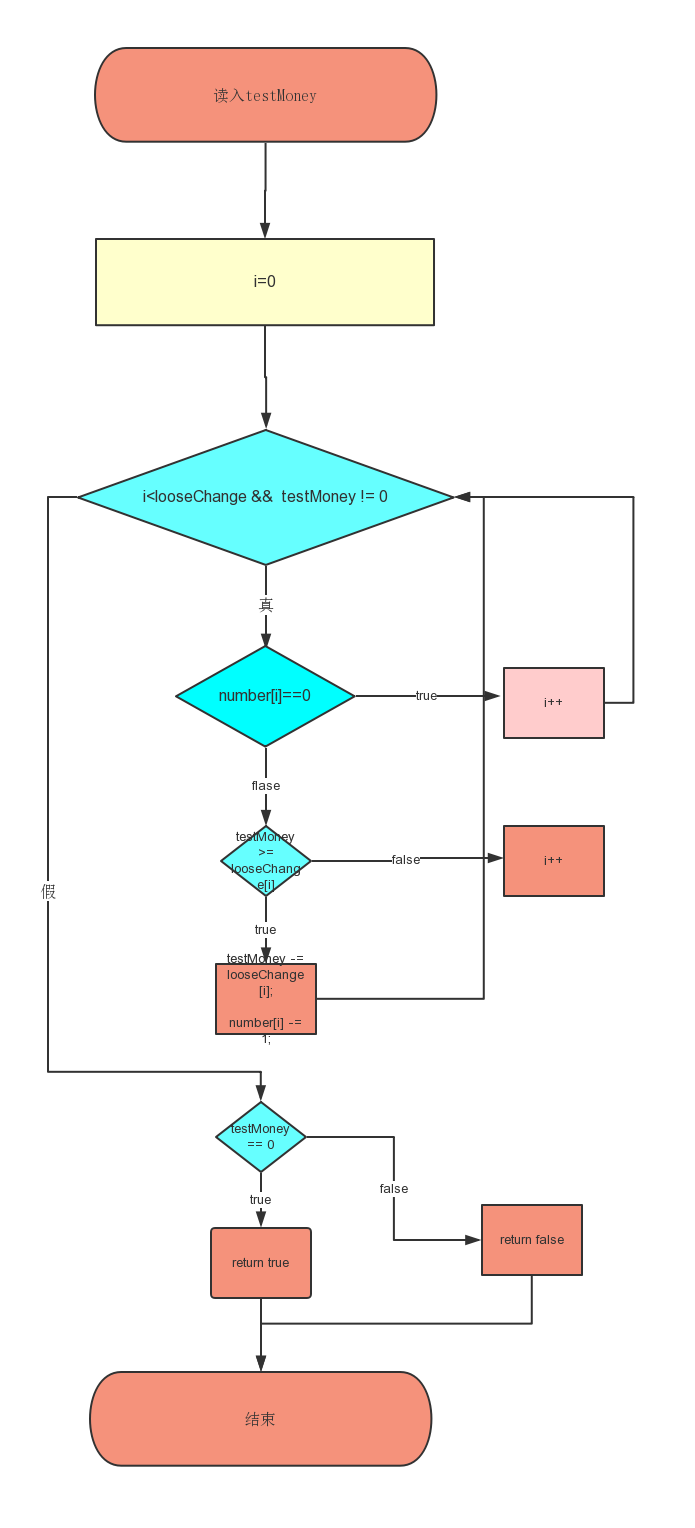
{

int testMoney = 99;

Calculator.looseOk(testMoney);

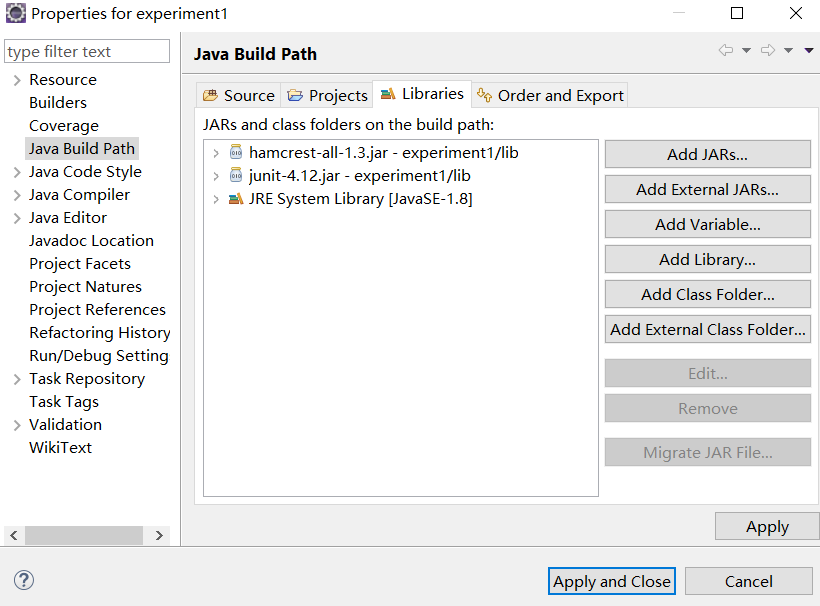
}

}

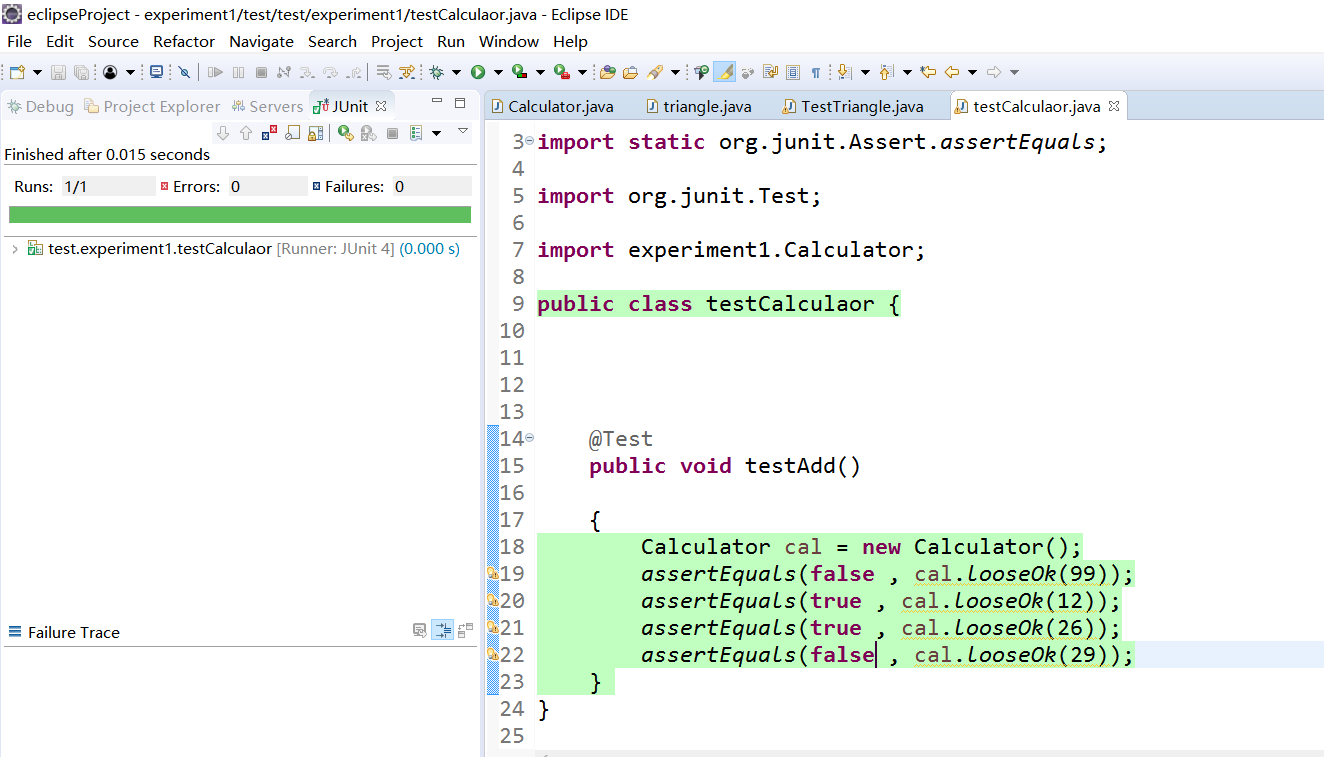


1. Junit、hamcrest安装过程

先将Junit和hamcrest的jar包导入项目

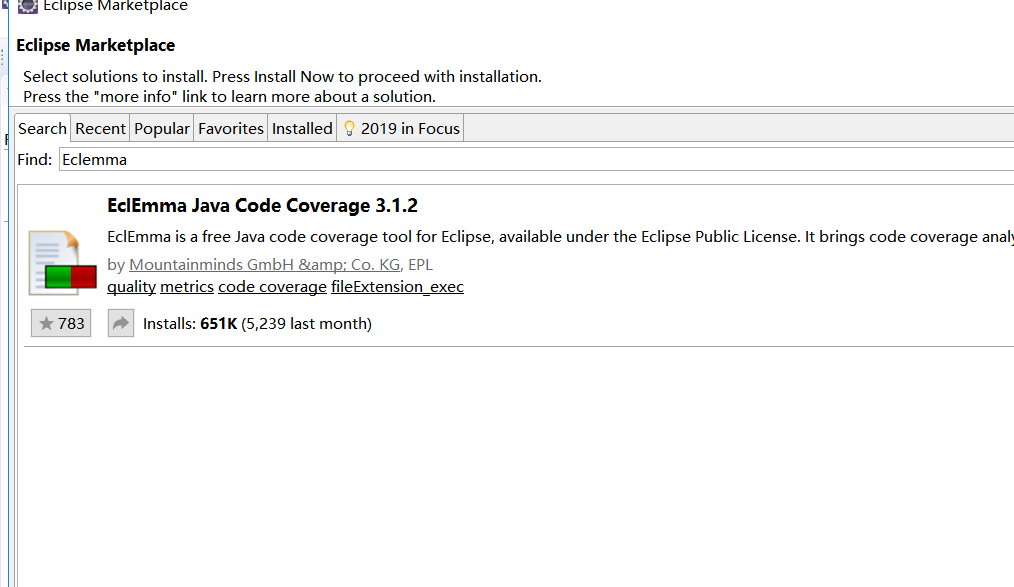


此时在编写好测试代码后，即可使用Junit测试

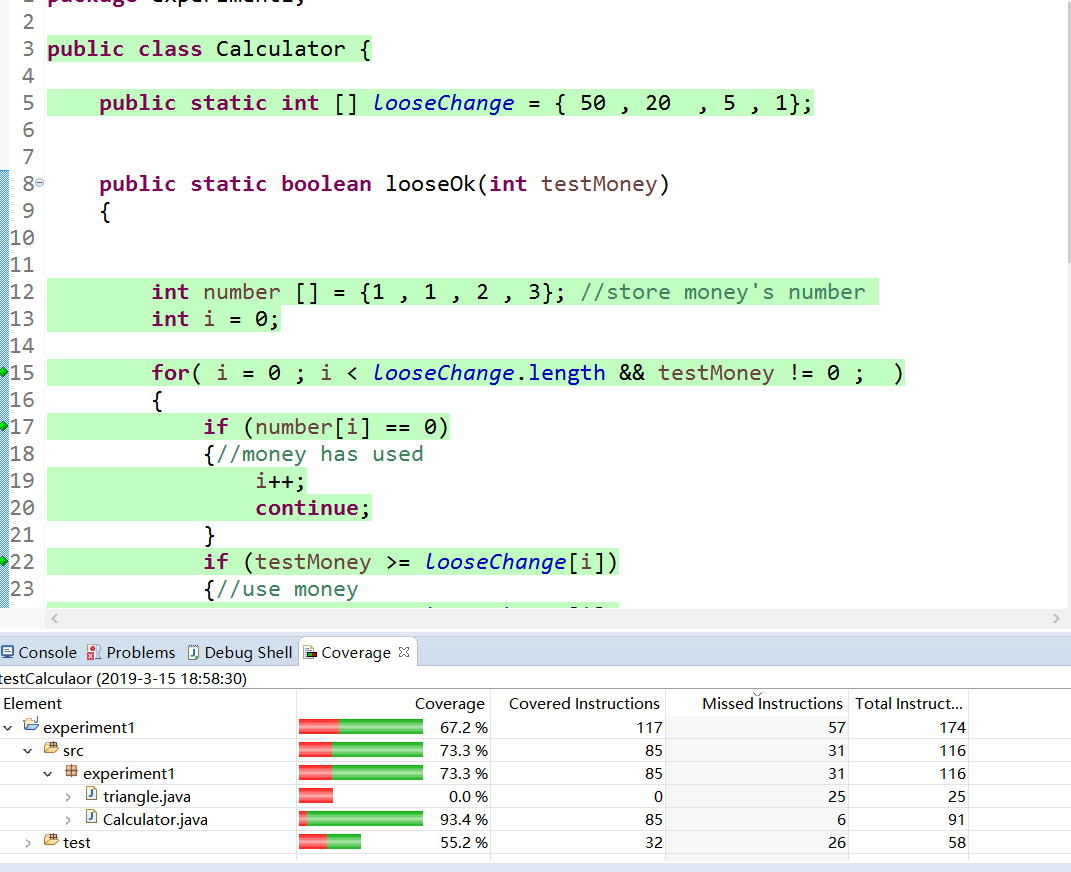


1. Eclemma安装

在eclipse的marketplace中搜索Eclemma插件，安装即可



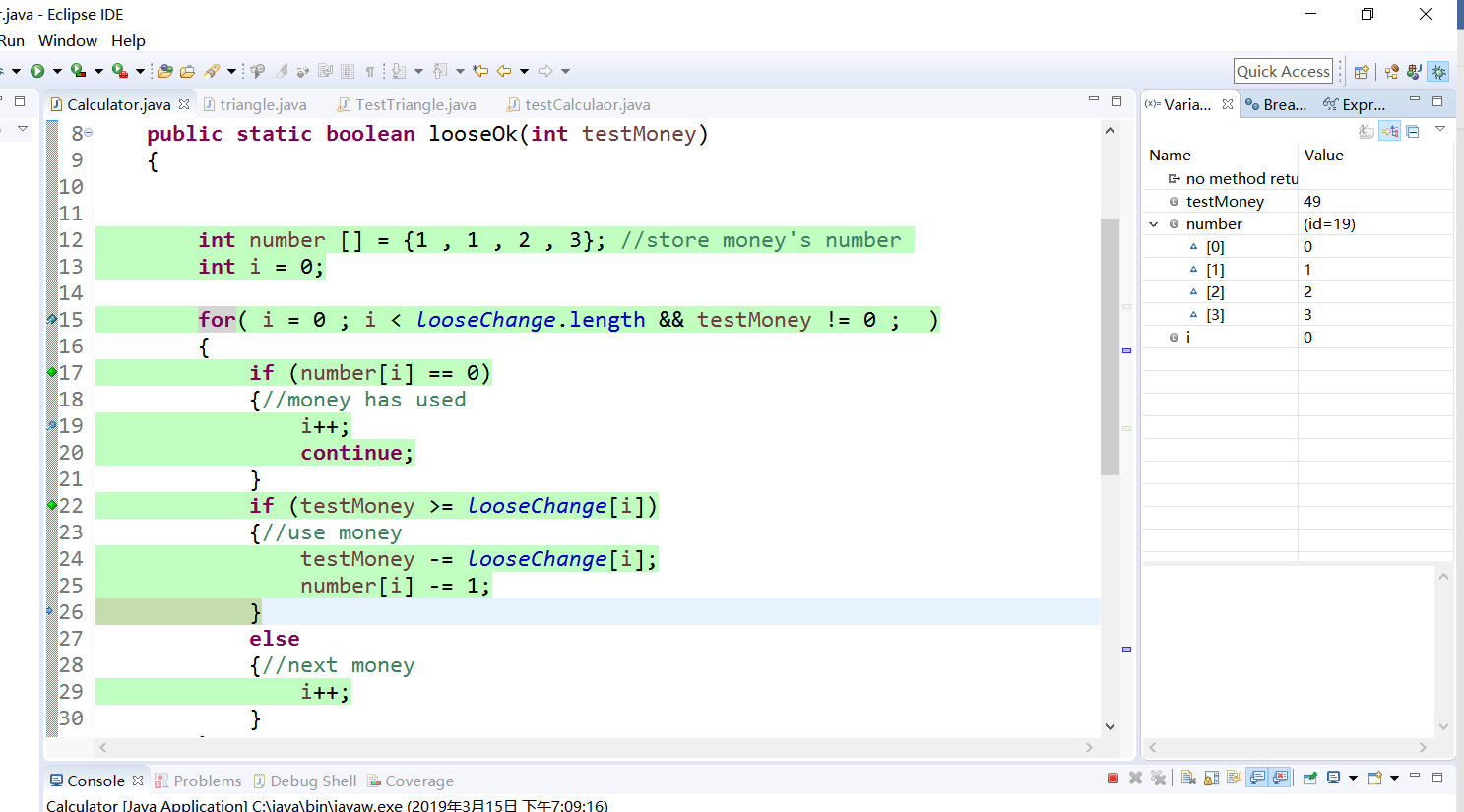
安装好后，点击coverage as即可进行覆盖测试



1. 调试分析（在实验过程中遇到的问题以及如何解决）

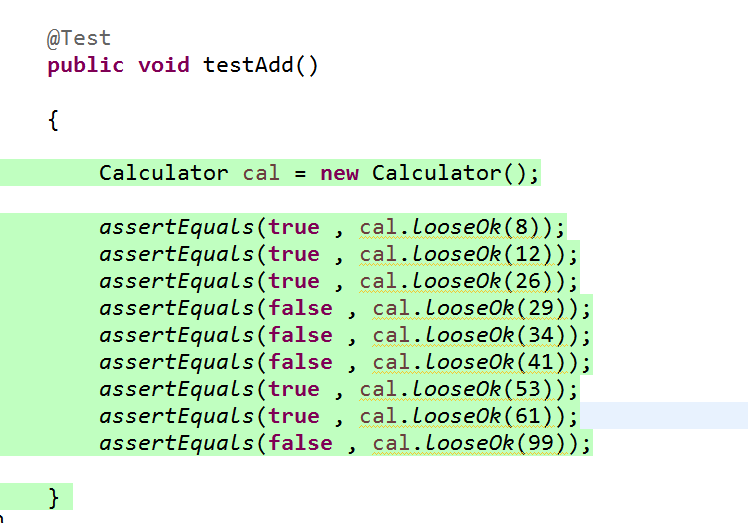
在实验过程中遇到一些问题，通过eclipse IDE的断点调试的功能，输出中间变量，发现程序中的bug

其中的一个bug是对于需要两张相同面额的找零存在问题，在经过检查后发现是判断条件有误，在纠正bug 后，程序结果正确

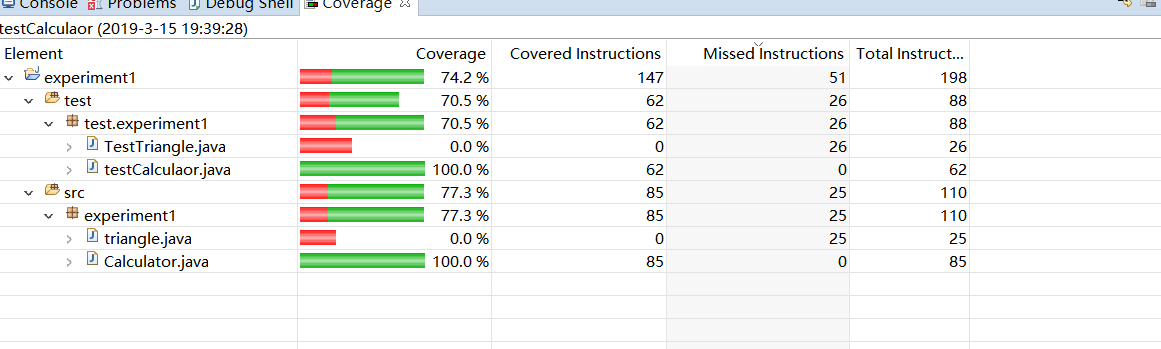


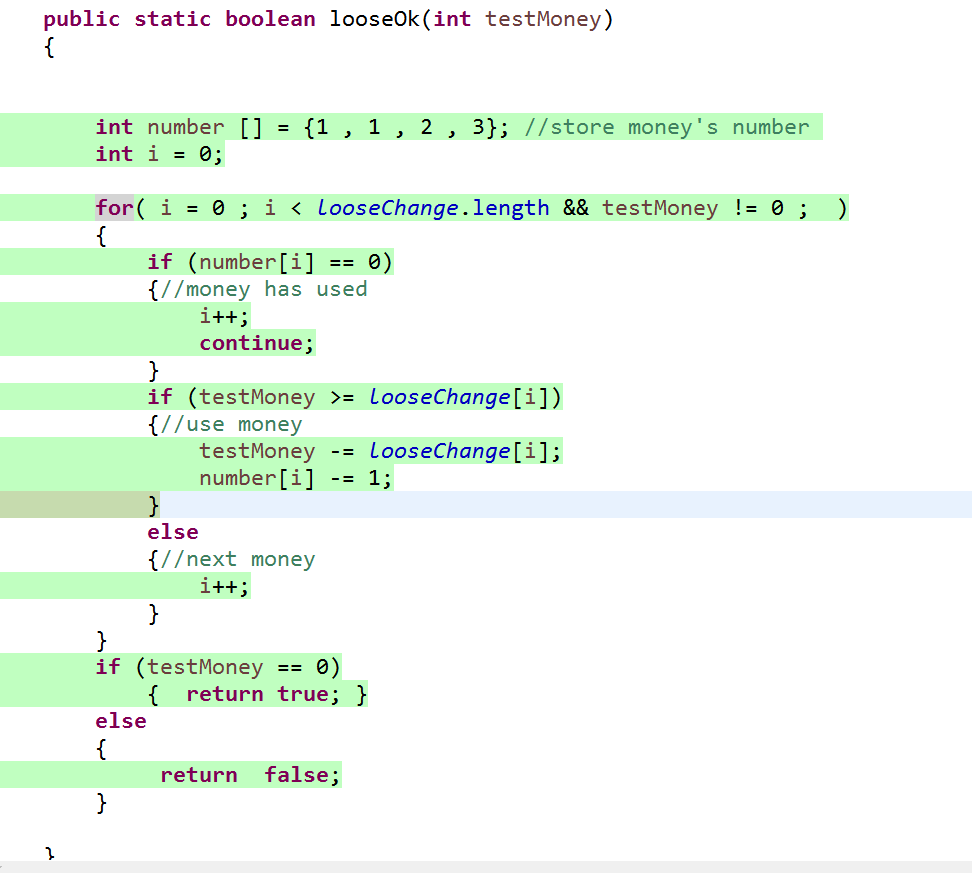
1. 测试结果（描述输入和输出）

我们采用的测试样例如下，所有样例均通过



测试的覆盖报告如下，覆盖图如下





在项目中我写了四个程序，一个是计算找零的，一个计算判断三角形的，还有对应的两个测试程序，

对于testCalculator覆盖率是100%，对于Calculator，测试函数全部覆盖，没覆盖到的是无关的的部分。

1. 总结

通过这次实验，首先分析数据特征，实现了判断找零的算法，然后使用工具junit和Eclemma进行了测试，分析了测试的覆盖率。通过这次实验直观的加深了自己对软件测试的理解，尤其是Eclemma对于覆盖率的分析，很好的指导了测试样例的设计。加深了对于软件测试的理解。