****

**软件测试技术第一次实验报告**



**学 院 软件学院**

**专 业 软件工程**

**年 级 16**

**姓 名 楼林**

**2019年 3 月 12 日**

# 软件测试技术第一次实验报告

## 需求分析

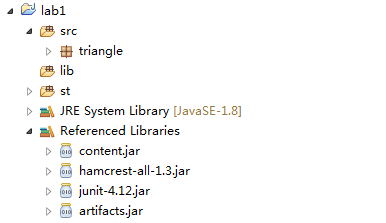
1. 在Eclipse中安装Junit(4.12)和Hamcret(1.3)。
2. 在Eclipse中安装Eclemma。
3. 为如下问题设计一个Java程序，并用JUnit进行测试。

问题描述：现有50元纸币1张，20元纸币1张，5元纸币2张，1元硬币3个。问：给定一个任意的金额，能否用现有的钱币凑出此金额。

## 概要设计

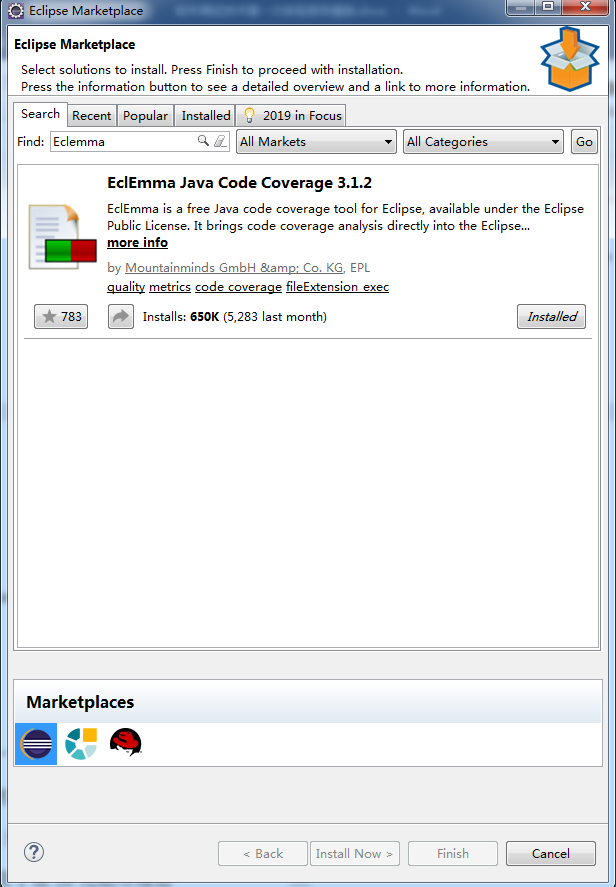
### 安装JUnit和Hamcret

首先在Eclipse中新建一个项目命名为lab1，在项目下新建一个文件夹lab，将JUnit和Hamcret的jar包拷贝到此文件夹下，然后右键Bulid Path -> Add to Build Path，这样就可以在项目中使用这两个jar包了。



### 安装Eclemma

在Eclipse的菜单栏点击 Help -> Eclipse Marketspace，在Find搜索框内输入Eclemma点击搜索，点击安装即可。



### 源程序的设计

对此问题进行观察之后，我发现这个问题有一个很大的特点，就是小数额的钱币的总和一定小于略大于它的钱币。例如：3个1元硬币相加为3元，小于5元纸币；3个1元硬币和2张5元纸币相加为13元，小于20元纸币；3个1元硬币和2张5元纸币以及1张20元纸币相加为33元，小于50元纸币。

由此我认为这个问题可以用**贪心算法**来解决。大概的思路为：按照50元，20元，5元，1元的先后顺序依次去对给定的金额进行取余操作直到金额为0。最后，如果需要的钱币数超过现有的钱币数，则认为不能通过现有的钱币凑出给定的金额。

### 测试程序的设计

使用注解@Parameters来标识一个用户自定义的方法，提供多组测试用例输入。然后用断言assertTrue来判断所测试方法的返回值（boolean类型）。

## 详细设计

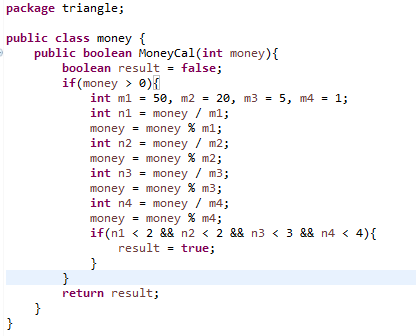
### 源程序的详细设计

首先，定义一个返回值为boolean类型的方法，参数为1个int值，也就是给定的那个金额，命名为money。

然后定义4种钱币，以及钱币的数量，均为int类型，并初始化。定义返回值（boolean），并初始化为false。

然后，依次用50,20,5,1对money进行取余操作，记录每种钱币需要的数量，如果每种钱币需要的数量小于现有的数量，就表示能凑出给定的金额，返回true。

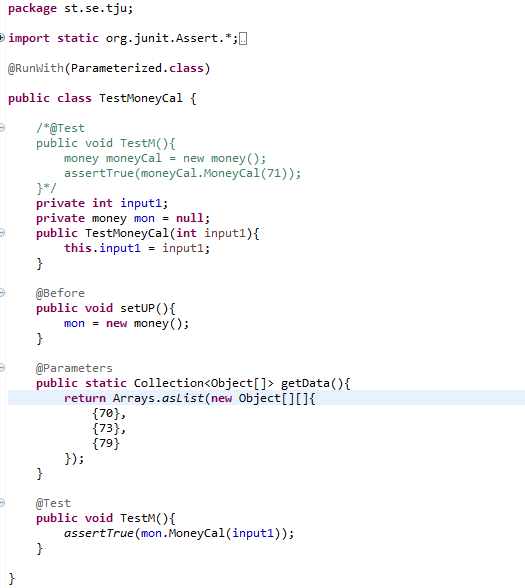
具体代码实现如下：



### 测试程序的详细设计

首先自定义一个方法，在每次测试程序开始前创建一个对象，然后通过一个数组和money类的变量来初始化当前对象。通过@Parameters方法返回的集合中的每一个值，JUnit都会创建一个新的测试类实例。

具体代码实现如下：



## 调试分析

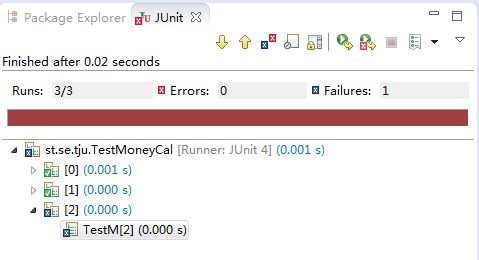
起初我认为应该用一个while循环来进行对给定的金额money进行多次循环取余，后来我发现其实这是没有必要的，因为最后的钱币是1元，最后的money值肯定为0，也就是说循环只会执行一遍，那么循环就没有必要了。除了这个小问题之外，还没有发现其他bug。

## 测试结果

### JUnit测试

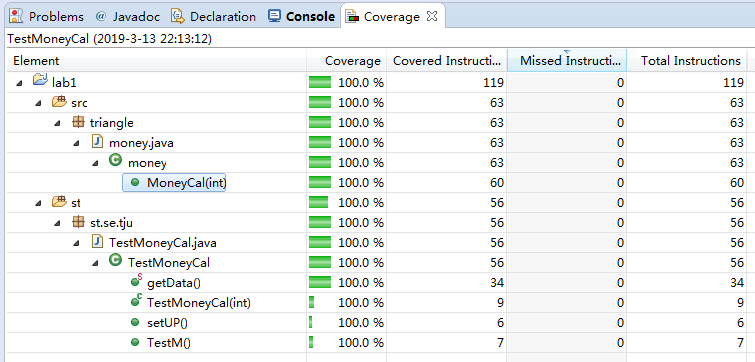
这里我采用了3组测试样例，分别是70,73,79，预期结果应该是前两个测试成功，最后一个测试失败。

实际测试结果如下：



### 覆盖性测试

实际测试结果如下：



## 总结

此次实验让我对利用JUnit和Eclemma进行软件测试有了一个大概的理解，同时也掌握了其基本的应用，可以说这是一次让我入门软件测试的实验。