****

**软件测试技术第一次实验报告**



**学 院 智能与计算学部**

**专 业 软件工程**

**年 级 2016**

**姓 名 傅淏**

**2019年 3 月 15 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）

1. Install Junit (4.12), Hamcrest (1.3) with Eclipse

2. Install Eclemma with Eclipse

3. Write a java program for the coin problem and test the program with Junit.

a) Description of coin problem:

There is one 50 yuan, one 20 yuan, two 5-yuan bills and three 1-yuan coins in your pocket. Write a program to find out whether you can take out a given number (x) yuan.

1. 概要设计（简单描述设计思路，配合UML图）

设计思路：将给定的数值作为输入值并与最大的钞票面值作比较，若大于最大的钞票面值就减去该面值数，并将所用钞票数量减一。循环至给定数值恰好减为0或者所有钞票用完给定数值仍有剩余或给定数值仍有剩余但已无法使用剩余钞票减为0，即可判断出给定数值能否“Take Out”。

流程图：

Start

fifty=1, twenty=1, five=2, one=3;

Y

num==0?

改变钞票面额

N

fifty>0 && num>=50

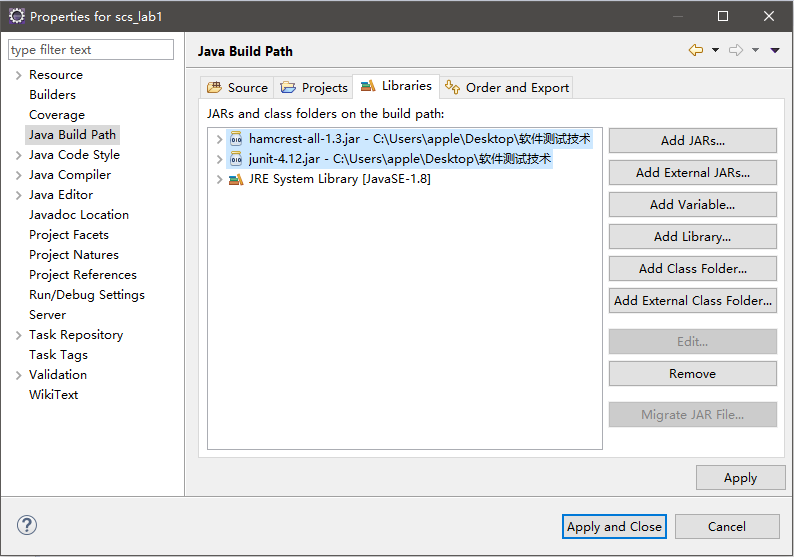
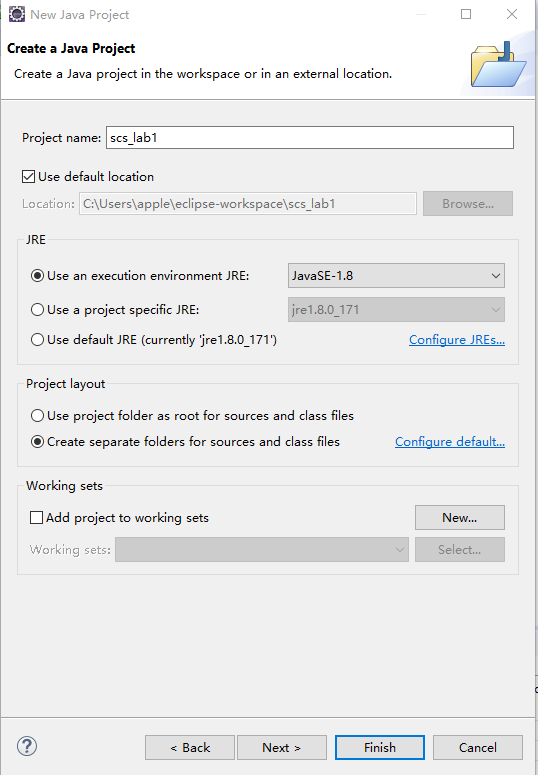
num-=50, fifty--

N

End

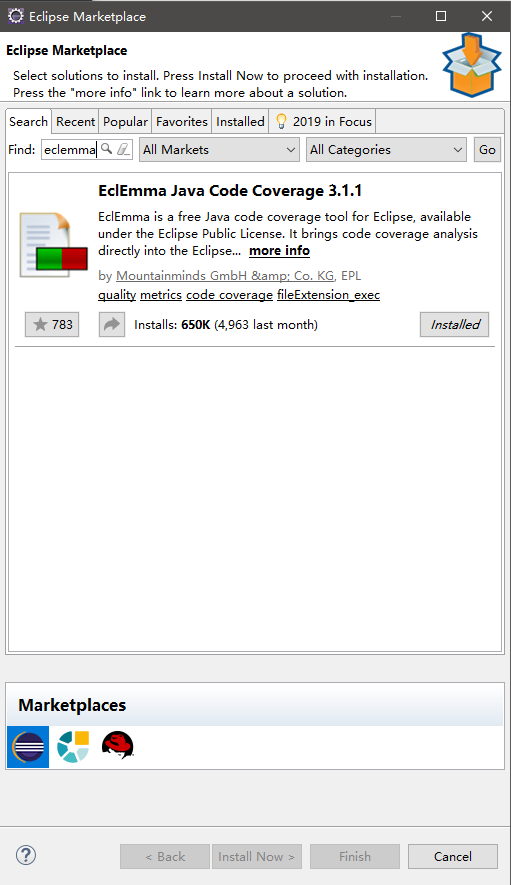
1. 详细设计（详细描述具体如何实现，附代码及说明）

1、新建Java项目，并将Junit和Hamcrest的jar包导入；



2、在Eclipse的Marketplace里面下载EclEmma，安装完毕出现测试选项；





3、编写程序；

**package** scs\_lab1;

**public** **class** MoneyProblem

{

//Judge whether the given number can be taken out

**public** **static** String Money (**int** num)

{

//Initial the number of paper money

**int** fifty = 1, twenty = 1, five = 2, one = 3;

//Subtract the number according to the money

**for** (**int** i = 1; i < 8; i++)

{

//The number is/comes to 0

**if** (num == 0)

**return** "Yes";

//Money is available and number is larger

**else** **if** (fifty > 0 && num >= 50)

{

fifty --;

num -= 50;

**continue**;

}

//Fifty is 0, jump to twenty

**else** **if** (twenty > 0 && num >= 20)

{

twenty --;

num -= 20;

**continue**;

}

//Twenty is 0, jump to five

**else** **if** (five > 0 && num >= 5)

{

five --;

num -= 5;

**continue**;

}

//Five is 0, jump to one

**else** **if** (one > 0 && num >= 1)

{

one --;

num -= 1;

**continue**;

}

//All money is used and still has number

**else** **if** (num > 0)

**return** "No";

}

//Final confirmation

**if** (num == 0)

**return** "Yes";

**else**

**return** "No";

}

}

**import** **static** org.junit.Assert.\*;

**import** java.util.Arrays;

**import** java.util.Collection;

**import** org.junit.Test;

**import** org.junit.runner.RunWith;

**import** org.junit.runners.Parameterized;

**import** org.junit.runners.Parameterized.Parameters;

**import** scs\_lab1.MoneyProblem;

@RunWith (Parameterized.**class**)

**public** **class** TestMoneyProblem

{

**private** **int** a;

**private** String expected;

**public** TestMoneyProblem (**int** a, String expected)

{

**this**.a = a;

**this**.expected = expected;

}

@Parameters

**public** **static** Collection<Object[]> getData ()

{

**return** Arrays.*asList* (**new** Object[][]{

{-1, "No"},

{0, "Yes"},

{1, "Yes"},

{4, "No"},

{7, "Yes"},

{9, "No"},

{12, "Yes"},

{16, "No"},

{54, "No"},

{79, "No"},

{83, "Yes"},

{84, "No"}

});

}

@Test

**public** **void** test ()

{

*assertEquals* (**this**.expected, MoneyProblem.*Money* (a));

}

}

1. 调试分析（在实验过程中遇到的问题以及如何解决）

1、无法将Junit的test文件放入专门文件夹中，后来通过Eclipse的属性实现“Add the Test folder as source folder”，然后test文件即可顺利运行。

2、进行测试时无法实现全语句覆盖，后来添加上了初始数值为0和负数的情况，最终实现了语句的全覆盖。

1. 测试结果（描述输入和输出）

输入测试用例：

{-1, "No"},

{0, "Yes"},

{1, "Yes"},

{4, "No"},

{7, "Yes"},

{9, "No"},

{12, "Yes"},

{16, "No"},

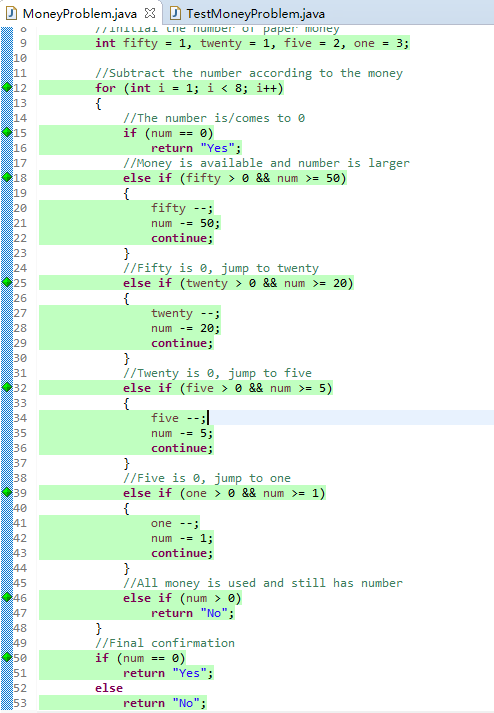
{54, "No"},

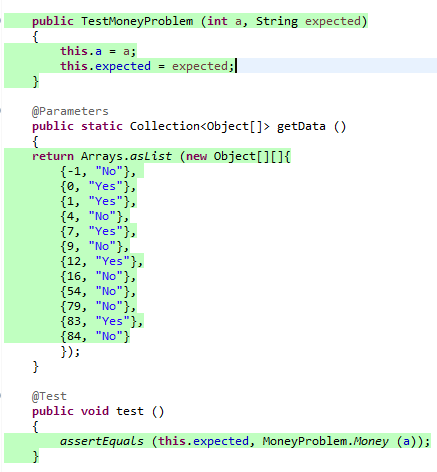
{79, "No"},

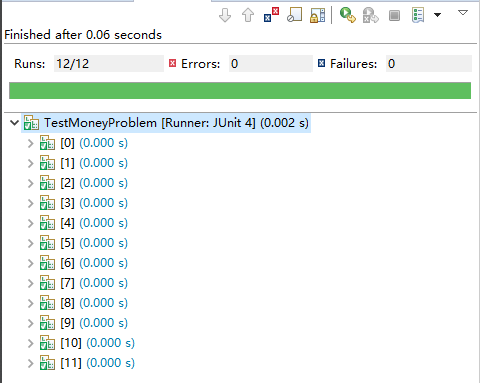
{83, "Yes"},

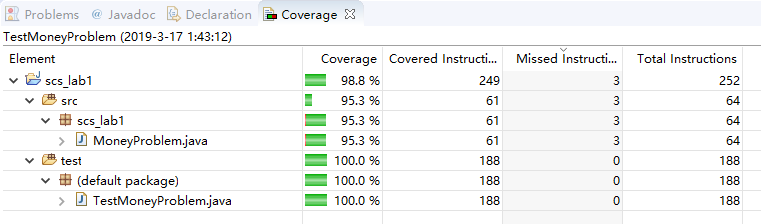
{84, "No"}

输出结果：









1. 总结

通过这次实验，我学会了使用Junit进行简单的程序测试，在测试过程中，掌握了使用EclEmma观察程序测试对每一条分支的覆盖的能力，对于测试中出现的问题也可以独立进行有效地解决。