****

**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 范立水**

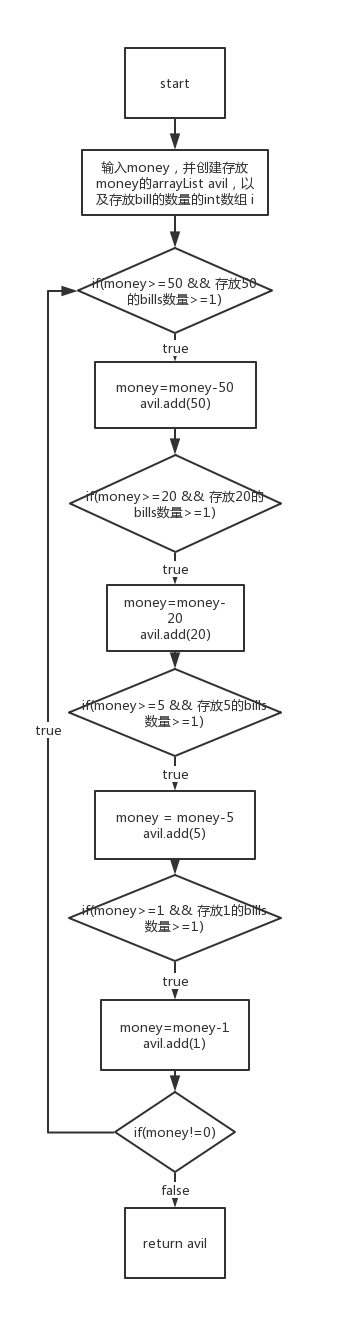
**2019年 3 月 12 日**

# 软件测试技术第一次实验报告

1. 需求分析（描述具体需求）
2. 安装junit，hamcrest,eclemma
3. 完成课堂所留任务即，有50，20元各一张钞票，5元钞票两张，1元钞票3张，给定一个number，然后找出所用的钞票组合
4. 检查代码并使用junit测试
5. 检查覆盖率
6. 概要设计（简单描述设计思路，配合UML图）

1. 将junit，hamcrest 的jar包导入即可，idea可以直接查看代码的覆盖率，不需要eclemma

2. 使用贪婪算法，建立arraylist来存储组成的策略，并使用int数组来存储各个数值钞票的数量，从大到小进行匹配，即先匹配50，20，5，1。不断循环，如果出现死循环，说明输入值错误，则报错，并将arraylist清空。

w

1. 详细设计（详细描述具体如何实现，附代码及说明）

package scs.Triangle;  
  
import java.util.ArrayList;

public class Triangle {  
 public static ArrayList<Integer> getMoney(int money){  
 int[] avil = {1,1,2,3}; //建立数组存储钞票数量  
 ArrayList<Integer> compose = new ArrayList<>(); //建立ArrayList存储策略  
 while (money>=0) //不断循环  
 { if(money==0)  
 {  
 return compose;  
 }  
 else if(money>=50&&avil[0]>=1)  
 {  
 money = money-50;  
 compose.add(50);  
 avil[0]--;  
 }  
 else if(money>=20&&money<=50&&avil[1]>=1)  
 {  
 money = money-20;  
 compose.add(20);  
 avil[1]--;  
 }  
 else if(money>=5&&avil[2]>=1)  
 {  
 money = money-5;  
 compose.add(5);  
 avil[2]--;  
 }  
 else if(money>=1&avil[3]>=1)  
 {  
 money = money-1;  
 compose.add(1);  
 avil[3]--;  
 }else { //如果出现其他情况，  
 System.*out*.println("input error");  
 compose.clear();  
 break;  
 }  
 }  
 return compose;  
 }

}

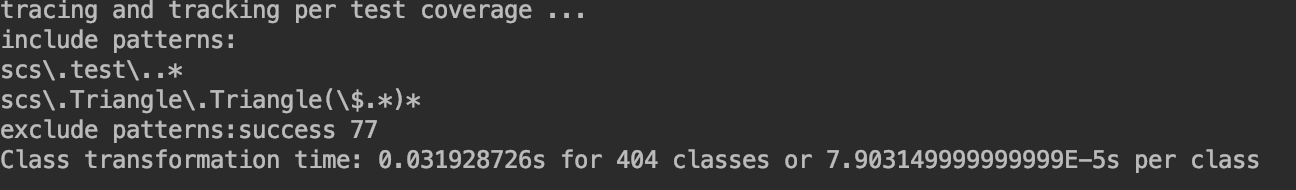
1. 调试分析（在实验过程中遇到的问题以及如何解决）

程序不够健壮，在输入9，14，>=84后，程序进入死循环，无法跳出，于是加入判断是否进入死循环，然后输出“input error” 提示用户重新输入

1. 测试结果（描述输入和输出）

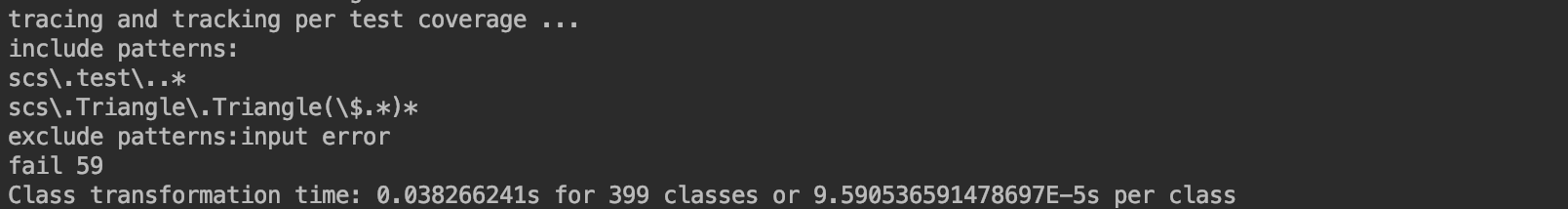
当输入77时，发现除了最后解决异常的判断没有被cover，其他该被cover的代码段皆被cover了





当输入59时，发现第13，23-25行代码段没有被cover，其余都被cover，与预想结果一致。





1. 总结
2. 不使用main进行测试，一开始还感觉不是太方便，可能是思维已经太久没有变过了，但是当开始使用junit时，发现junit比main要好用出很多，比如junit保证了测试代码与软件代码的分离，以保证代码不会过于混乱，而且使用junit测试起来，还很牢靠，保证了程序员可以访问类接口的成员变量和方法，而使用main无法达到这种效果。
3. 使用junit对代码进行单元测试，增强了代码的健壮性，并且通过覆盖率的查看，可以帮助我们在运行代码时，了解代码中哪一部分出现了问题，并进行改正。