**数学与信息技术学院**

实 验 报 告

**实验课程名称： 软件工程**

**实验项目名称：** 软件测试

**专 业 班 级： 21计科接本2班**

**学 号：**  202100431214

**姓 名：**  刘思雨

**指 导 教 师：**  霍丽娜

**实验目的：**

1.理解软件测试的特点及其主要方法

2.掌握黑盒测试的主要方法

3.掌握JUint的使用方法

**实验内容：**

1.在Eclipse中使用Junit进行单元测试

参照：https://www.cnblogs.com/borter/p/9608649.html

2.在Idea中使用Jnit进行单元测试，

参照：https://www.jianshu.com/p/f953c4dd84d5

注意：第1题和第2题选其中之一做，根据自己电脑上的开发环境选择一个。

3.一个销售人员代销步枪，包括枪机（locks）、枪托（stocks）、枪管（barrels）,枪机45美元、枪托30美元、枪管25美元；销售每走访一个城市就要向厂家发一封电报，汇报销售数量，月末会发一封电报，当发送“枪机销售数量为-1”时结算。佣金计算方法为：销售总额在1000元以下部分（含1000元）佣金为10%;1000至1800之间部分佣金为15%，超过1800元部分佣金为20%，

佣金报表汇出销售的枪机、枪托、枪管总数量、总销售额及佣金额。

用任意一种熟悉的程序设计语言，编写一个应用程序，实现数据的输入、汇总结果的输出。用等价类划分、边界值分析法设计测试用例。

1. 根据下列程序代码，设计满足覆盖条件的测试用例。
2. 根据下列程序代码，画出其程序流图，求出环形复杂度，并确定独立路径，选择合适的测试用例，并对其进行测试

public static void SortNum(int numA, int numB)

{

int x = 0;

int y = 0;

while (numA--> 0)

{

if (numB == 0)

x = y +2;

else

if (numB == 1)

x = y + 10;

else

x = y +20;

}

}

1.

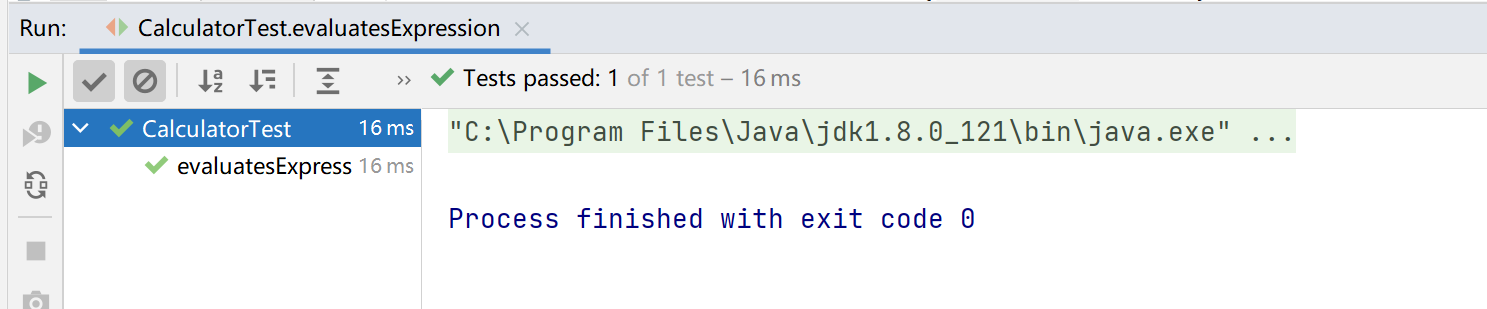
**实验步骤：**

public class Calculator {  
 public int evaluate(String expression){  
 int sum = 0;  
 for(String summand:expression.split("\\+"))  
 sum += Integer.*valueOf*(summand);  
 return sum;  
 }  
}

import org.junit.Test;  
import static junit.framework.TestCase.*assertEquals*;  
public class CalculatorTest {  
 @Test  
 public void evaluatesExpression(){  
 Calculator calculator = new Calculator();  
 int sum = calculator.evaluate("1+2+3");  
 *assertEquals*(6,sum);  
 }  
}

期望值与实际值一样

**实验结果:**



2.

**实验步骤：**

@Test  
public void evaluatesExpression(){  
 Calculator calculator = new Calculator();  
 int sum = calculator.evaluate("1+2+3");  
 *assertEquals*(6,sum);  
 *assertEquals*(false,true);  
}

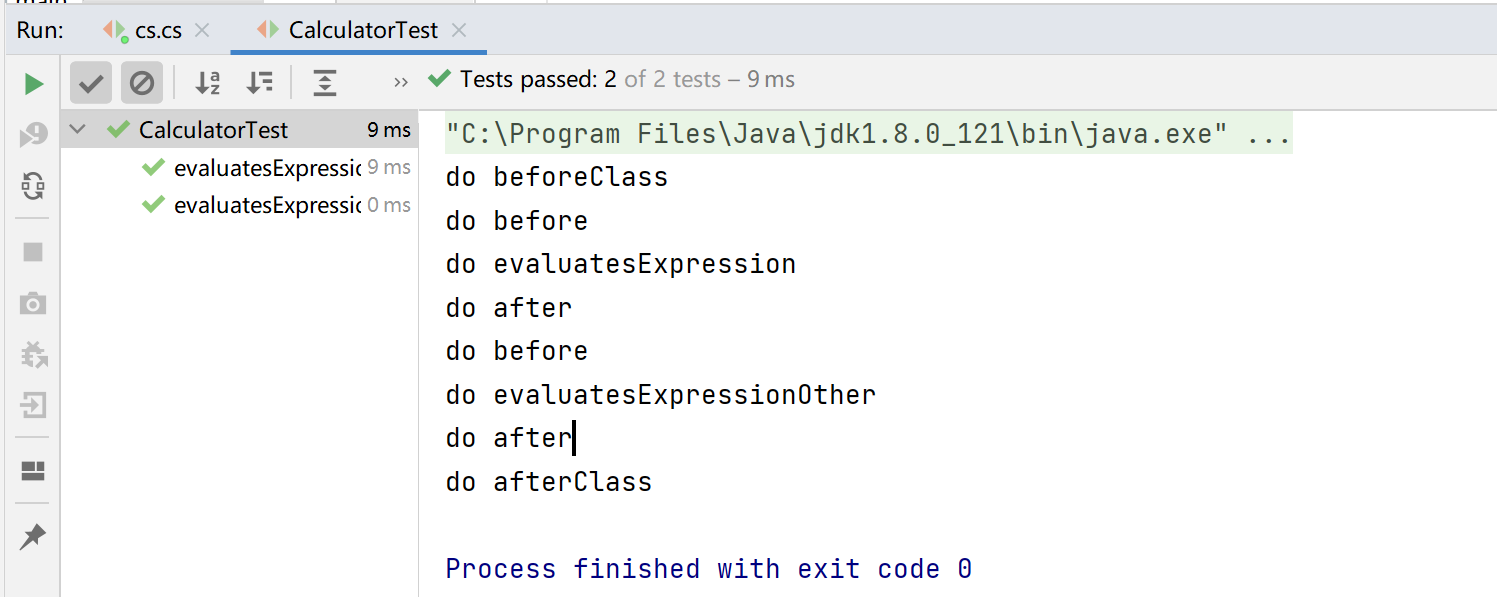
期望值与实际值不一样，报错

3.

**实验步骤：**

import org.junit.\*;  
import static junit.framework.TestCase.*assertEquals*;  
public class CalculatorTest {  
  
 @Before  
 *//每次测试方法执行前，该方法执行一次* public void before(){  
 System.*out*.println("do before");  
 }  
 @BeforeClass  
 *//仅执行一次，在测试一开始执行* public static void beforeClass(){  
 System.*out*.println("do beforeClass");  
 }  
  
  
 @Test  
 public void evaluatesExpression(){  
 System.*out*.println("do evaluatesExpression");  
 Calculator calculator = new Calculator();  
 int sum = calculator.evaluate("1+2+3");  
 *assertEquals*(6,sum);  
 *// assertEquals(false,true);* }  
  
 @Test  
 public void evaluatesExpressionOther(){  
 System.*out*.println("do evaluatesExpressionOther");  
 Calculator calculator = new Calculator();  
 int sum = calculator.evaluate("1+2+3+4");  
 *//对比可以是两个数据的值,也可以是两个数据的地址值。正确情况下,不会有显示,会正常执行流程  
 assertEquals*(10,sum);  
 }  
 @After  
 *//每次测试方法执行结束时，该方法执行一次* public void after(){  
 System.*out*.println("do after");  
 }  
 @AfterClass  
 *//仅执行一次，在测试结束执行* public static void afterClass(){  
 System.*out*.println("do afterClass");  
 }  
  
}

**实验结果:**

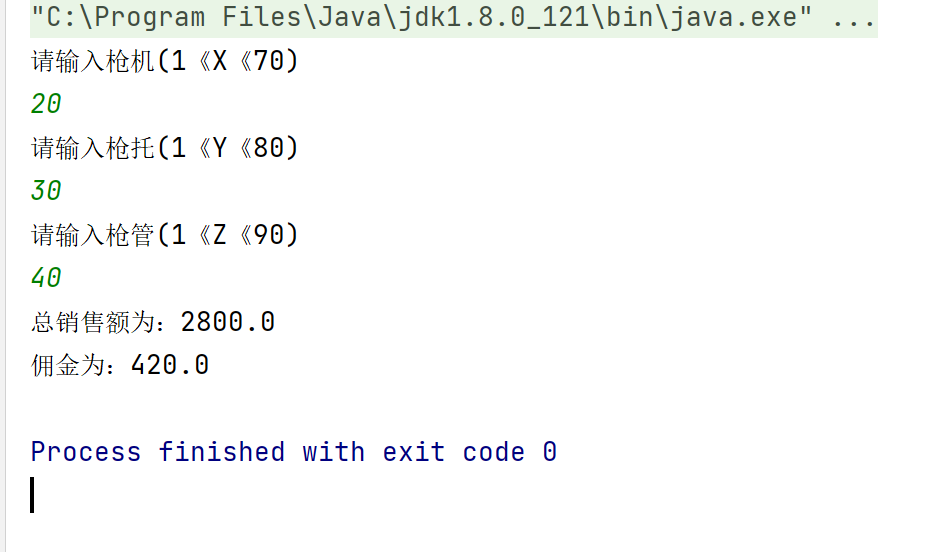


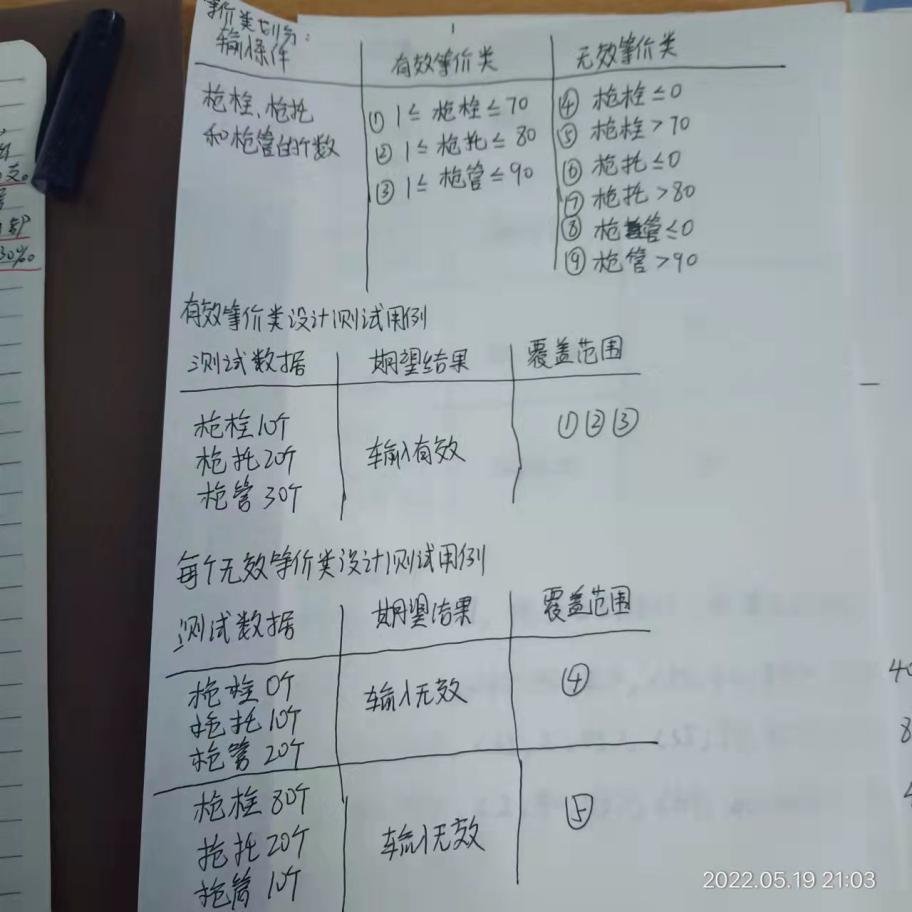
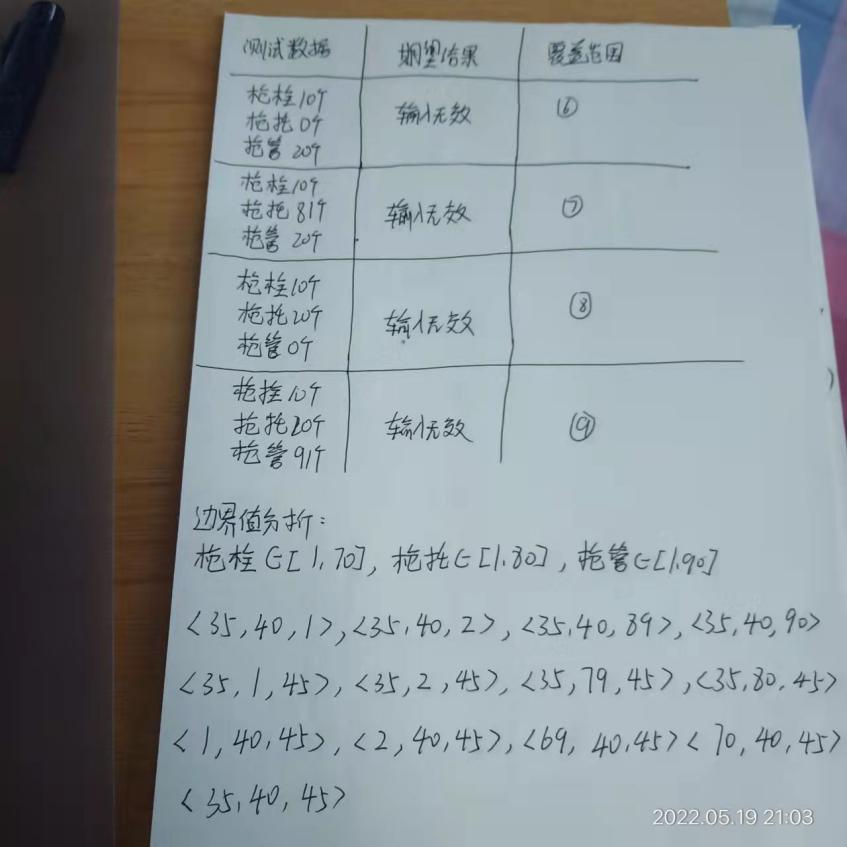
二．

**实验步骤：**

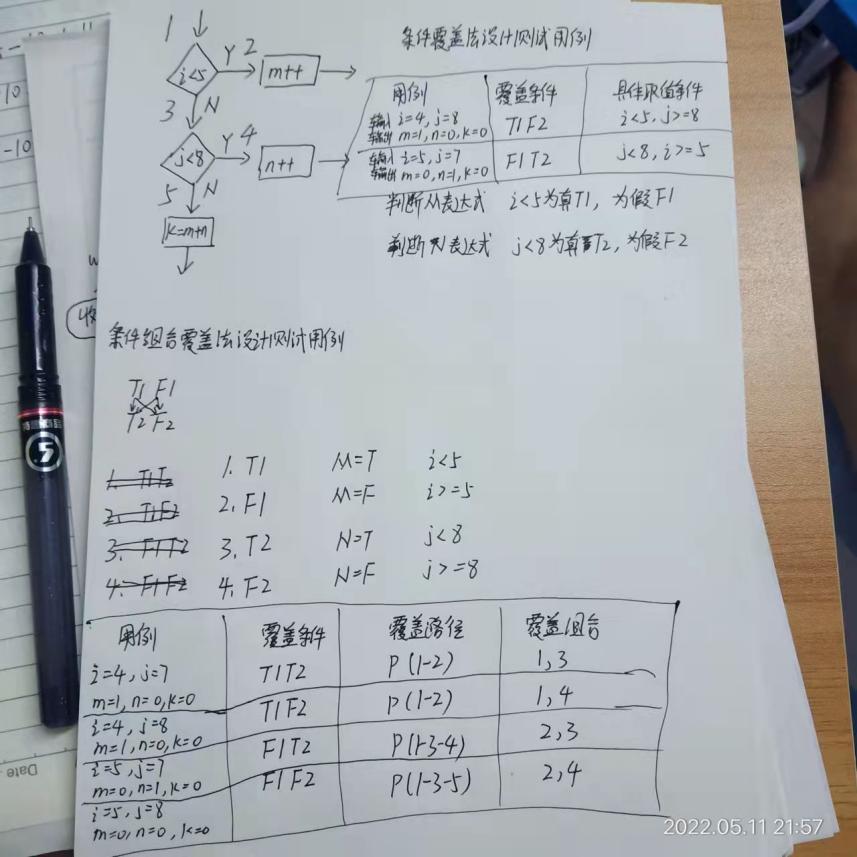
import java.util.Scanner;  
public class cs {  
 public static void main(String[] args) {  
 int a,b,c;  
 double sum = 0 ,money = 0;  
 boolean flag = true;  
 Scanner sc = new Scanner(System.*in*);  
  
 while(flag){  
 System.*out*.println("请输入枪机(1《X《70)");  
 a = sc.nextInt();  
 System.*out*.println("请输入枪托(1《Y《80)");  
 b = sc.nextInt();  
 System.*out*.println("请输入枪管(1《Z《90)");  
 c = sc.nextInt();  
 if((a<=70&&a>=1)&&(b<=80&&b>=1)&&(c<=90&&c>=1)){  
 sum=(a\*45)+(b\*30)+(c\*25);  
 if(sum<=1000){  
 money=sum\*0.1;  
 }if(sum<=1800&&sum>1000){  
 money=100+(sum-1000)\*0.15;  
 }if(sum>1800){  
 money=100+120+(sum-1800)\*0.2;  
 }  
  
 flag=false;  
 }else {  
 System.*out*.println("输入数目不正确");  
 flag=true;  
 }  
  
 }  
 System.*out*.println("总销售额为："+sum);  
 System.*out*.println("佣金为："+money);  
 }

}  
**实验结果:**

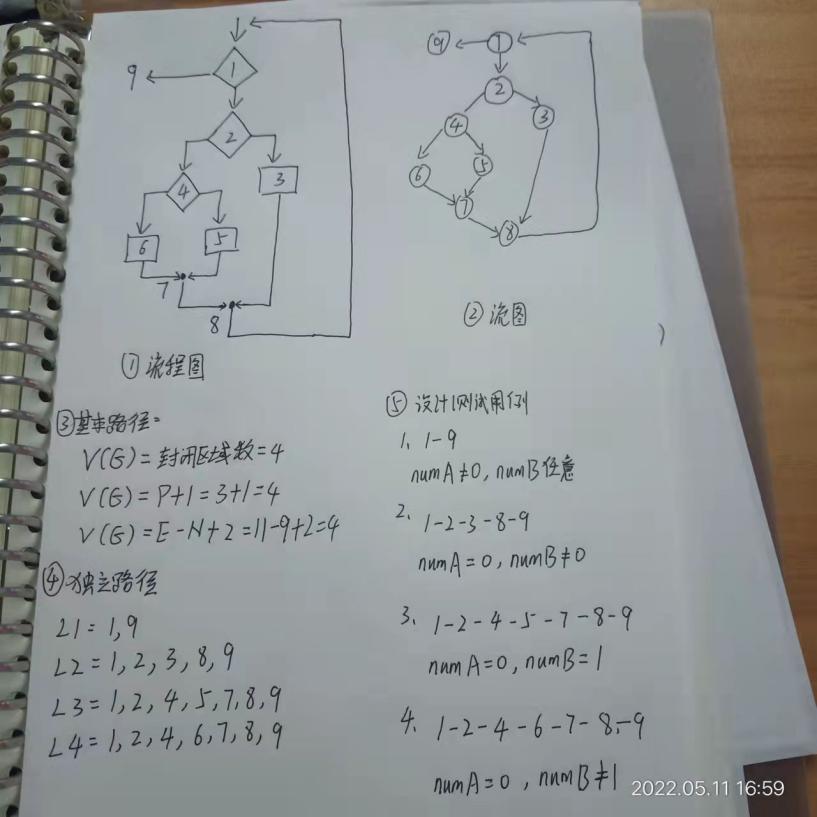




三.



四．



**实验总结：**

**我学会了@Test注解的使用。@Before在方法前使用,@BeforeClass仅执行一次，测试一开始,@After在方法后使用,@AfterClass仅执行一次，测试结束后，注解使用的分别用法。**

**软件测试：黑盒测试和白盒测试**