# 四则运算单元测试报告

**1.源码**

package 四则运算;

import java.util.Random;  
import java.util.Scanner;

import javax.print.DocFlavor.INPUT\_STREAM;

public class producer {  
public int t\_answer=0;//记录答对题目数  
public boolean com=false;//出题成功判断标志

//答案校验模块  
void check(int answer)  
{     
 @SuppressWarnings("resource")  
 Scanner scanner=new Scanner(System.in);  
    int u\_answer;  
    System.out.println("请回答：");  
    while(true){  
    try{  
    u\_answer=scanner.nextInt();  
    break;  
    }  
    catch (Exception e) {  
     System.out.println("您输入的不是正整数，请重新输入");   
  scanner.next();  
 }  
    }  
 if(u\_answer==answer)  
 {     
  System.out.println("答案正确");  
  this.t\_answer++;  
 }  
 else   
 {  
 System.out.println("答案错误");   
 }  
}

//加法式产生  
int p\_add(int left,int right)  
{     
 int answer;  
 answer=left+right;  
 if(answer<=100)  
 {  
 System.out.println(left+"+"+right+"=");  
 this.com=true;  
 }  
 return answer;  
}

//减法式产生  
int p\_sub(int left,int right)  
{  
 int answer;  
 answer=left-right;  
    if(answer>=0)  
 {  
 System.out.println(left+"-"+right+"=");  
 this.com=true;  
 }  
 return answer;  
}

//乘法式产生  
int  p\_mul(int left,int right)  
{  
 int answer;  
 answer=left\*right;  
 if(answer<=100)  
 {  
 System.out.println(left+"\*"+right+"=");  
 this.com=true;  
 }  
 return answer;  
}

//除法式产生  
int p\_div(int left,int right)  
{  
 int answer;  
 int success;//余数判断标志  
 success=left%right;  
 answer=left/right;  
    if(right!=0&success==0)  
    {  
 System.out.println(left+"/"+right+"=");  
 this.com=true;  
    }  
 return answer;  
}

public static void main(String arg[])  
{  
 int answer=0;//存储答案  
 int left;//产生式左部  
 int right;//产生式右部  
 producer producer=new producer();  
 @SuppressWarnings("resource")  
 Scanner scanner=new Scanner(System.in);  
 Random random=new Random();  
 System.out.println("请输入要出的题目个数：");  
 int p\_number=0;  
 while(true){  
 try {  
  p\_number=scanner.nextInt();/\*输入的出题数目\*/  
  break;  
  } catch (Exception e) {  
  System.out.println("您输入的不是正整数，请重新输入");   
  scanner.next();  
 }  
 }  
 for(int i=0;i<p\_number;i++)  
 {  
  int r=random.nextInt(4)+1;/\*随机式子类型判断\*/  
  //System.out.println(r);  
  while(!(producer.com)){  
  left=random.nextInt(101);  
  right=random.nextInt(101);  
  //System.out.println("("+left+","+right+")");  
  switch (r) {  
  case 1:  
   /\*产生加法式\*/  
   answer=producer.p\_add(left,right);  
   if(producer.com)  
   producer.check(answer);  
   break;  
  case 2:  
   /\*产生减法式\*/  
   answer=producer.p\_sub(left,right);  
   if(producer.com)  
   producer.check(answer);  
   break;  
  case 3:  
   /\*产生乘法式\*/  
   answer=producer.p\_mul(left,right);  
   if(producer.com)  
   producer.check(answer);  
   break;  
  case 4:  
   /\*产生除法式\*/  
   answer=producer.p\_div(left,right);  
   if(producer.com)  
   producer.check(answer);  
   break;  
    }  
  }  
  producer.com=false;  
 }  
 System.out.println("您的得分是"+producer.t\_answer\*100/p\_number);/\*计算并打印得分\*/  
}  
}

**2.测试用例设计**

**加法测试用例**

由于产生和为100内的加法

设计正常测试用例

2+3=5

边界测试用例

55+45=100

56+45=101

期望输出：

加法测试结果如下  
2+3=  
55+45=

**减法测试用例**

100之内的减法，且差不可以小于0

设计正常测试用例

3-2=1

边界测试用例

3-3=0

1-2=-1

期望输出

减法测试结果如下  
3-2=  
3-3=

**乘法测试用例**

乘积小于等于100

设计正常测试用例

2\*3=6

0乘用例

0\*3=0

边界测试用例

10\*10=100

11\*10=110

期望输出

乘法测试结果如下  
2\*3=  
0\*3=  
10\*10=

**除法测试用例**

100以内除法，除数不可为0且必须整除

正常测试用例

6/2=3

不可整除用例

5/3=1

被除数为0用例

0/3=0

由于除数生成有0值判断，会提示异常，且代码内已避免此情况，此处不做处理

期望输出

除法测试结果如下  
6/3=  
0/3=

**核对测试用例**

核对两次答案1

分别输入A,+,1,0

期望输出

请回答：  
A  
您输入的不是正整数，请重新输入  
+  
您输入的不是正整数，请重新输入  
1  
答案正确  
请回答：  
0  
答案错误

**3.测试框架介绍，安装过程**

**junit框架介绍**

JUnit是一个Java语言的单元测试框架。它由Kent Beck和Erich Gamma建立，逐渐成为源于Kent Beck的sUnit的xUnit家族中最为成功的一个。 JUnit有它自己的JUnit扩展生态圈。多数Java的开发环境都已经集成了JUnit作为单元测试的工具。 [1]

JUnit是由 Erich Gamma 和 Kent Beck 编写的一个回归测试框架（regression testing framework）。Junit测试是程序员测试，即所谓白盒测试，因为程序员知道被测试的软件如何（How）完成功能和完成什么样（What）的功能。Junit是一套框架，继承TestCase类，就可以用Junit进行自动测试了。

特性

JUnit是一个开放源代码的Java测试框架，用于编写和运行可重复的测试。他是用于[单元测试](https://baike.baidu.com/item/%E5%8D%95%E5%85%83%E6%B5%8B%E8%AF%95" \t "_blank)框架体系xUnit的一个实例（用于java语言）。它包括以下特性：

1、用于测试期望结果的断言（Assertion）

2、用于共享共同测试数据的测试工具

3、用于方便的组织和运行测试的测试套件

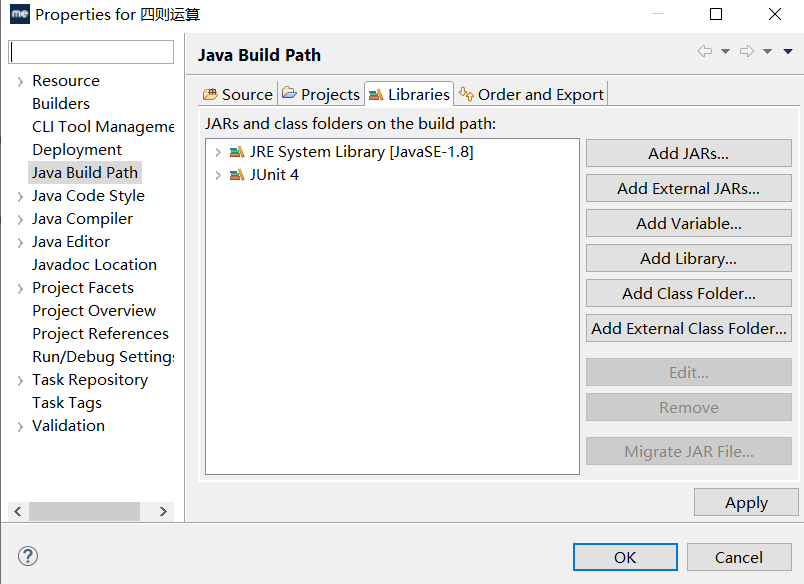
4、图形和文本的测试运行器

安装过程

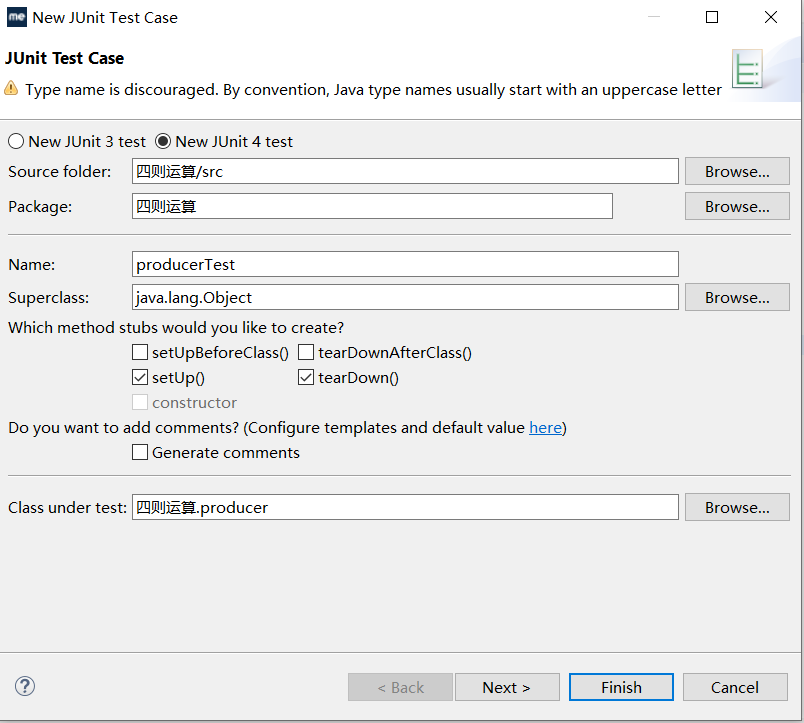
测试过程中使用的junit由myeclspe自带只需加入library库即可

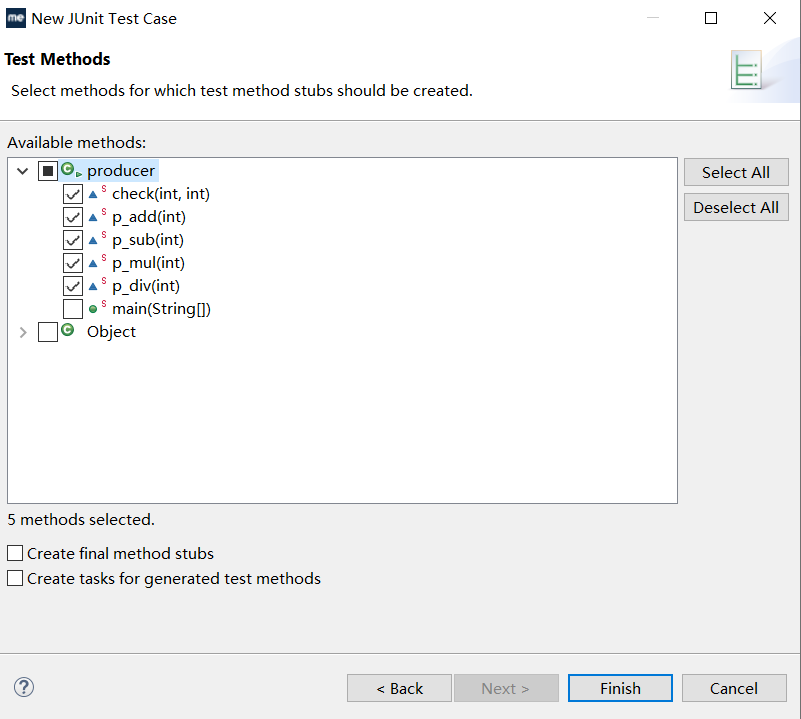
操作流程如下

junit包的增加



测试类的创建





**4.测试代码**

package 四则运算;

import static org.junit.Assert.\*;

import org.junit.After;  
import org.junit.Before;  
import org.junit.Test;

public class producerTest {  
 producer producer=new producer();  
 @Before  
 public void setUp() throws Exception {  
 }

 @After  
 public void tearDown() throws Exception {  
 }

 @Test  
 public void testCheck() {  
  producer.check(1);  
  producer.check(1);  
 }

 @Test  
 public void testP\_add() {  
  System.out.println("加法测试结果如下");  
  assertEquals(5,producer.p\_add(2,3));  
  assertEquals(100,producer.p\_add(55,45));  
  assertEquals(101,producer.p\_add(56,45));  
 }

 @Test  
 public void testP\_sub() {  
  System.out.println("减法测试结果如下");  
  assertEquals(1,producer.p\_sub(3,2));  
  assertEquals(0,producer.p\_sub(3,3));  
  assertEquals(-1,producer.p\_sub(1,2));  
 }

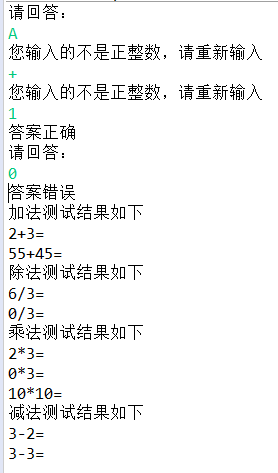
 @Test  
 public void testP\_mul() {  
  System.out.println("乘法测试结果如下");  
  assertEquals(6,producer.p\_mul(2,3));  
  assertEquals(0,producer.p\_mul(0,3));  
  assertEquals(100,producer.p\_mul(10,10));  
  assertEquals(110,producer.p\_mul(11,10));  
 }

 @Test  
 public void testP\_div() {  
  System.out.println("除法测试结果如下");  
  assertEquals(2,producer.p\_div(6,3));  
  assertEquals(1,producer.p\_div(5,3));  
  assertEquals(0,producer.p\_div(0,3));  
 }

}

**5.测试结果**

**测试结果如下图，同预期结果相符，不合约束的测试用例并未产生实际输出结果**

****