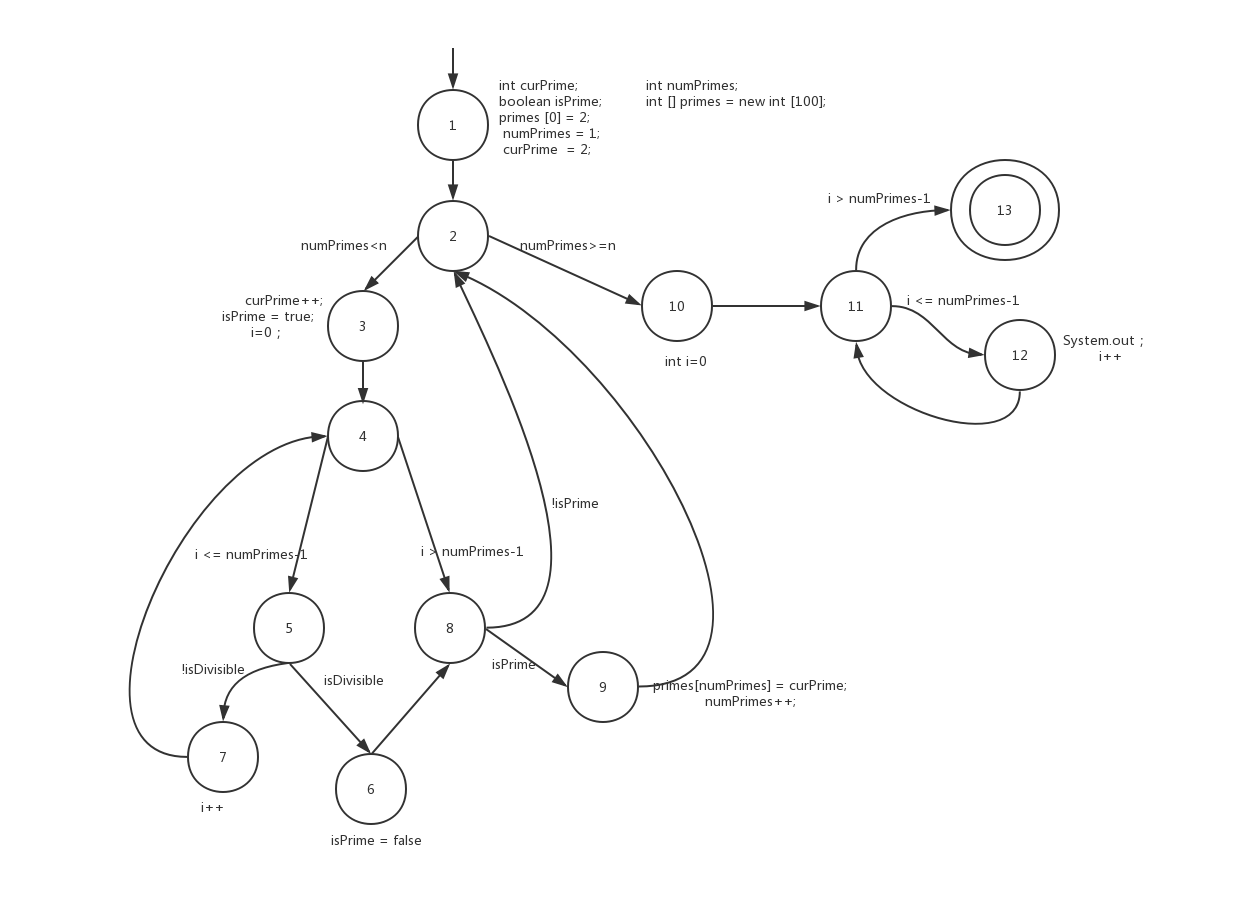
曾晓东 3016218101

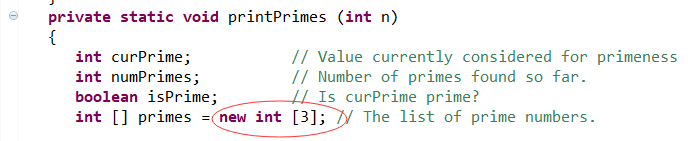
Homework3：

a)控制流图（process on画图）

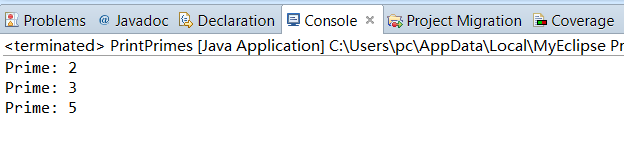


b)n=3测试的数据量比n=5要小

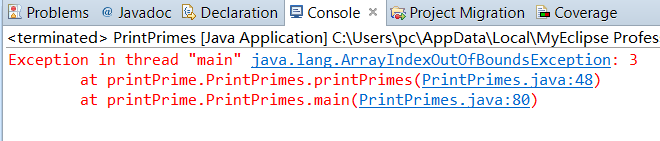
将代码中存储prime的数组的大小改成3



测试t1 n=3的时候没有错误，正常输出



测试t2 n=5的时候抛出数组越界的异常



c)连接while语句的起点和for语句的边，且不进入循环体—即1-2-12

当n=1的时候，numPrimes=n，故不会进入循环，且经过2-12边，故满足条件

d)结点覆盖：{1,2,3,4,5,6,7,8,9,10,11,12,13}

边覆盖：

{(1,2),(2,3),(2,10),(3,4),(4,5)(4,8),(5,6) ,(5,7) ,(6,8) ,(7,4) ,(8,2) ,(8,9) ,(9,2) ,(10,11) ,(11,12) ,(11,13) ,(12,11) }

主路径覆盖：

(1,2,3,4,5,6,8,9), (1,2,3,4,5,7), (1,2,3,4,8,9), (1,2,10,11,12),

(1,2,10,11,13),

(2,3,4,5,6,8,2), (2,3,4,5,6,8,9,2),(2,3,4,8,2), (2,3,4,8,9,2),

(3,4,5,6,8,2,3), (3,4,5,6,8,2,10,11,12), (3,4,5,6,8,2,10,11,13),

(3,4,5,6,8,9,2,3), (3,4,5,6,8,9,2,10,11,12), (3,4,5,6,8,9,2,10,11,13),

(3,4,8,2,3), (3,4,8,2,10,11,12), (3,4,8,2,10,11,13), (3,4,8,9,2,3),

(3,4,8,9,2,10,11,12), (3,4,8,9,2,10,11,13),

(4,5,6,8,2,3,4), (4,5,6,8,9,2,3,4), (4,5,7,4), (4,8,2,3,4), (4,8,9,2,3,4),

(5,6,8,2,3,4,5), (5,6,8,9,2,3,4,5), (5,7,4,5), (5,7,4,8,2,3),

(5,7,4,8,2,10,11,12), (5,7,4,8,2,10,11,13), (5,7,4,8,9,2,3),

(5,7,4,8,9,2,10,11,12), (5,7,4,8,9,2,10,11,13),

(6,8,2,3,4,5,6), (6,8,2,3,4,5,7), (6,8,9,2,3,4,5,6), (6,8,9,2,3,4,5,7),

(7,4,5,6,8,2,3), (7,4,5,6,8,2,10,11,12), (7,4,5,6,8,2,10,11,13),

(7,4,5,6,8,9,2,3), (7,4,5,6,8,9,2,10,11,12), (7,4,5,6,8,9,2,10,11,13),

(7,4,5,7),

(8,2,3,4,5,6,8), (8,2,3,4,8), (8,9,2,3,4,5,6,8), (8,9,2,3,4,8),

(9,2,3,4,5,6,8,9), (9,2,3,4,8,9),

(11,12,11),

(12,11,12), (12,11,13) }

主路径覆盖取n=1和n=5

1. 反射方法

由于函数是一个private方法，考虑去年Java课上的反射方法，获取method，setAccessible为true后，用invoke的方法进行调用。

1. 重定向

由于原文件只有System.out的输出，在不改变原文件的基础上:

@Before

**public** **void** setUpStreams() {

System.*setOut*(**new** PrintStream(outContent));

}

@After

**public** **void** cleanUpStreams() {

System.*setOut*(**null**);

}

**outContent.toString();**

把输出到console的信息转化成String,然后在Junit中和固有信息进行比对

代码：

package printPrime;

import static org.junit.Assert.assertEquals;

import java.io.ByteArrayOutputStream;

import java.io.PrintStream;

import java.lang.reflect.InvocationTargetException;

import java.lang.reflect.Method;

import java.util.Scanner;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class testprimes {

private final ByteArrayOutputStream outContent = new ByteArrayOutputStream();

//console信息转化到string中

@Before

public void setUpStreams() {

System.setOut(new PrintStream(outContent));

}

@After

public void cleanUpStreams() {

System.setOut(null);

}

@Test

public void testp() throws NoSuchMethodException, SecurityException, IllegalAccessException, IllegalArgumentException, InvocationTargetException{

PrintPrimes pp=new PrintPrimes();

//由于是private方法，采用反射机制，用method的方法调用函数

Class pc=pp.getClass();

Method m=pc.getDeclaredMethod("printPrimes", new Class[]{int.class});

m.setAccessible(true);

m.invoke(pp, 5);//n=5

m.invoke(pp, 1);//n=1

String expect="Prime: 2"+'\r'+'\n';

expect=expect+"Prime: 3"+'\r'+'\n';

expect=expect+"Prime: 5"+'\r'+'\n';

expect=expect+"Prime: 7"+'\r'+'\n';

expect=expect+"Prime: 11"+'\r'+'\n';//n=5

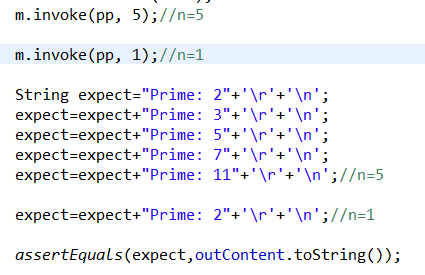
expect=expect+"Prime: 2"+'\r'+'\n';//n=1

assertEquals(expect,outContent.toString());

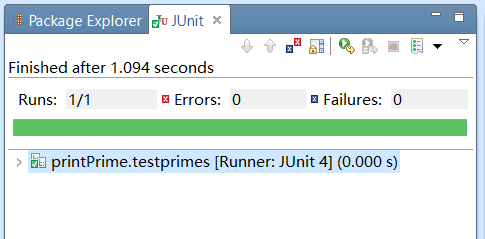
}

}

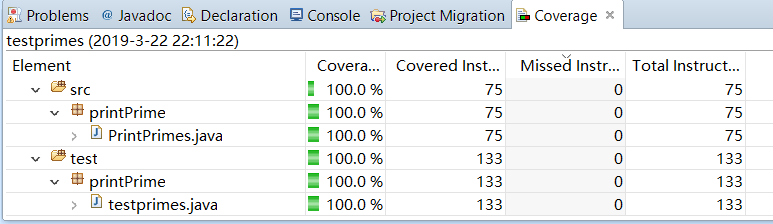
Input： n=5 和n=1



Output：



n=5和n=1放到了一个assertEquals里，故仅显示1/1



实现主路径覆盖的测试

Github：