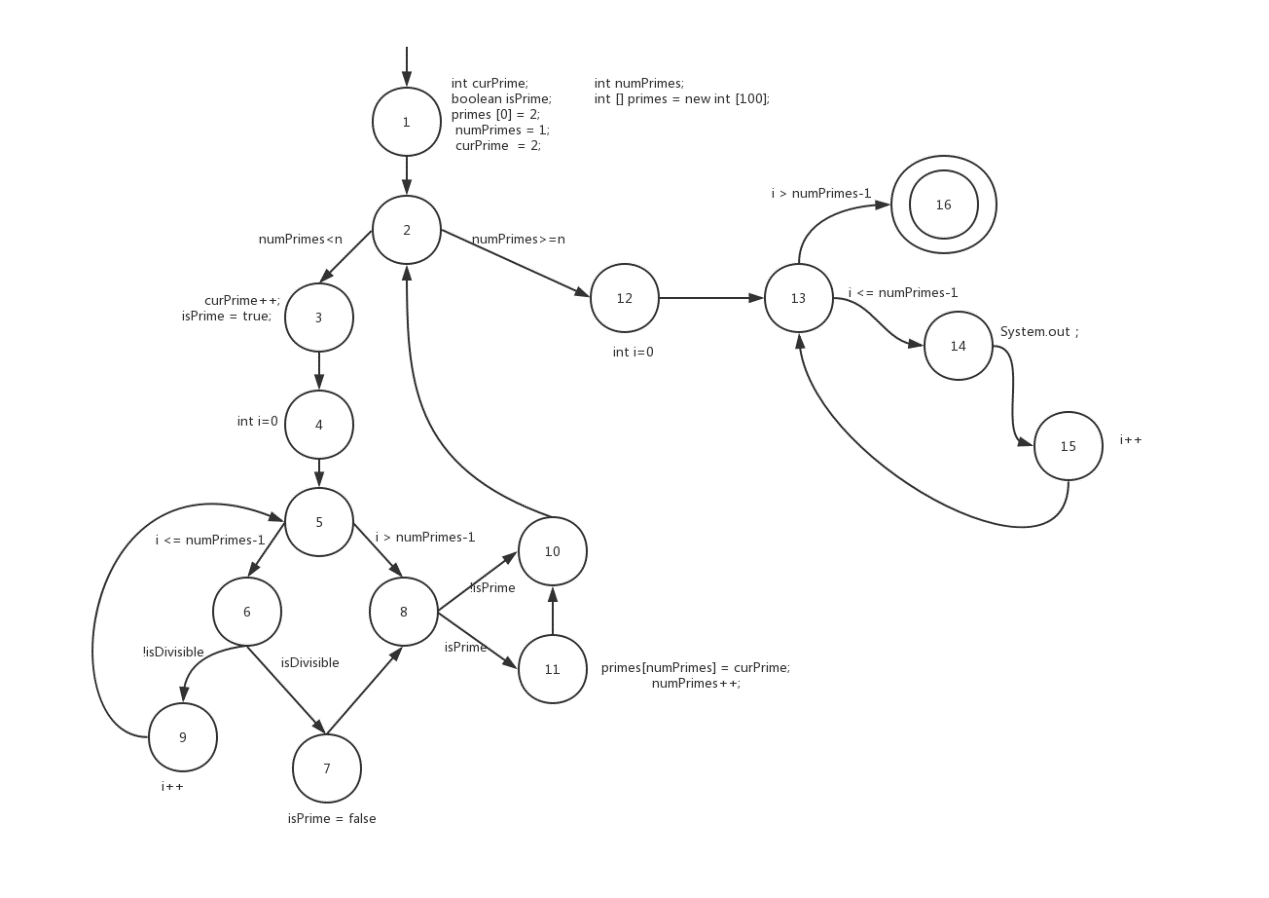
曾晓东 3016218101

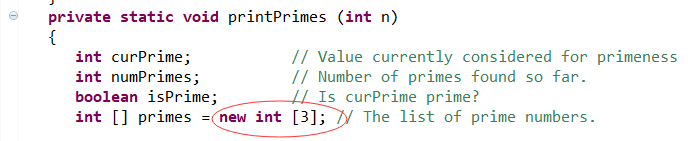
Homework3：

a)控制流图（process on画图）

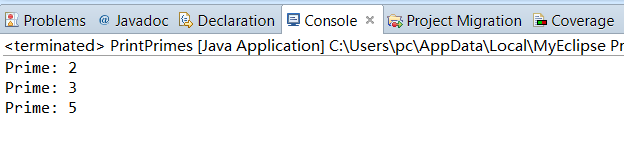


b)n=3测试的数据量比n=5要小

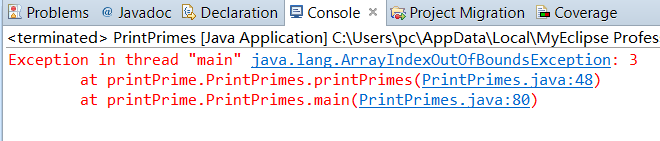
将代码中存储prime的数组的大小改成3



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：{1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16}

边覆盖：

{(1,2),(2,3),(2,12),(3,4),(4,5) ,(5,6) ,(5,8) ,(6,7) ,(6,9) ,(7,8) ,(8,10) ,(8,11) ,(9,5) ,(10,2) ,(11,10) ,(12,13) ,(13,14) ,(13,16) ,(14,15) ,(15,13) }

主路径覆盖：

{(1,2,3,4,5,6,9),(1,2,3,4,5,6,7,8,11,10),(1,2,3,4,5,6,7,8,10),(1,2,3,4,5,8,11,10),(1,2,3,4,5,8,10),(1,2,12,13,14,15),(1,2,12,13,16),

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

(3,4,5,6,7,8,11,10,2,12,13,14,15),(3,4,5,6,7,8,10,2,12,13,14,15),(3,4,5,6,7,8,11,10,2,12,13,16),(3,4,5,6,7,8,10,2,12,13,16),(3,4,5,8,11,10,2,12,13,14,15),(3,4,5,8,10,2,12,13,14,15),(3,4,5,8,11,10,2,12,13,16),(3,4,5,8,10,2,12,13,16),

(5,6,9,5),

(6,9,5,8,11,10,2,12,13,14,15),(6,9,5,8,10,2,12,13,14,15),(6,9,5,8,11,10,2,12,13,16),(6,9,5,8,10,2,12,13,16),

(13,14,15,13),

(14,15,13,16)}

设计主要思路

1. 如何全覆盖：

首先筛选出语义不可达的路径

a.按照正常运行方式，当程序走向1,2,3时，必然是number=3的情况，只会沿4,5,6,9方向走，在1,2,3的组合中去掉1.

b.如果isdivisable的话，必然不是质数，故6,7,8,11的路径语义不可达。

c.在4结点初始化i=0，numPrimes初始值为1，且numPrimes逐渐增大，故不可能初始化以后就满足i>numPrimes-1，故含4,5,8的路径语义不可达。

d.如果当前num满足！isPrime，则numPrimes不会变化，即不可能退出while循环，那么8,10,2,12的路径语义不可达

e.12,13,16同理c也不可达

利用上述规则对主路径进行拆分且归属于更长路径的子路径不做单独考虑得到：

{(1,2,3,4,5,6,9) ,

(1,2,12,13,14,15) ,

(2,3,4,5,6,7,8,10,2),

(5,6,9,5),

(6,9,5,8,11,10,2,12,13,14,15) ,

(13,14,15,13),

(14,15,13,16)}

(1,2,3,4,5,6,9)当前num=3即可覆盖此路径 n>=2

(1,2,12,13,14,15) n=1即可覆盖此路径

(2,3,4,5,6,7,8,10,2) 当前num=4即可覆盖此路径，但num=4无法退出循环，最小为下一个质数5 n>=3

(5,6,9,5) 当前num=3即可覆盖此路径 n>=2

(6,9,5,8,11,10,2,12,13,14,15) 当前num=3以及以上的质数并退出即可覆盖此路径 n>=2

(13,14,15,13)打印至少两个数，n>=2

(14,15,13,16)为一个正常结果打印的过程。

综上，要实现主路径覆盖需要一个n=1的测试例子，以及n>=3的测试例子。

由于题目中第二问的缘故，我选择了n=5。

最后取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

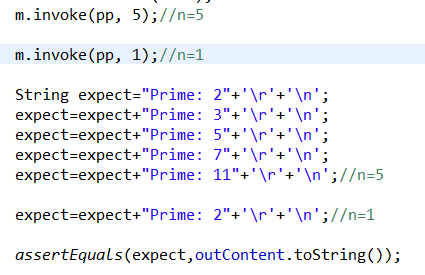
expect=expect+"Prime: 2"+'\r'+'\n';//n=1

assertEquals(expect,outContent.toString());

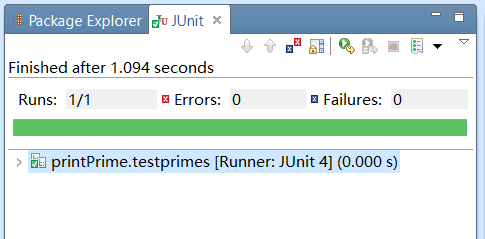
}

}

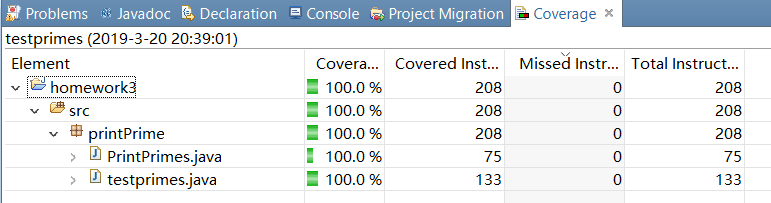
Input： n=5 和n=1



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：https://github.com/nkaccounting/softwaretest18192/tree/master/homework3/src/printPrime