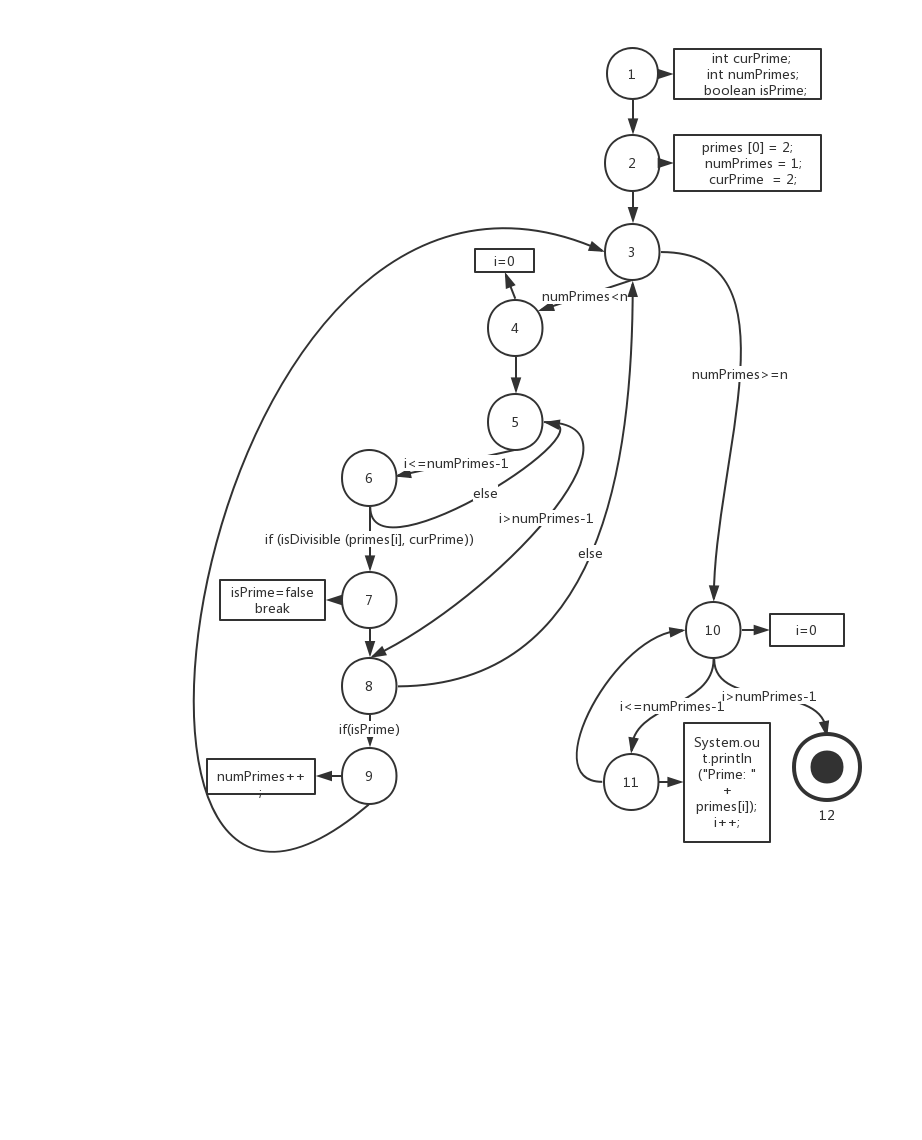
一：Introduction to Software Testing

P115 T7

a) Draw the control flow graph for the printPrimes() method.绘制控制流图

A:

控制流图如下



b) Consider test cases t1=(n=3) and t2=(n=5). Although these tour the same prime paths in printPrimes(), they do not necessarily find the same faults.Design a simple fault that t2 would be more likely to discover than t1 would.

考虑测试用例，n = 3 和 n = 5，它们游历相同的主路径。设计一个简单的错误，t2发现而t1不会发现。

A：

在Int [ ] primes = new int [100] 这一句中将100改为4，这样当进行t2测试的时候就会发生越界错误。而进行t1测试时不会发生越界错误

c) For printPrimes(), find a test case such that the corresponding test path visits the edge that connects the beginning of the while statement to the for statement without going through the body of the while loop.

设计一个测试用例，使其相应的测试路径访问连接while语句的起点和for语句的边，而不必执行while的循环体。

A:

令n=1,此时会访问while语句的起点和for语句的边，但并不执行while的循环体。

d)Enumerate the test requirements for node coverage, edge coverage, and prime path coverage for the graph for printPrimes().

列举节点覆盖，边覆盖和主路径覆盖的测试要求。

A:

点覆盖：{1,2,3,4,5,6,7,8,9,10,11,12}

边覆盖：{1-2, 2-3, 3-4, 3-10, 4-5, 5-6, 5-8, 6-5, 6-7, 7-8, 8-3, 8-9, 9-3,

10-11, 10-12, 11-10}

主路径覆盖：{[1,2,3,4,5,6,7,8,9], [1,2,3,10,11], [1,2,3,10,12], [1,2,3,4,5,6,7,8,9]

[3,4,5,6,7,8,9,3], [3,4,5,8,9,3], [3,4,5,6,7,8,3], [3,4,5,8,3], [4,5,6,7,8,9,3,10,11]

[4,5,6,7,8,9,3,10,12], [4,5,6,7,8,3,10,11], [4,5,6,7,8,3,10,12], [4,5,7,8,3,10,11], [4,5,7,8,3,10,12], [4,5,7,8,9,3,10,11], [4,5,7,8,9,3,10,12], [5,6,5], [5,6,7,8,9,3,4,5],

[5,6,7,8,3,4,5], [5,8,3,4,5], [5,8,9,3,4,5], [6,5,6], [6,7,8,9,3,4,5,6], [6,7,8,3,4,5,6],

[7,8,3,4,5,6,7], [7,8,9,3,4,5,6,7], [8,3,4,5,6,7,8], [8,3,4,5,8], [8,9,3,4,5,6,7,8],

[8,9,3,4,5,8], [9,3,4,5,8,9], [9,3,4,5,6,7,8,9], [10,11,10], [11,10,12], [11,10,11] }

二：基于Junit及Eclemma实现一个主路径覆盖的测试

测试代码如下图所示

package softwareTest;

import static org.junit.Assert.\*;

import org.junit.After;

import org.junit.Before;

import org.junit.Test;

public class PrimeTest {

@Test

public void test1() {

PrintPrimes.printPrimes(10);

}

@Test

public void test2() {

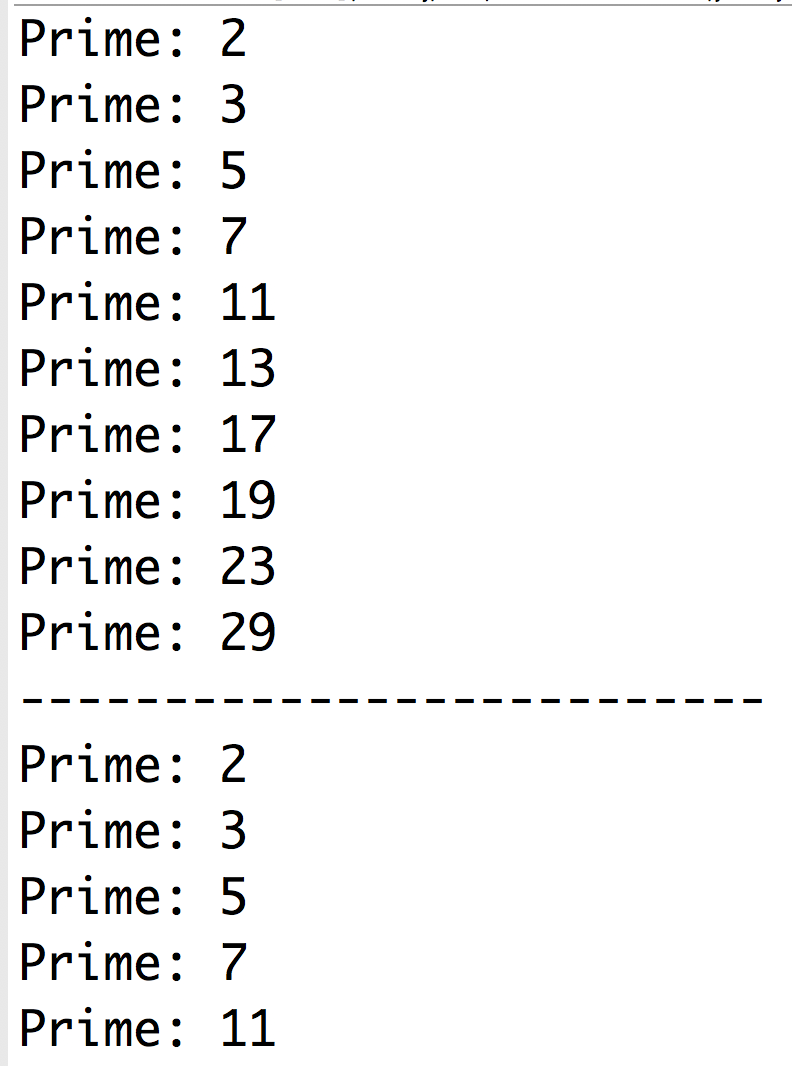
System.out.println("--------------------------");

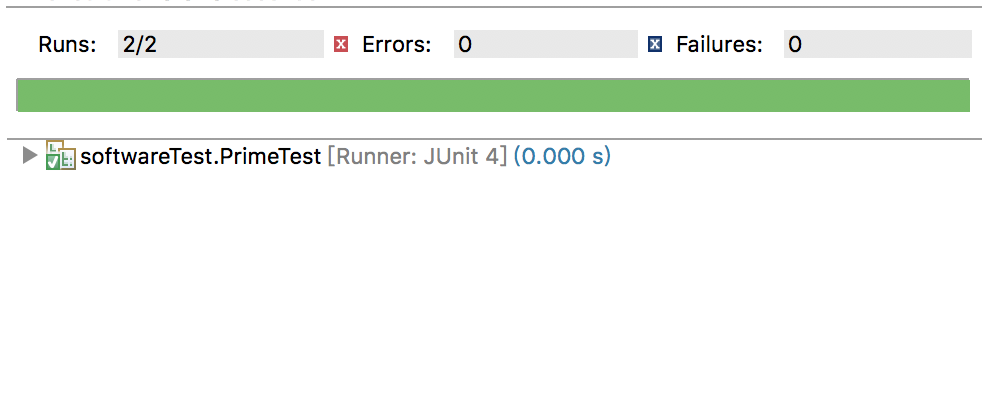
PrintPrimes.printPrimes(5);

}

}

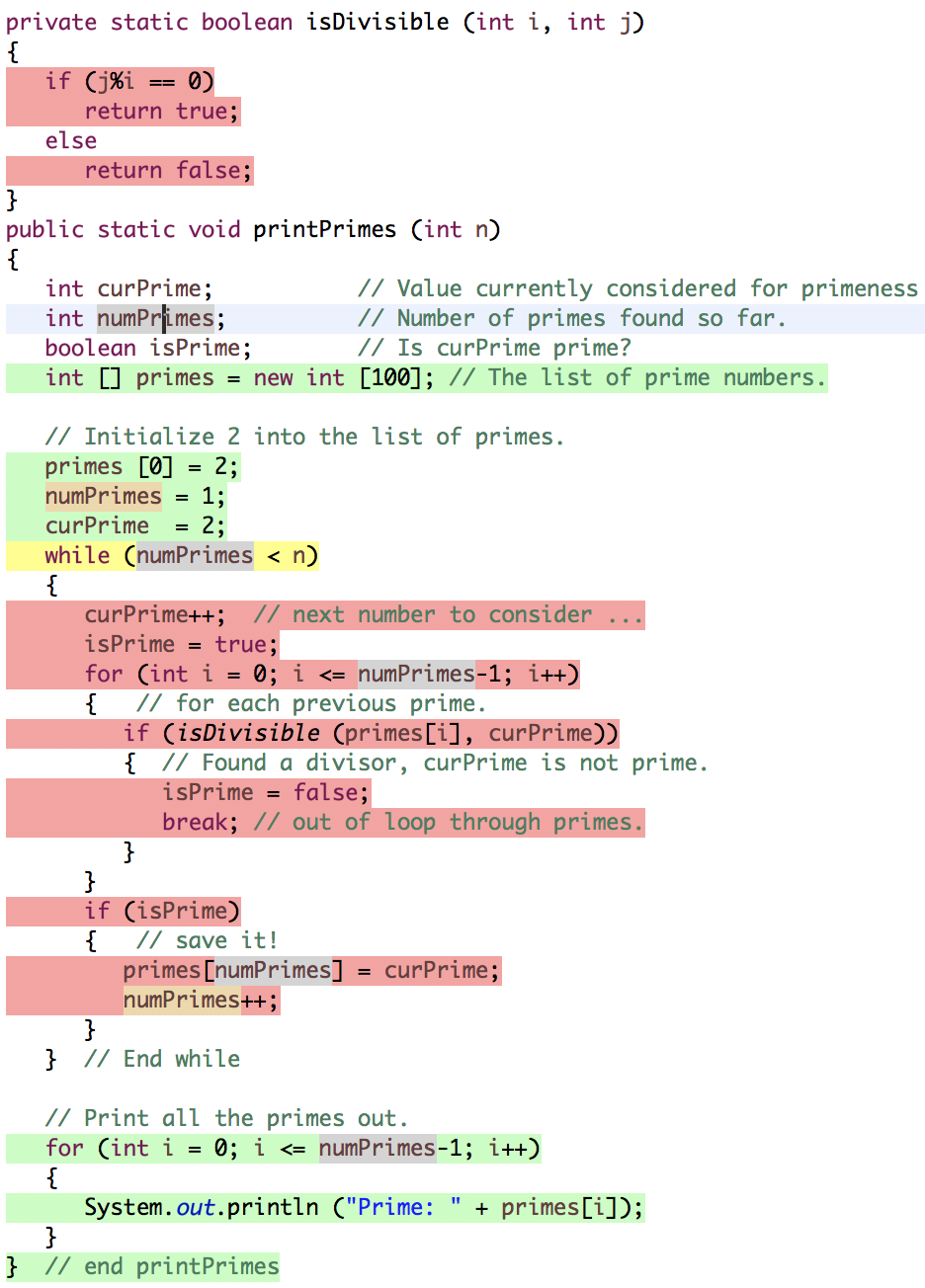
运行后输出和测试结果如下图所示





使用Eclemma进行覆盖测试

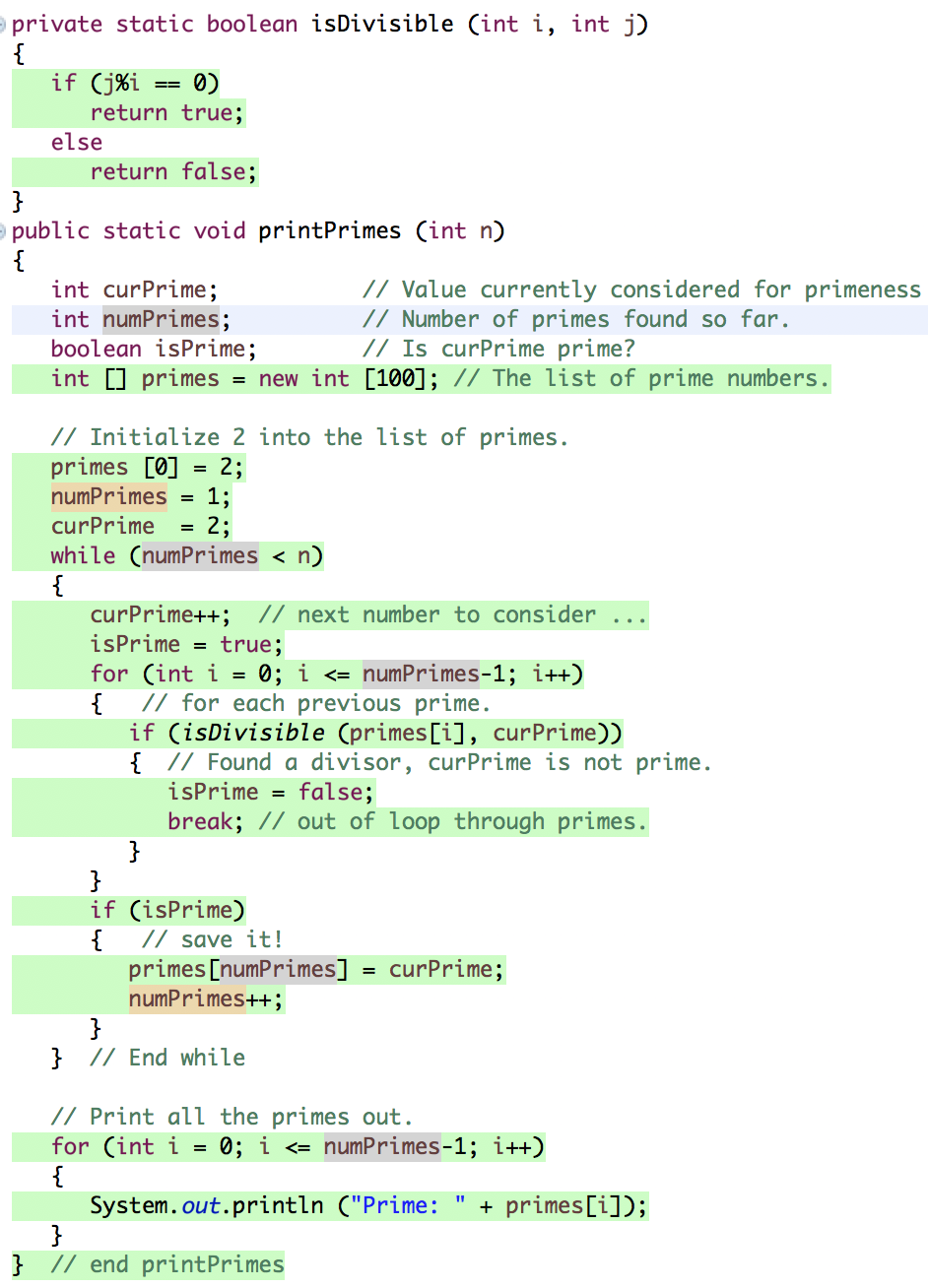
当测试参数为1时，覆盖图以及覆盖率如下

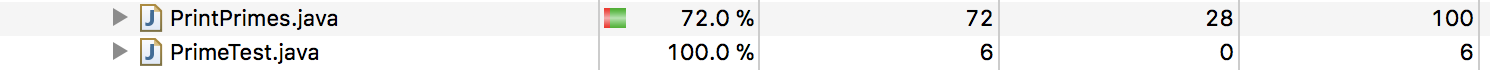




可以看出跳过了while循环直接进行for循环的print，因为一开始就默认存在一个质数2。

当测试参数为10时，覆盖图以及覆盖率如下





可以看出printPrimes()函数中的代码被全部执行。