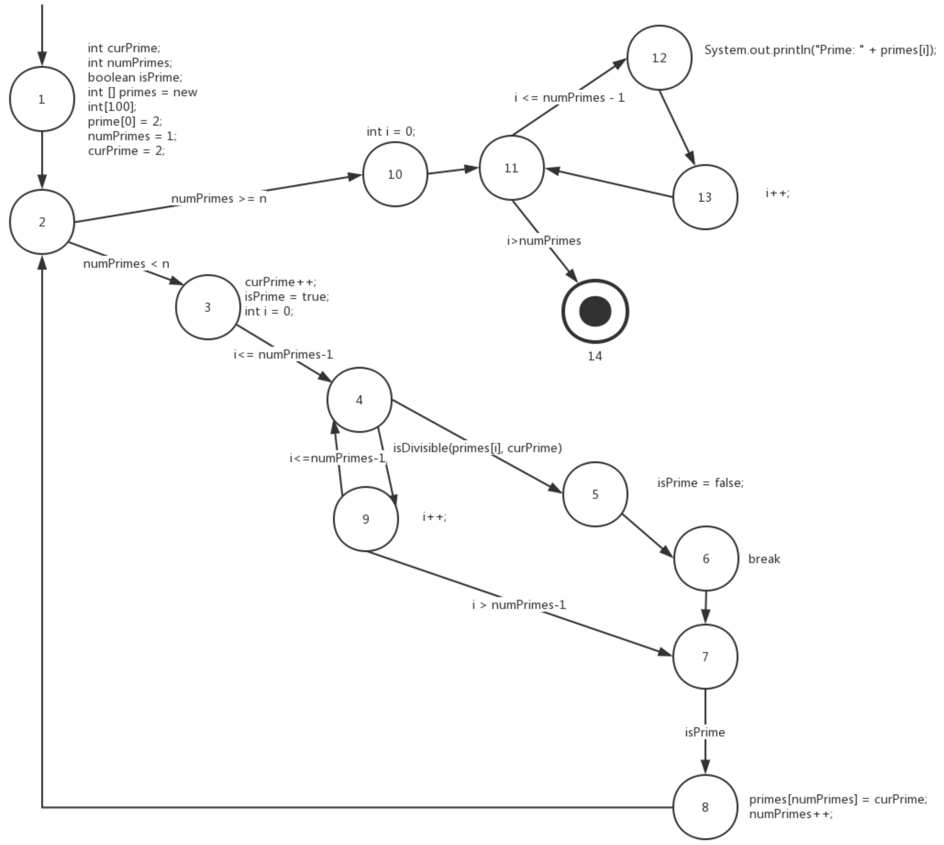
一：Introduction to Software Testing

a) 控制流图如下



b)将数组 primes 的元素个数改成 4

c)令 n=1

d)点覆盖：{1,2,3,4,5,6,7,8,9,10,11,12,13,14}

边覆盖：{(1,2), (2,3), (2,10), (3,4), (10,11), (4,5), (4,9), (11,12),(11,14), (5,6), (9,4), (9,7), (12,13), (13,11), (6,7), (7,8), (8,2)}

主路径覆盖：

{[1,2,3,4,9,7,8],

[1,2,3,4,5,6,7,8],

[1,2,10,11,12,13],

[1,2,10,11,14]

[2,3,4,5,6,7,8,2],

[2,3,4,9,7,8,2],

[2,10,11,14],

[3,4,5,6,7,8,2,10,11,12,13],

[3,4,9,7,8,2,10,11,12,13],

[4,9,4],

[4,5,6,7,8,2,3,4],

[4,9,7,8,2,3,4],

[4,5,6,7,8,2,10,11,14],

[4,9,7,8,2,10,11,12,13],

[5,6,7,8,2,3,4,5],

[5,6,7,8,2,10,11,12,13],

[5,6,7,8,2,10,11,14],

[6,7,8,2,3,4,5,6],

[6,7,8,2,10,11,14],

[6,7,8,2,10,11,12,13],

[7,8,2,3,4,5,6,7],

[7,8,2,3,4,9,7],

[7,8,2,10,11,14],

[7,8,2,10,11,12,13],

[8,2,3,4,9,7,8],

[8,2,3,4,5,6,7,8],

[8,2,10,11,14],

[8,2,10,11,12,13],

[9,7,8,2,3,4,9],

[11,12,13,11],

[12,13,11,14]

[12,13,11,12]

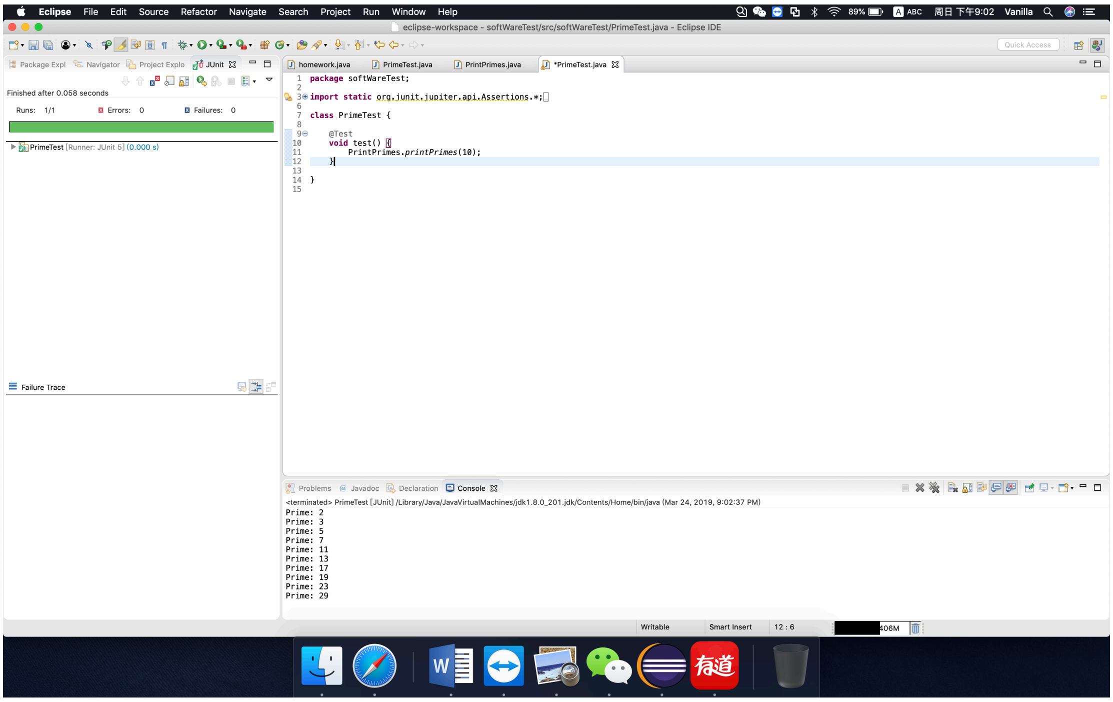
[13,11,12]}

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



Finished after 0.058 seconds

Runs: 1/1 Errors: 0 Failures: 0

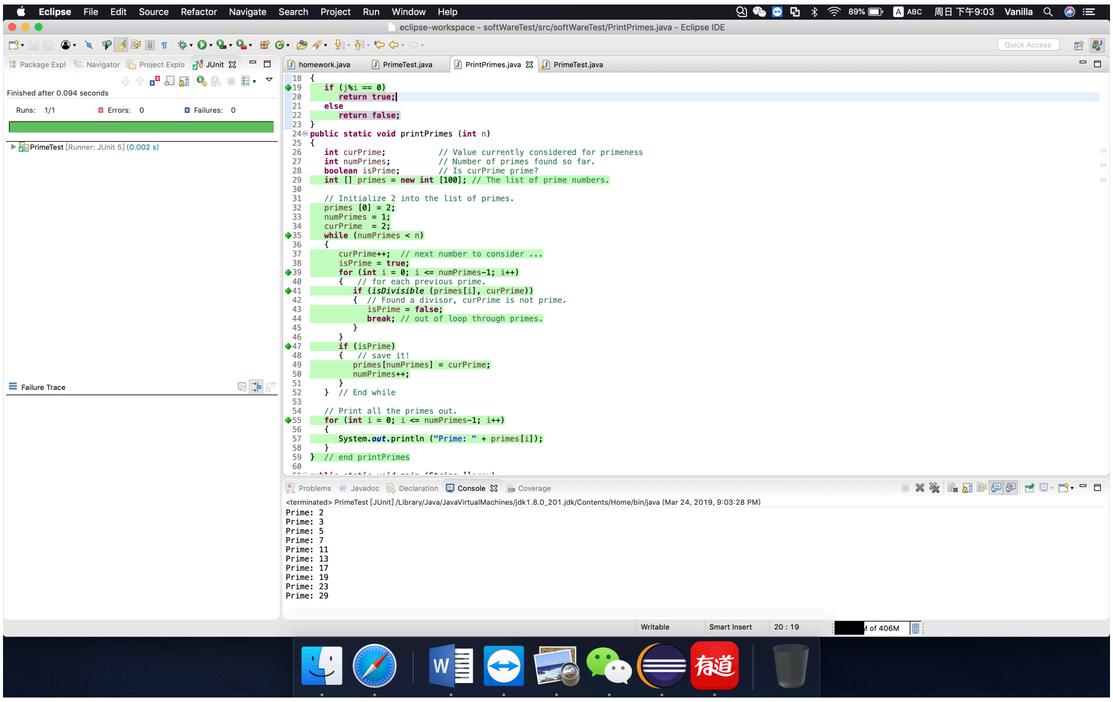
PrimeTest [Runner: JUnit 5] (0.000 s)

```
package softWareTest;

import static org.junit.jupiter.api.Assertions.*;

class PrimeTest {

    @Test
    void test() {
        PrintPrimes.printPrimes(10);
    }

}
```

```
<terminated> PrimeTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java (Mar 24, 2019, 9:02:37 PM)
Prime: 2
Prime: 3
Prime: 5
Prime: 7
Prime: 11
Prime: 13
Prime: 17
Prime: 19
Prime: 23
Prime: 29
```

使用 Eclemma 进行覆盖测试

当测试参数为 10 时，覆盖图以及覆盖率如下



Finished after 0.094 seconds

Runs: 1/1 Errors: 0 Failures: 0

PrimeTest [Runner: JUnit 5] (0.002 s)

```
{
    if (j%i == 0)
        return true;
    else
        return false;
}
public static void printPrimes (int n)
{
    int curPrime;              // Value currently considered for primeness
    int numPrimes;             // Number of primes found so far.
    boolean isPrime;           // Is curPrime prime?
    int [] primes = new int [100]; // The list of prime numbers.

    // Initialize 2 into the list of primes.
    primes [0] = 2;
    numPrimes = 1;
    curPrime  = 2;
    while (numPrimes < n)
    {
        curPrime++;  // next number to consider ...
        isPrime = true;
        for (int i = 0; i <= numPrimes-1; i++)
        {   // for each previous prime.
            if (isDivisible (primes[i], curPrime))
            {  // Found a divisor, curPrime is not prime.
                isPrime = false;
                break; // out of loop through primes.
            }
        }
        if (isPrime)
        {   // save it!
            primes[numPrimes] = curPrime;
            numPrimes++;
        }
    }  // End while

    // Print all the primes out.
    for (int i = 0; i <= numPrimes-1; i++)
    {
        System.out.println ("Prime: " + primes[i]);
    }
}  // end printPrimes

public static void main (String []argv)
```

```
<terminated> PrimeTest [JUnit] /Library/Java/JavaVirtualMachines/jdk1.8.0_201.jdk/Contents/Home/bin/java (Mar 24, 2019, 9:03:28 PM)
Prime: 2
Prime: 3
Prime: 5
Prime: 7
Prime: 11
Prime: 13
Prime: 17
Prime: 19
Prime: 23
Prime: 29
```

Writable Smart Insert 20 : 19 [redacted]M of 406M

可以看出 printPrimes()函数中的代码被全部执行。