Neha Maddali

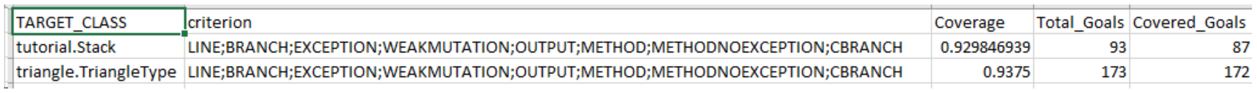
ComS417 Assignment 2

Problem 1 and Problem 2

Evosuite downloaded and extracted. Command lines run successfully.

Problem 3

a) statistics.csv

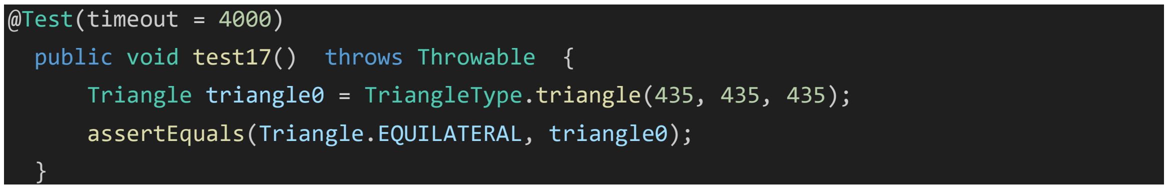


TARGET_CLASS	criterion	Coverage	Total_Goals	Covered_Goals
tutorial.Stack	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.929846939	93	87
triangle.TriangleType	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.9375	173	172

b) Number of tests for Triangle: 20

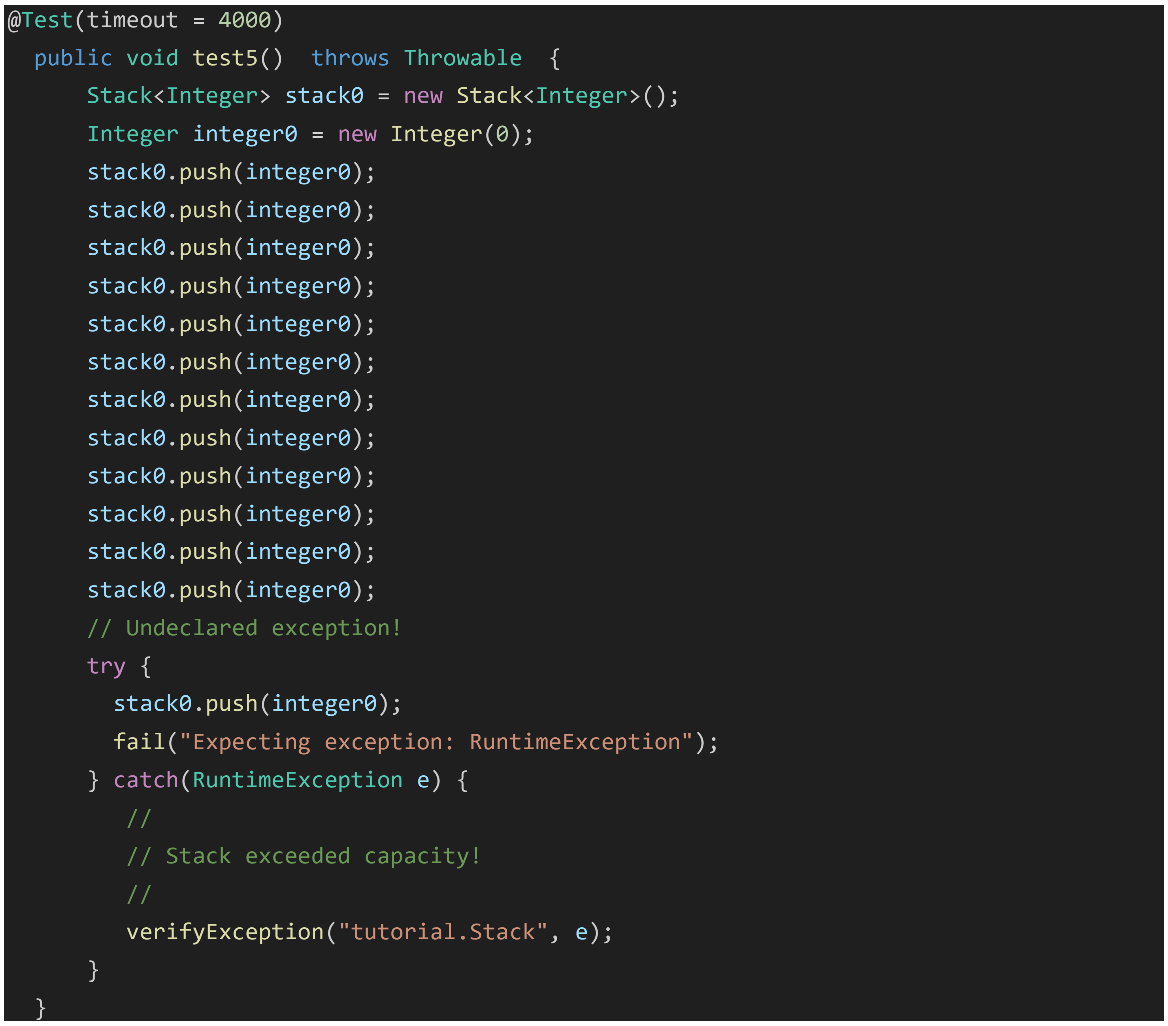
Number of tests for Tutorial: 6

c) Example test



```
@Test(timeout = 4000)
  public void test17()  throws Throwable  {
      Triangle triangle0 = TriangleType.triangle(435, 435, 435);
      assertEquals(Triangle.EQUILATERAL, triangle0);
  }
```

d) Example test

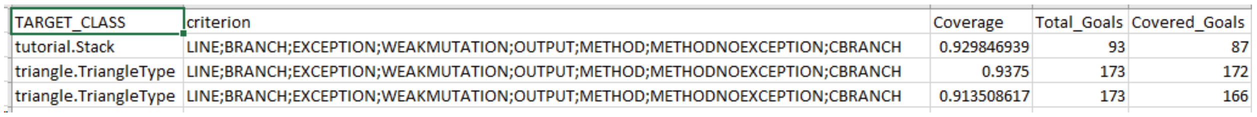


```
@Test(timeout = 4000)
  public void test5()  throws Throwable  {
      Stack<Integer> stack0 = new Stack<Integer>();
      Integer integer0 = new Integer(0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      stack0.push(integer0);
      // Undeclared exception!
      try {
        stack0.push(integer0);
        fail("Expecting exception: RuntimeException");
      } catch(RuntimeException e) {
         //
         // Stack exceeded capacity!
         //
         verifyException("tutorial.Stack", e);
      }
  }
```

Neha Maddali

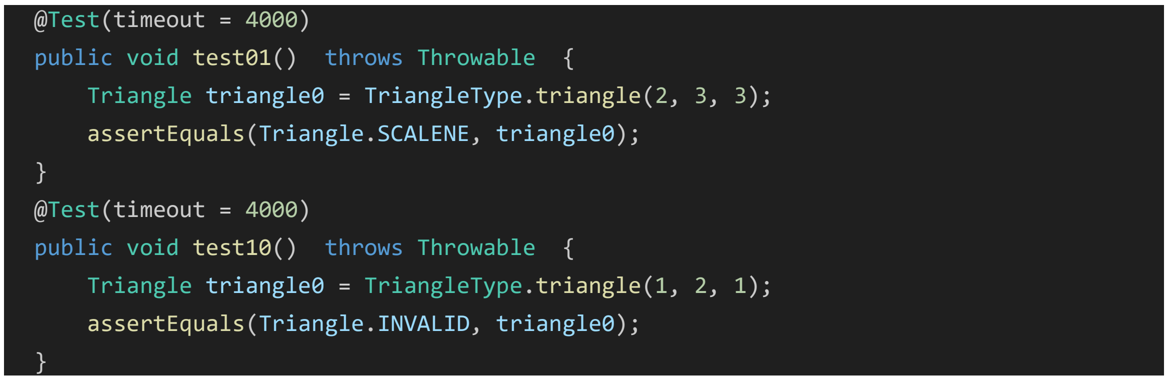
Problem 4

a) After modifying the TriangleType.java by adding the fault into the isosceles branch the number of tests generated were 18. The 3rd row shows the report.



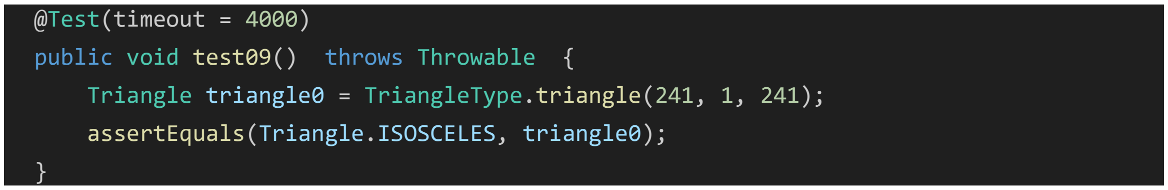
TARGET_CLASS	criterion	Coverage	Total_Goals	Covered_Goals
tutorial.Stack	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.929846939	93	87
triangle.TriangleType	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.9375	173	172
triangle.TriangleType	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.913508617	173	166

There are a couple of faults in the test cases. The following should be isosceles triangles, but they are not.



```
@Test(timeout = 4000)
public void test01()  throws Throwable  {
    Triangle triangle0 = TriangleType.triangle(2, 3, 3);
    assertEquals(Triangle.SCALENE, triangle0);
}
@Test(timeout = 4000)
public void test10()  throws Throwable  {
    Triangle triangle0 = TriangleType.triangle(1, 2, 1);
    assertEquals(Triangle.INVALID, triangle0);
}
```

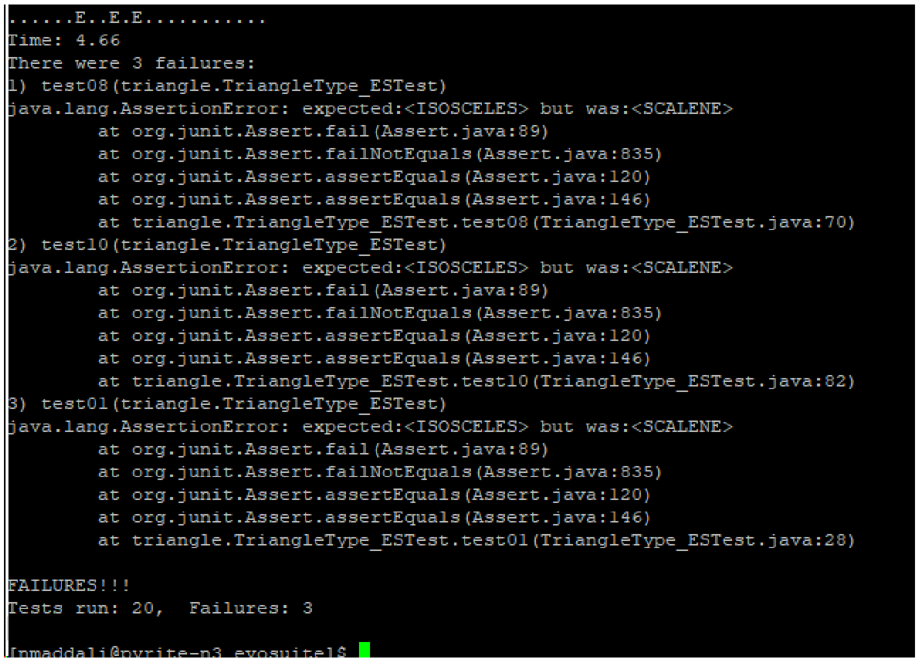
b) Here is the isosceles test that covers the faulty branch. This passed because the faulty line was: *if ((s1 == s2) || (s1 == s3) || (s1 == s3))*
241 = 241 in this test case, so it will pass as isosceles.



```
@Test(timeout = 4000)
public void test09()  throws Throwable  {
    Triangle triangle0 = TriangleType.triangle(241, 1, 241);
    assertEquals(Triangle.ISOSCELES, triangle0);
}
```

The tests I mentioned in part (a) are classified incorrectly because of the faulty code.

c) Here is a screenshot of the result of running the original tests on the faulty program.

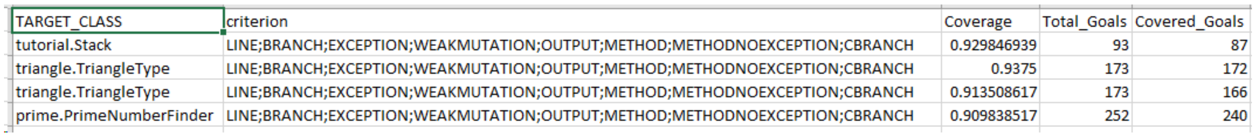


```
......E..E.E..........
Time: 4.66
There were 3 failures:
1) test08(triangle.TriangleType_ESTest)
java.lang.AssertionError: expected:<ISOSCELES> but was:<SCALENE>
        at org.junit.Assert.fail(Assert.java:89)
        at org.junit.Assert.failNotEquals(Assert.java:835)
        at org.junit.Assert.assertEquals(Assert.java:120)
        at org.junit.Assert.assertEquals(Assert.java:146)
        at triangle.TriangleType_ESTest.test08(TriangleType_ESTest.java:70)
2) test10(triangle.TriangleType_ESTest)
java.lang.AssertionError: expected:<ISOSCELES> but was:<SCALENE>
        at org.junit.Assert.fail(Assert.java:89)
        at org.junit.Assert.failNotEquals(Assert.java:835)
        at org.junit.Assert.assertEquals(Assert.java:120)
        at org.junit.Assert.assertEquals(Assert.java:146)
        at triangle.TriangleType_ESTest.test10(TriangleType_ESTest.java:82)
3) test01(triangle.TriangleType_ESTest)
java.lang.AssertionError: expected:<ISOSCELES> but was:<SCALENE>
        at org.junit.Assert.fail(Assert.java:89)
        at org.junit.Assert.failNotEquals(Assert.java:835)
        at org.junit.Assert.assertEquals(Assert.java:120)
        at org.junit.Assert.assertEquals(Assert.java:146)
        at triangle.TriangleType_ESTest.test01(TriangleType_ESTest.java:28)

FAILURES!!!
Tests run: 20,  Failures: 3

[nmaddali@pyrite-n3 evosuite]$
```

Neha Maddali

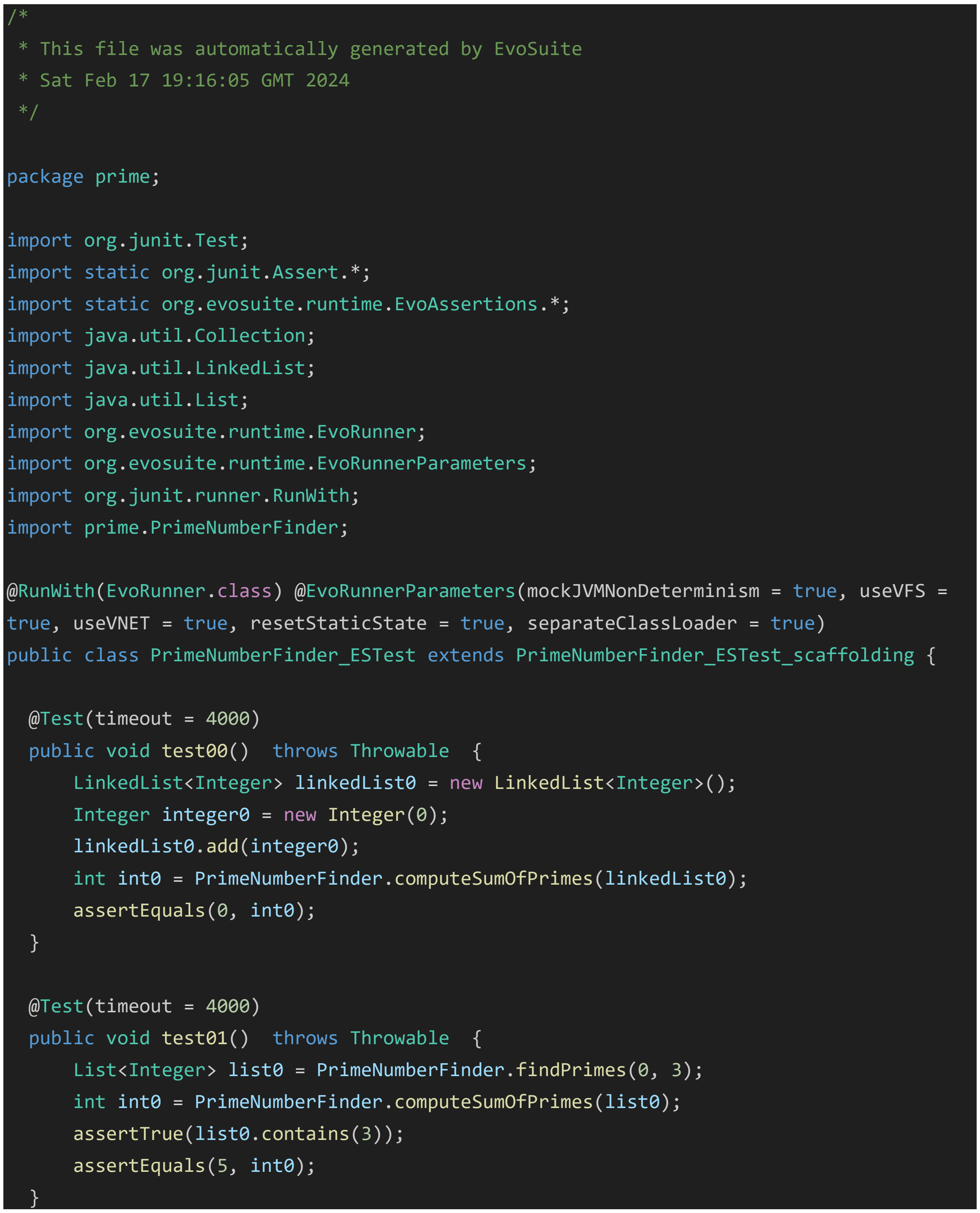
Problem 5

a) statistics.csv



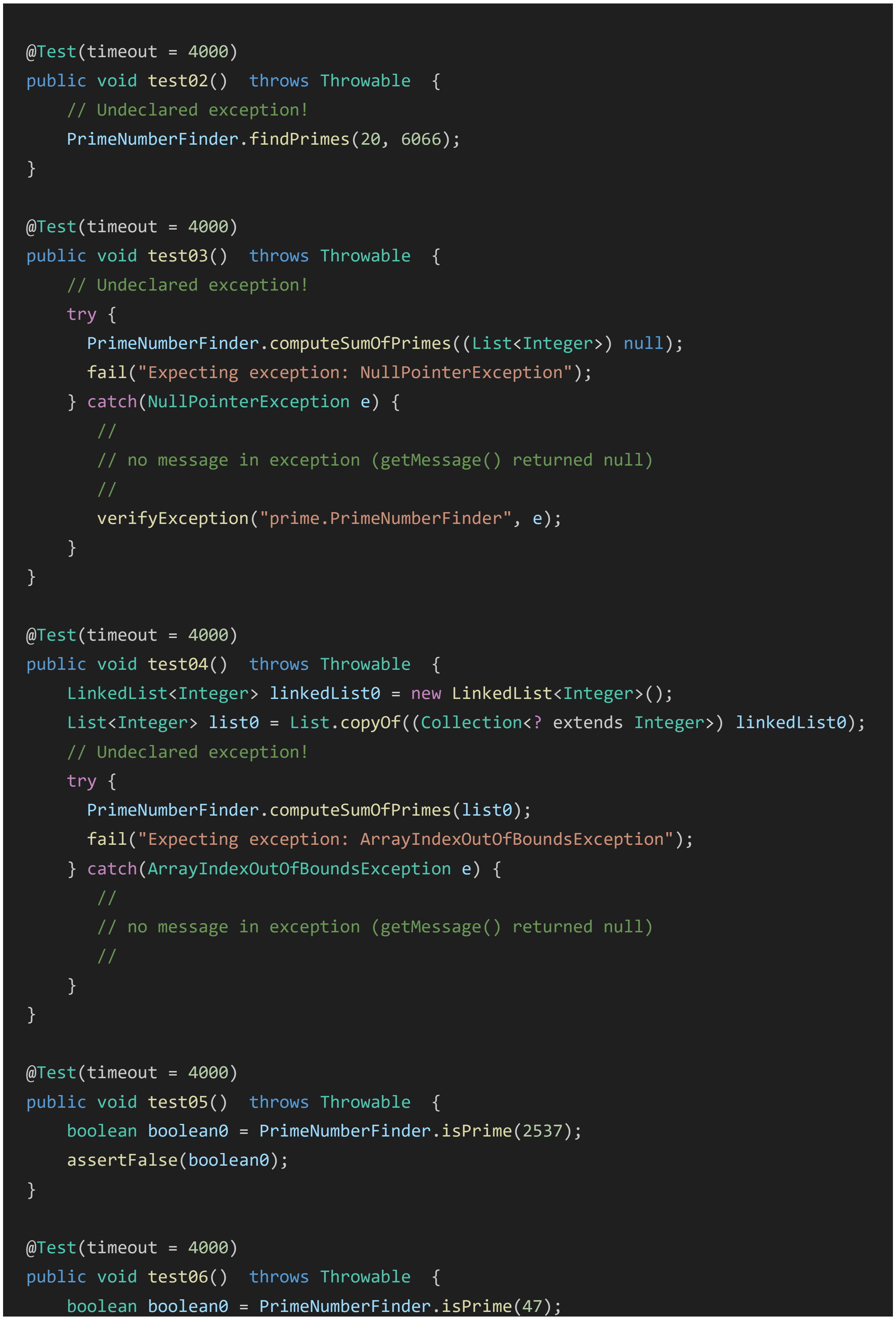
TARGET_CLASS	criterion	Coverage	Total_Goals	Covered_Goals
tutorial.Stack	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.929846939	93	87
triangle.TriangleType	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.9375	173	172
triangle.TriangleType	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.913508617	173	166
prime.PrimeNumberFinder	LINE;BRANCH;EXCEPTION;WEAKMUTATION;OUTPUT;METHOD;METHODNOEXCEPTION;CBRANCH	0.909838517	252	240

b) PrimeNumberFinder_ESTest.java



```
/*
 * This file was automatically generated by EvoSuite
 * Sat Feb 17 19:16:05 GMT 2024
 */

package prime;

import org.junit.Test;
import static org.junit.Assert.*;
import static org.evosuite.runtime.EvoAssertions.*;
import java.util.Collection;
import java.util.LinkedList;
import java.util.List;
import org.evosuite.runtime.EvoRunner;
import org.evosuite.runtime.EvoRunnerParameters;
import org.junit.runner.RunWith;
import prime.PrimeNumberFinder;

@RunWith(EvoRunner.class) @EvoRunnerParameters(mockJVMNonDeterminism = true, useVFS =
true, useVNET = true, resetStaticState = true, separateClassLoader = true)
public class PrimeNumberFinder_ESTest extends PrimeNumberFinder_ESTest_scaffolding {

  @Test(timeout = 4000)
  public void test00()  throws Throwable  {
      LinkedList<Integer> linkedList0 = new LinkedList<Integer>();
      Integer integer0 = new Integer(0);
      linkedList0.add(integer0);
      int int0 = PrimeNumberFinder.computeSumOfPrimes(linkedList0);
      assertEquals(0, int0);
  }

  @Test(timeout = 4000)
  public void test01()  throws Throwable  {
      List<Integer> list0 = PrimeNumberFinder.findPrimes(0, 3);
      int int0 = PrimeNumberFinder.computeSumOfPrimes(list0);
      assertTrue(list0.contains(3));
      assertEquals(5, int0);
  }
```

Neha Maddali

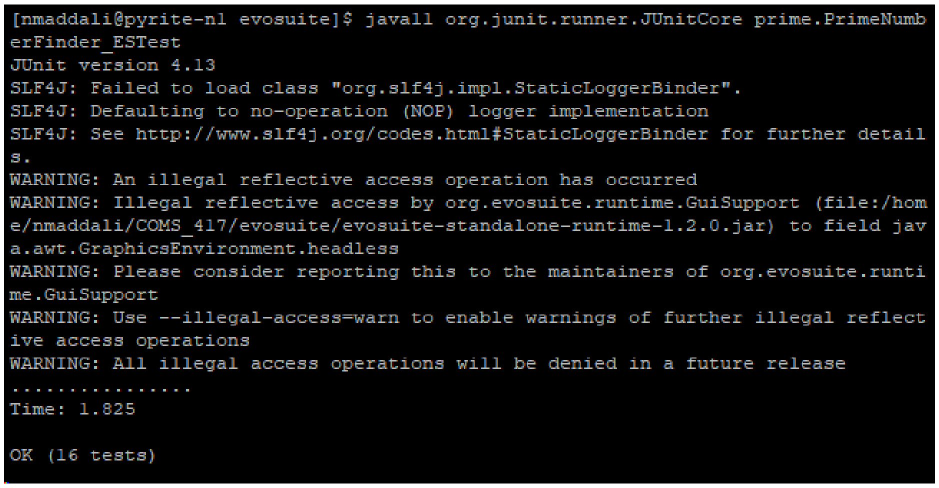


```
@Test(timeout = 4000)
public void test02()  throws Throwable  {
    // Undeclared exception!
    PrimeNumberFinder.findPrimes(20, 6066);
}

@Test(timeout = 4000)
public void test03()  throws Throwable  {
    // Undeclared exception!
    try {
      PrimeNumberFinder.computeSumOfPrimes((List<Integer>) null);
      fail("Expecting exception: NullPointerException");
    } catch(NullPointerException e) {
       //
       // no message in exception (getMessage() returned null)
       //
       verifyException("prime.PrimeNumberFinder", e);
    }
}

@Test(timeout = 4000)
public void test04()  throws Throwable  {
    LinkedList<Integer> linkedList0 = new LinkedList<Integer>();
    List<Integer> list0 = List.copyOf((Collection<? extends Integer>) linkedList0);
    // Undeclared exception!
    try {
      PrimeNumberFinder.computeSumOfPrimes(list0);
      fail("Expecting exception: ArrayIndexOutOfBoundsException");
    } catch(ArrayIndexOutOfBoundsException e) {
       //
       // no message in exception (getMessage() returned null)
       //
    }
}

@Test(timeout = 4000)
public void test05()  throws Throwable  {
    boolean boolean0 = PrimeNumberFinder.isPrime(2537);
    assertFalse(boolean0);
}

@Test(timeout = 4000)
public void test06()  throws Throwable  {
    boolean boolean0 = PrimeNumberFinder.isPrime(47);
```

```
        assertTrue(boolean0);
    }

    @Test(timeout = 4000)
    public void test07()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime(2793);
        assertFalse(boolean0);
    }

    @Test(timeout = 4000)
    public void test08()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime(16);
        assertFalse(boolean0);
    }

    @Test(timeout = 4000)
    public void test09()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime(2);
        assertTrue(boolean0);
    }

    @Test(timeout = 4000)
    public void test10()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime(3);
        assertTrue(boolean0);
    }

    @Test(timeout = 4000)
    public void test11()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime(4225);
        assertFalse(boolean0);
    }

    @Test(timeout = 4000)
    public void test12()  throws Throwable  {
        boolean boolean0 = PrimeNumberFinder.isPrime((-3516));
        assertFalse(boolean0);
    }

    @Test(timeout = 4000)
    public void test13()  throws Throwable  {
        Integer integer0 = new Integer((-3516));
        List<Integer> list0 = List.of(integer0, integer0);
        int int0 = PrimeNumberFinder.computeSumOfPrimes(list0);
        assertEquals((-7032), int0);
```

```
  }

  @Test(timeout = 4000)
  public void test14()  throws Throwable  {
      List<Integer> list0 = PrimeNumberFinder.findPrimes(41, 0);
      // Undeclared exception!
      try {
        PrimeNumberFinder.computeSumOfPrimes(list0);
        fail("Expecting exception: IndexOutOfBoundsException");
      } catch(IndexOutOfBoundsException e) {
      }
  }

  @Test(timeout = 4000)
  public void test15()  throws Throwable  {
      PrimeNumberFinder primeNumberFinder0 = new PrimeNumberFinder();
  }
}
```

c) Test results



```
[nmaddali@pyrite-n1 evosuite]$ javall org.junit.runner.JUnitCore prime.PrimeNumb
erFinder_ESTest
JUnit version 4.13
SLF4J: Failed to load class "org.slf4j.impl.StaticLoggerBinder".
SLF4J: Defaulting to no-operation (NOP) logger implementation
SLF4J: See http://www.slf4j.org/codes.html#StaticLoggerBinder for further detail
s.
WARNING: An illegal reflective access operation has occurred
WARNING: Illegal reflective access by org.evosuite.runtime.GuiSupport (file:/hom
e/nmaddali/COMS_417/evosuite/evosuite-standalone-runtime-1.2.0.jar) to field jav
a.awt.GraphicsEnvironment.headless
WARNING: Please consider reporting this to the maintainers of org.evosuite.runti
me.GuiSupport
WARNING: Use --illegal-access=warn to enable warnings of further illegal reflect
ive access operations
WARNING: All illegal access operations will be denied in a future release
................
Time: 1.825

OK (16 tests)
```

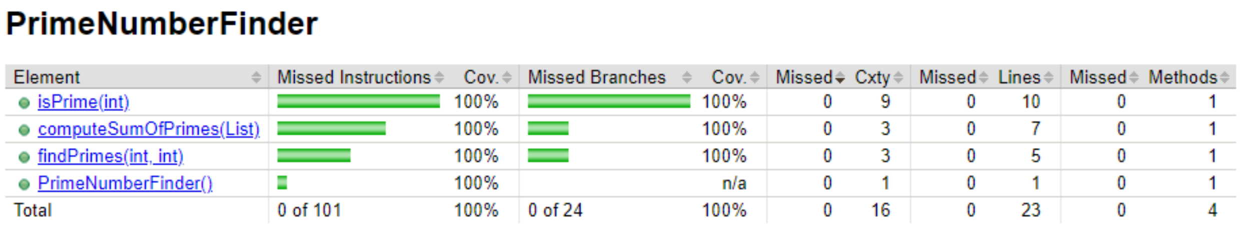
d) This test is interesting because it appears to be intentionally causing an exception with calling *findPrimes* using a range of [20,6066]. This test seems to be checking the behavior of the method when provided with a potentially invalid range.

```
@Test(timeout = 4000)
  public void test02()  throws Throwable  {
      // Undeclared exception!
      PrimeNumberFinder.findPrimes(20, 6066);
  }
```

Neha Maddali

Problem 6

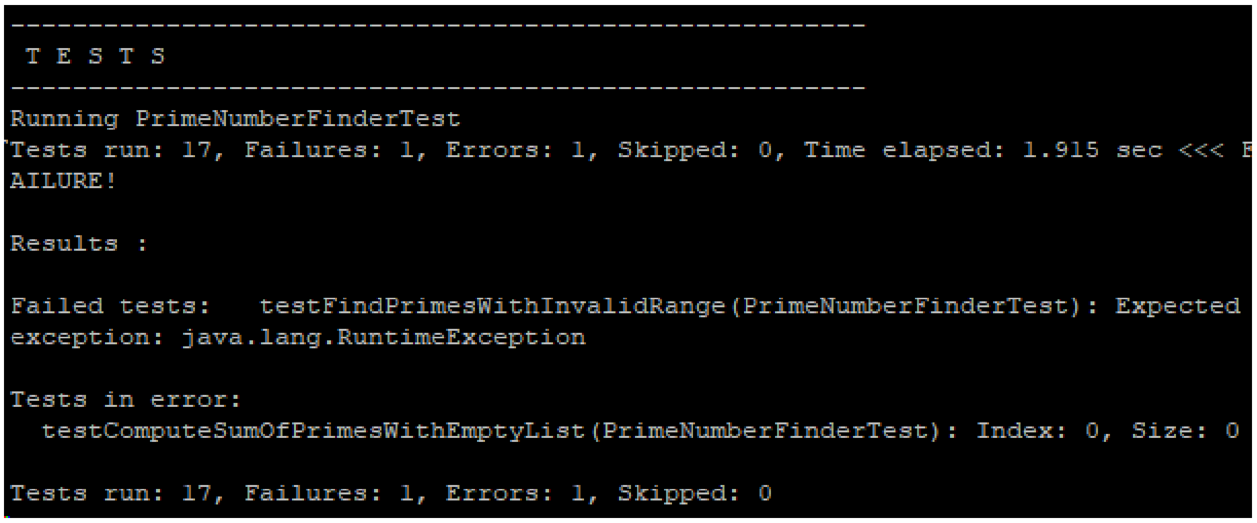
a) From Assignment 1, after creating my own test cases, I got 100% coverage with jacoco



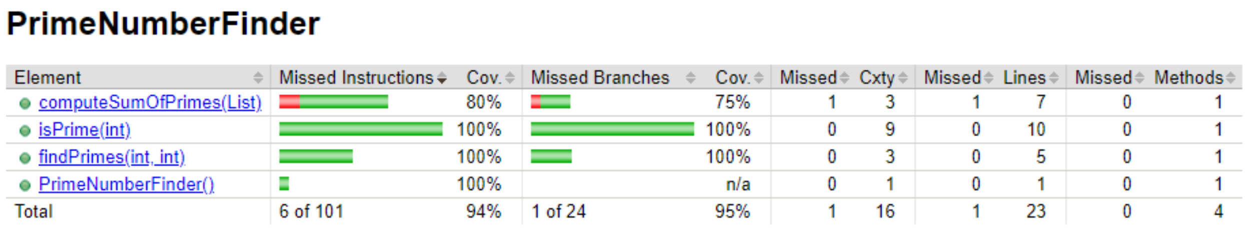
PrimeNumberFinder

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
isPrime(int)	🟩	100%	🟩	100%	0	9	0	10	0	1
computeSumOfPrimes(List)	🟩	100%	🟩	100%	0	3	0	7	0	1
findPrimes(int, int)	🟩	100%	🟩	100%	0	3	0	5	0	1
PrimeNumberFinder()	🟩	100%		n/a	0	1	0	1	0	1
Total	0 of 101	100%	0 of 24	100%	0	16	0	23	0	4

Here is the coverage with the test cases that came from evosuite which I converted into Junit format and ran with jacoco:



```
-------------------------------------------------------
 T E S T S
-------------------------------------------------------
Running PrimeNumberFinderTest
Tests run: 17, Failures: 1, Errors: 1, Skipped: 0, Time elapsed: 1.915 sec <<< F
AILURE!

Results :

Failed tests:   testFindPrimesWithInvalidRange(PrimeNumberFinderTest): Expected
exception: java.lang.RuntimeException

Tests in error:
  testComputeSumOfPrimesWithEmptyList(PrimeNumberFinderTest): Index: 0, Size: 0

Tests run: 17, Failures: 1, Errors: 1, Skipped: 0
```

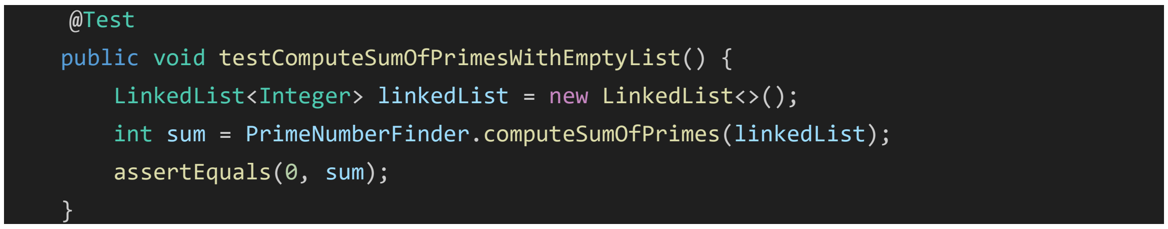


PrimeNumberFinder

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed	Cxty	Missed	Lines	Missed	Methods
computeSumOfPrimes(List)	🟥🟩	80%	🟥🟩	75%	1	3	1	7	0	1
isPrime(int)	🟩	100%	🟩	100%	0	9	0	10	0	1
findPrimes(int, int)	🟩	100%	🟩	100%	0	3	0	5	0	1
PrimeNumberFinder()	🟩	100%		n/a	0	1	0	1	0	1
Total	6 of 101	94%	1 of 24	95%	1	16	1	23	0	4

The test cases generated from evosuite didn't provide 100% coverage when run with jacoco for the *computeSumOfPrimes* method.

b) Using the test cases generated from evosuite, *mvn test* was able to find the exception in *computeSumOfPrimes*. The test case that checked this was



```
 @Test
public void testComputeSumOfPrimesWithEmptyList() {
    LinkedList<Integer> linkedList = new LinkedList<>();
    int sum = PrimeNumberFinder.computeSumOfPrimes(linkedList);
    assertEquals(0, sum);
}
```