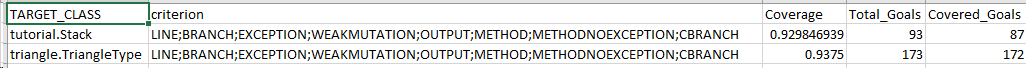
ComS417 Assignment 2

**Problem 1 and Problem 2**

Evosuite downloaded and extracted. Command lines run successfully.

**Problem 3**

1. statistics.csv



1. Number of tests for Triangle: 20

Number of tests for Tutorial: 6

1. Example test

@Test(timeout = 4000)

  public void test17()  throws Throwable  {

      Triangle triangle0 = TriangleType.triangle(435, 435, 435);

      assertEquals(Triangle.EQUILATERAL, triangle0);

  }

1. 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!

         //

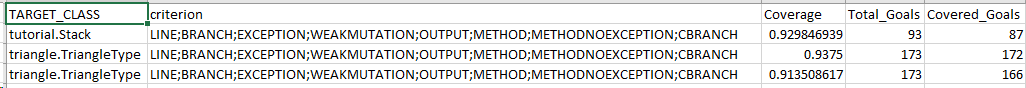
         verifyException("tutorial.Stack", e);

      }

  }

**Problem 4**

1. 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.



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);

  }

1. 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.

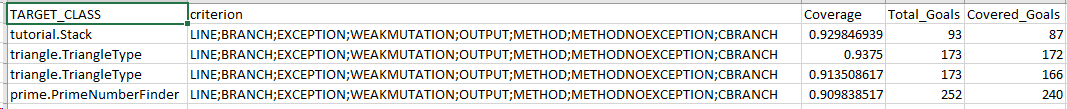
1. Here is a screenshot of the result of running the original tests on the faulty program.

A computer screen shot of a black screen

Description automatically generated

**Problem 5**

1. statistics.csv



1. 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);

  }

  @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();

  }

}

1. Test results

A screen shot of a computer

Description automatically generated

1. 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);

  }

**Problem 6**

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

A screenshot of a computer

Description automatically generated

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

A screen shot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

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

1. 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);

    }