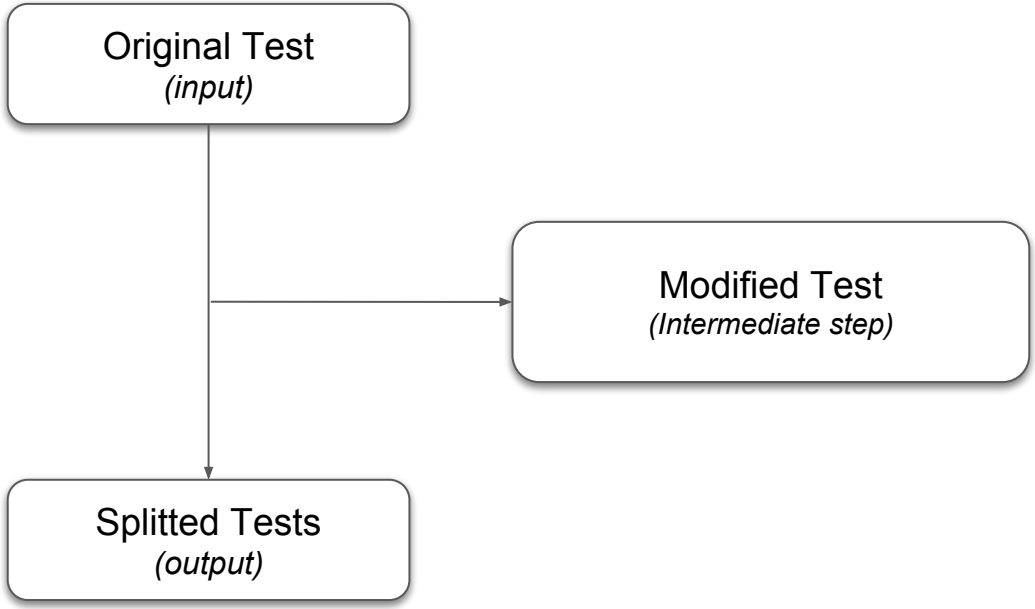
Project: Test-Splitter

Oguz Demir - University of Texas at Austin
oguz@utexas.edu

Draft of September 26, 2018

Summary


Original Test
(input)
Modified Test
(Intermediate step)
Splitted Tests
(output)

Example from JFreeChart Library

Original test:

```
@Test
public void testGetSubTaskCount() {
      Task t1 = new Task("T", new Date(100), new Date(200));
      assertEquals(0, t1.getSubtaskCount());
      t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
      assertEquals(1, t1.getSubtaskCount());
      Task s2 = new Task("S2", new Date(111), new Date(120));
      t1.addSubtask(s2);
      assertEquals(2, t1.getSubtaskCount());
      t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
      assertEquals(3, t1.getSubtaskCount());
      t1.removeSubtask(s2);
      assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Splitted tests:

"TaskTest_testGetSubTaskCount" is changed to **"testName"** for visibility.

```
@Test
public void generatedU1() {
   Task t1 = new Task("T", new Date(100), new Date(200));
   assertEquals(0, t1.getSubtaskCount());
}

@Test
public void generatedU2() {
   Task t1 = (Task) ObjectRecorder.readObject("testName",1);
   // Split Point: 1
   t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
   assertEquals(1, t1.getSubtaskCount());
}

@Test
public void generatedU3() {
   Task t1 = (Task) ObjectRecorder.readObject("testName",2);
   // Split Point: 2
   Task s2 = new Task("S2", new Date(111), new Date(120));
   t1.addSubtask(s2);
   assertEquals(2, t1.getSubtaskCount());
}
```

```
@Test
public void generatedU4() {
   Task s2 = (Task)ObjectRecorder.readObject("testName",3);
   Task t1 = (Task)ObjectRecorder.readObject("testName" , 3);
   // Split Point: 3
   t1.addSubtask(new Task("S3",new Date(121),new Date(130)));
   assertEquals(3, t1.getSubtaskCount());
}

@Test
public void generatedU5() {
   Task s2 = (Task)ObjectRecorder.readObject("testName" , 4);
   Task t1 = (Task)ObjectRecorder.readObject("testName" , 4);
   // Split Point: 4
   t1.removeSubtask(s2);
   assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Modified test which will record the snapshots:

"TaskTest_testGetSubTaskCount" is changed to **"testName"** for visibility.

```java
@Test
public void testGetSubTaskCount() {
   Task t1 = new Task("T", new Date(100), new Date(200));
   assertEquals(0, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", t1, 1);
   ObjectRecorder.finalizeWriting("testName", 1);
   // Split Point: 1
   t1.addSubtask(new Task("S1", new Date(100), new Date(110)));
   assertEquals(1, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", t1, 2);
   ObjectRecorder.finalizeWriting("testName", 2);
   // Split Point: 2
   Task s2 = new Task("S2", new Date(111), new Date(120));
   t1.addSubtask(s2);
   assertEquals(2, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", s2, 3);
   ObjectRecorder.writeObject("testName", t1, 3);
   ObjectRecorder.finalizeWriting("testName", 3);
   // Split Point: 3
   t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
   assertEquals(3, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", s2, 4);
   ObjectRecorder.writeObject("testName", t1, 4);
   ObjectRecorder.finalizeWriting("testName", 4);
   // Split Point: 4
   t1.removeSubtask(s2);
   assertEquals(2, t1.getSubtaskCount());
}
```

Example from JFreeChart Library

Snapshots taken:

```
@Test
public void testGetSubTaskCount() {
   Task t1 = new Task("T",new Date(100),new Date(200));
   assertEquals(0, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", t1, 1);
   ObjectRecorder.finalizeWriting("testName", 1);
   // Split Point: 1
   t1.addSubtask(new Task("S1",new Date(100),new Date(110)));
   assertEquals(1, t1.getSubtaskCount());
   ObjectRecorder.writeObject("testName", t1, 2);
   ObjectRecorder.finalizeWriting("testName", 2);
   // Split Point: 2
   ..
   ..
   ..
}
```

```
<org.jfree.data.gantt.Task>
  <description>T</description>
  <duration class="org.jfree.data.time.SimpleTimePeriod">
    <start>100</start>
    <end>200</end>
  </duration>
  <subtasks/>
</org.jfree.data.gantt.Task>
```

```
<org.jfree.data.gantt.Task>
  <description>T</description>
  <duration class="org.jfree.data.time.SimpleTimePeriod">
    <start>100</start>
    <end>200</end>
  </duration>
  <subtasks>
    <org.jfree.data.gantt.Task>
      <description>S1</description>
      <duration class="org.jfree.data.time.SimpleTimePeriod">
        <start>100</start>
        <end>110</end>
      </duration>
      <subtasks/>
    </org.jfree.data.gantt.Task>
  </subtasks>
</org.jfree.data.gantt.Task>
```

Example from Commons-Codec Library

Original test:

```
@Test
public void testMd2HexLength() {
   String hashMe = "this is some string that is longer than 32 characters";
   String hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
   hashMe = "length < 32";
   hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
}
```

Example from Commons-Codec Library

Splitted tests:

"DigestUtilsTest testMd2HexLength" is changed to "testName" for visibility.

```
@Test
public void generatedU1() {
    String hashMe = "this is some string that is longer than 32 characters";
    String hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
}

@Test
public void generatedU2() {
    String hash = (String) ObjectRecorder.readObject( "testName", 1);
    String hashMe = (String) ObjectRecorder.readObject( "testName", 1);
    // Split Point: 1
    hashMe = "length < 32";
    hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
    assertEquals(32, hash.length());
}
```

Example from Commons-Codec Library

Modified test which will record the snapshots:

```
@Test
public void testMd2HexLength() {
   String hashMe = "this is some string that is longer than 32
characters";
   String hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
   ObjectRecorder.writeObject( "testName", hash, 1);
   ObjectRecorder.writeObject( "testName", hashMe, 1);
   ObjectRecorder.finalizeWriting( "testName", 1);
   // Split Point: 1
   hashMe = "length < 32";
   hash = DigestUtils. md2Hex(getBytesUtf8(hashMe));
   assertEquals(32, hash.length());
}
```

"DigestUtilsTest testMd2HexLength" is changed to **"testName"** for visibility.

Current Results with 5 Different Open-Source Projects

	Original #Tests	#Tests after Split	Total size of Snapshots	#Snapshots
JFreeChart	2176	6620	212.1 MB	4444
Commons-lang	4417	6856	23.1 MB	2175
commons-codec	887	1208	2.6 MB	312
commons-net	287	292	20 KB	6
commons-io	1349	1648	971 KB	167

- All modified tests are executed to generate all snapshots.

Current Results with 5 Different Open-Source Projects

	ORIGINAL			SPLITTED						
	#Tests	Failures	Errors	#Tests	Failures	Errors	Original Time	Instr. Time	Snapshot Generation Time	Splitted Tests Time
JFreeChart	2176	0	0	6620	8	0	6.462	5.324	20.708	23.806
commons-lang	4417	0	0	6856	24	22	25.546	7.717	31.755	27.67
commons-codec	887	0	0	1208	0	0	13.802	3.123	15.753	16.422
commons-net	287	0	0	292	0	0	85.734	1.423	87.051	83.94
commons-io	1349	0	0	1648	50	0	70.52	3.567	83.503	79.902

- Original time: Time to execute all tests without modification
- Instrumentation time: Time to create modified and splitted tests
- Generation time: Time to generate snapshots from modified tests.
- Splitted tests time: Time to execute all splitted tests

Optimization Ideas: Simplification

```
@Test
public void generatedU4() {
  Task s2 = (Task) ObjectRecorder.readObject("testName" , 3);
  Task t1 = (Task) ObjectRecorder.readObject("testName" , 3);
  // Split Point: 3
  t1.addSubtask(new Task("S3", new Date(121), new Date(130)));
  assertEquals(3, t1.getSubtaskCount());
}
```

- Task s2 object is written to disk in snapshot generation and read back from disk for executing generatedU4.
- However, s2 object is not used during the execution of generatedU4.
- Therefore, this object can be omitted from both snapshot file and method generatedU4.

Configuring Test Splitter

Options	
-p	Path to test file
-c (optional)	Target class name that includes the methods to be splitted. All the test class files will be processed by default.
-t (optional, repeated)	Target method name(s) to split. All methods with (**@Test**) annotation will be processed by default.
-s (optional, repeated)	Split points (method names). All method calls in the test function will be considered as split point.
-a	Enabling splitting in assertions rather than method calls.