Assignment HW4

**Cover Page**

Prepared for:

Dr. Mehra Borazjany

Trung Hieu Tran

Prepared by:

Alex Lundin

Ali Haider

SE-4367.0U1-Testing

**Assignment Choice:**

N/A for this assignment

18 June, 2018

**Proof of Working Software**

GitHub link:

<https://github.com/AlexLundinEducational/SE-4367-Testing>

Branch Summary:

master – managed by Alex, only fully completed pulls allowed to make TA’s life easy. Master only contains assignment material once they reach completed status.

working – flexible branch for team, ideally, this material should build without causing technical debt during the project.

Commit for grading:

HW4\_TEAM1 Complete, Ready for Merge to master and Ready for Grading

Phase 1 Development

06-18-2018

Ali

JUnit test flaw

Wrote intro for part 2 a

Evaluation of test class in part 2 b

Phase 2 Development

06-19-2018

Alex

JUnit testing for facts

Wrote test cases for search 1a

Answered questions on controllability and observability 1b

Proof of Working Tests (10 seconds)

<https://autode.sk/2taZEfw>

**Documentation Log**

1. Reductions
   1. File 1

* Removed a few import statements in the FactServlet.java that caused compilation errors

1. Refactors
   1. StringConstants.java

* this is a refactoring of the strings used in the JavaServlet.java file

1. Modifications
   1. testRemoveDuplicate()

* Modified the original source code and added toString() method to set an int value of the array before and after removing all duplicates

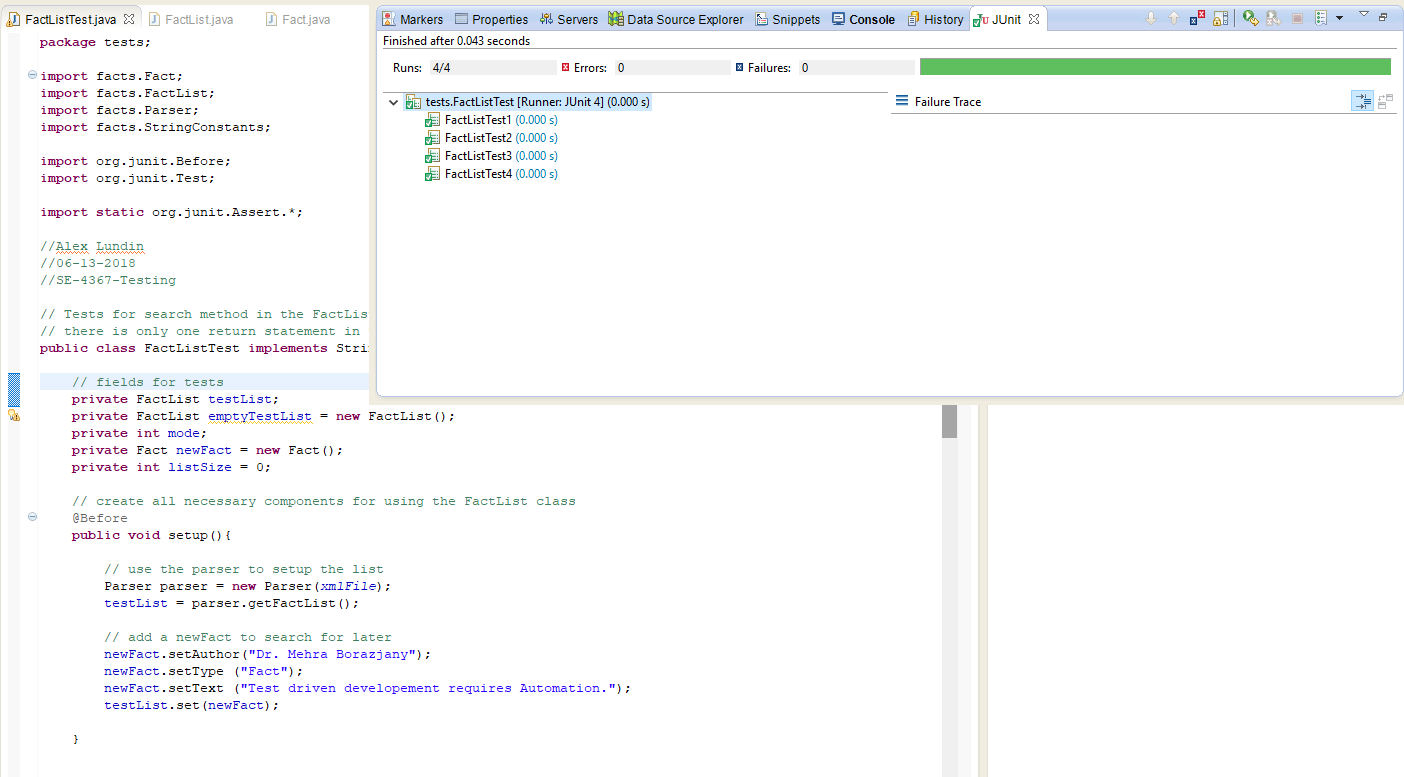
1. Additions
   1. FactListTest.java

* Used to test the search method in FactList.java
* setup method to create the list before All Tests
* 4 test methods
* Covers all return statements in FactList.java search method

**Junit Testing for Facts**

**A.**

**Image of test completion**

****

**B.**

**Observability - How easy it is to observe the behavior of a program in terms of its outputs, effects on the environment and other hardware and software components (s8-Ch03-automation Slide 4).**

Breakdown by method name

setup - this does not affect the observability at all

FactListTest1 - the getSize method allows the test to check how many Facts returned in the search

FactListTest2 - the getSize method allows the test to check how many Facts returned in the search

FactListTest3 - the getSize method allows the test to check how many Facts returned in the search

FactListTest4 - the getSize method allows the test to check how many Facts returned in the search

**C.**

**Controllability – How easy it is to provide a program with the needed inputs, in terms of values, operations, and behaviors (s8-Ch03-automation Slide 4).**

Breakdown by method name

setup - the setup addresses the components necessary to provide the FactList with inputs.

FactListTest1 - sets fact search mode to 0

FactListTest2 - sets fact search mode to 1

FactListTest3 - sets fact search mode to 2

FactListTest4 - sets fact search mode to 3

**Junit Test Flaw**

**A.**

The RIPR model states 4 conditions (reachability, infection, propagation, reveal)

necessary for a failure to be observed. The first condition we observe is reachability and since it

is stated that the test method for removeDuplicate() compiles and runs to completion we can

say that the location of the fault has been reached. The problem with the test method have to do

with the infection and propagation conditions. This means that the state of the the program must

be incorrect and the state must cause some incorrect output. With the ‘add’ operation, strings

are stored in the order that they are inserted which in this case [“orange”, “apple”, “apple”,

“banana”]. As mentioned, the order of string should be maintained when the elements are

withdrawn, so if it is assumed that removeDuplicates() works as it is described, the final state of

the program should be [“orange”, “apple”, “banana”]. However this is not the case. The test

method uses method assertTrue if the first word is “orange”. Since order is maintained when

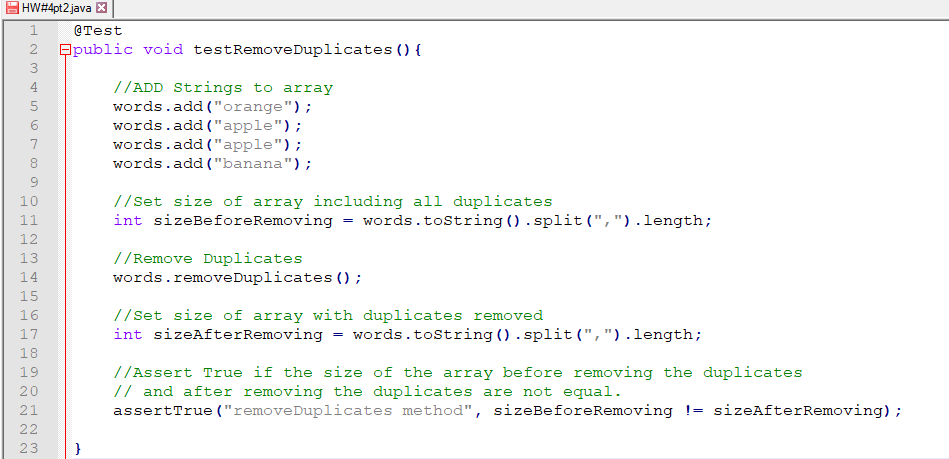
strings are added and withdrawn, the first element returned from getFirst() will remain orange,

even if the removeDuplicate() works without error. Therefore, the final state of the program will

be correct, making the current test method incorrect for testing removeDuplicate().

**B.**

**Image of code**



**B.**

**Actual source code**

@Test

public void testRemoveDuplicates(){

//ADD Strings to array

words.add("orange");

words.add("apple");

words.add("apple");

words.add("banana");

//Set size of array including all duplicates

int sizeBeforeRemoving = words.toString().split(",").length;

//Remove Duplicates

words.removeDuplicates();

//Set size of array with duplicates removed

int sizeAfterRemoving = words.toString().split(",").length;

//Assert True if the size of the array before removing the duplicates

// and after removing the duplicates are not equal.

assertTrue("removeDuplicates method", sizeBeforeRemoving != sizeAfterRemoving);

}