SWE 437-001

Software Testing and Maintenance

Assignment 6: Test Driven Development

*Students:*

Behrad Behmardi

Pisith Yim

*Professor:*

Dr. Jeff Offutt

Git Repository: http://github.com/behmardibehrad/SWE-437-001-Assignment-1.git

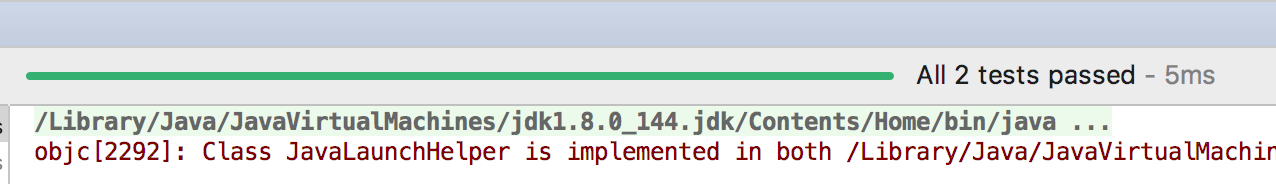
**User story 1**: Users can create keyword and add to a particular quote

**Test 1**: Check to see if keyword is created successfully. (TestKeyword.java, Keyword.java)

**public class** TestKeyword {  
 **int counter**;  
 Keyword **newKeyword**;  
 @Before  
 **public void** setUp(){  
 **counter** = Keyword.*counter*;  
 **newKeyword** = **new** Keyword();  
 }  
 @Test  
 **public void** testCreateKeyWord(){  
 **newKeyword**.setKeyword(**"New keyword"**);  
 *assertEquals*(**counter** + **": New keyword"**, **newKeyword**.toString());  
 }  
  
 @Test  
 **public void** testCreateDiffKeyWord(){  
 **newKeyword**.setKeyword(**"Another keyword"**);  
 *assertEquals*(**counter** + **": Another keyword"**, **newKeyword**.toString());  
 }  
  
 @After  
 **public void** tearDown(){  
 Keyword.*counter* = -1;  
 **newKeyword** = **null**;  
 }  
}

**Description**: At first we didn’t have *Keyword* class, where we can instantiate the *newKeyword* object, we wrote the test first. When we tried to run this test, it gave a compiler error. In order to make the compiler happy, we created the *Keyword.java* class with the necessary constructor and *setKeyword* method. After creating the Keyword class, the compiler is now happy, but our test failed. We then proceeded to fix our Keyword class by implementing the logic to set new keyword. When we tried to run the test again, the test now passes. To check that this also works with a different input, we add another test with a different input. We ran the test again, the both tests are now passed.

**Refactor:** TestKeyword.java, we added @Before and @After, we added variables and objects that will be useful for each test. In Keyword.java we organized the code, added comments, added static counter to identify each keyword id(kid). We made sure everything still works and passes.



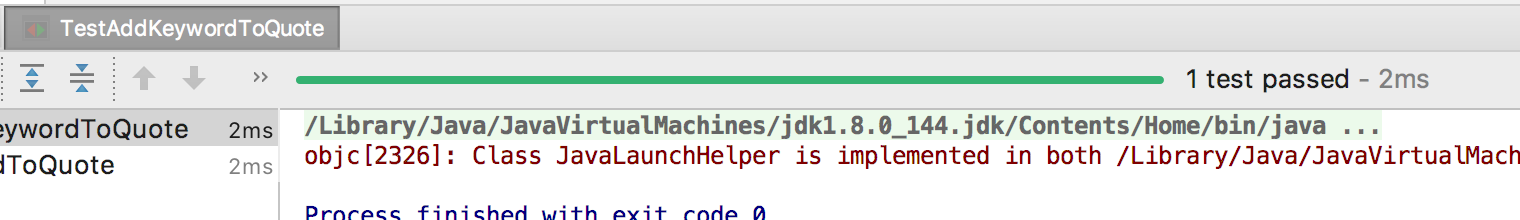
**Test 2**: Check to see if we can add keyword to quote. (TestAddKeywordToQuote.java, Quote.java)

**public class** TestAddKeywordToQuote {  
 Keyword **newKeyword**;  
 Quote **newQuote**;  
 @Before **public void** setUp(){  
 **newKeyword** = **new** Keyword();  
 **newKeyword**.setKeyword(**"New keyword"**);  
 **newQuote** = **new** Quote();  
 **newQuote**.addKeyword(**newKeyword**);  
 }  
  
 @Test **public void** testAddToQuote(){  
 *assertTrue*(**newQuote**.isKeyword(**newKeyword**));  
 }

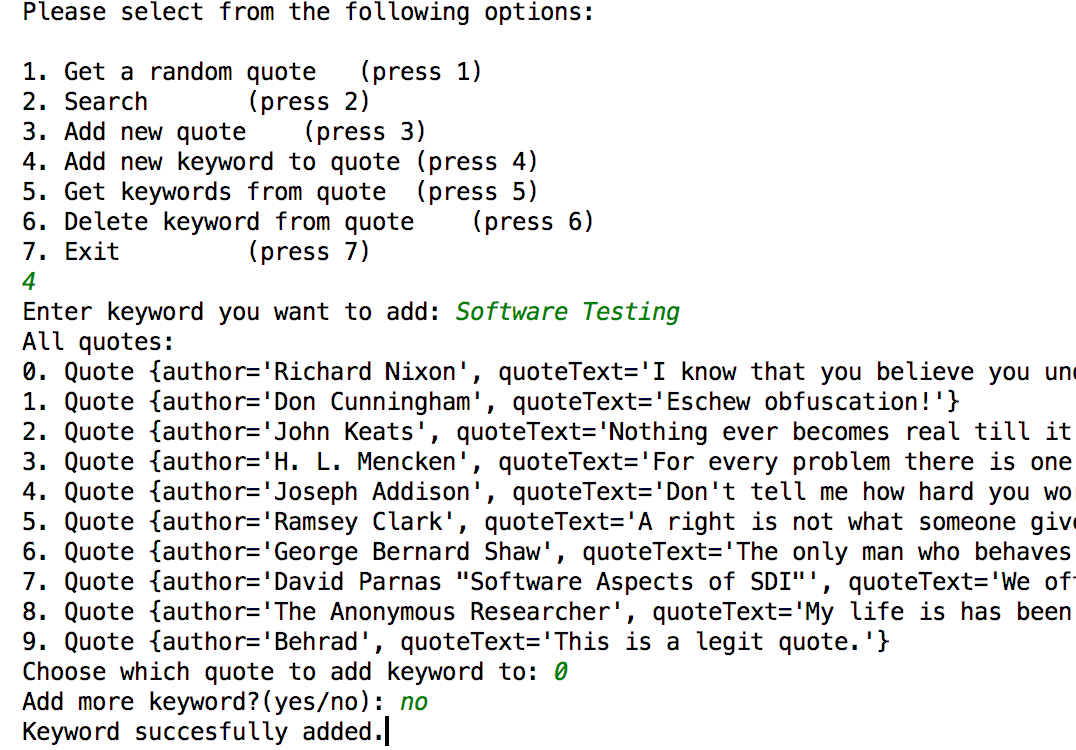
@After **public void** tearDown(){  
 **newQuote** = **null**;  
 **newKeyword** = **null**;  
 }

}

**Description**: This test assume that user can create a new keyword (see Test 1 above). Then we write *addKeyWord* that can be called from *newQuote* object. This will cause the compiler error, because *addKeyWord* method hasn’t been created yet. We added the *addKeyWord* method with minimum code in Quote.java to make the compiler happy, but the test failed. We added the logic to add a keyword to a quote by introducing an *ArrayList* that holds keywords in quote object. (see quote.java). We ran the test again, the test now passes. We refactor our test class by adding @Before and @After method, and our quote class by adding comments and checking the quote class still works, ran test once again, the test still passed.



**Screenshot of new feature user story 1:** User create keyword and added to a quote



**User story 2**: Users can delete keyword from a quote.

**Test 1**: Check to see if we can delete keyword to quote. (TestAddKeywordToQuote.java, Quote.java)

**public class** TestAddKeywordToQuote {  
 Keyword **newKeyword**;  
 Quote **newQuote**;  
 @Before **public void** setUp(){  
 **newKeyword** = **new** Keyword();  
 **newKeyword**.setKeyword(**"New keyword"**);  
 **newQuote** = **new** Quote();  
 **newQuote**.addKeyword(**newKeyword**);  
 }

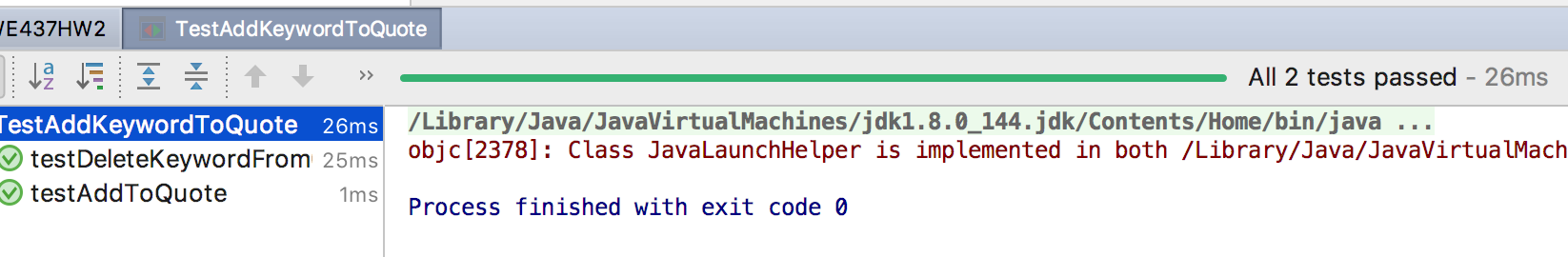
@Test **public void** testDeleteKeywordFromQuote(){  
 **newQuote**.deleteKeyword(**newKeyword**);  
 *assertFalse*(**newQuote**.isKeyword(**newKeyword**));

}

@After **public void** tearDown(){  
 **newQuote** = **null**;  
 **newKeyword** = **null**;  
 }

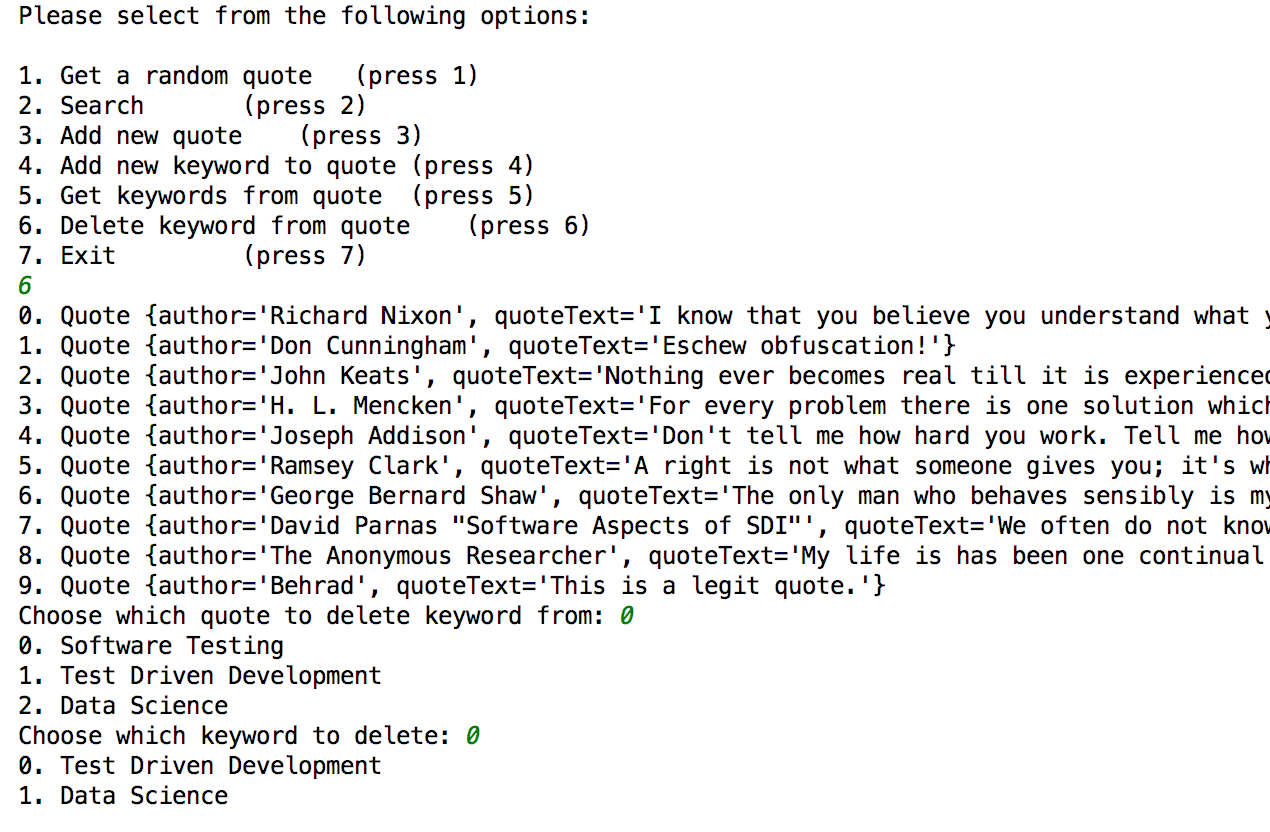
}

**Description:** This test assume that user can add a keyword to a quote (see user story 1, test 2). Our initial code is adding *deleteKeywordFromQuote* method, the compiler will throw a compiler error when the test is run. We fixed this by adding this method with no code inside to *quote.java.* The test now compiles, but failed. We added our logic to *deleteKeywordFromQuote* method in *quote.java.* Run the test again, it passed. We added @Before and @After to our test class for refactoring, then we fixed up our variables name for easier and more readable code. Tested again, test passed.



**Screenshot of working user story 2:** Users can delete keyword from a quote.

Note: This test assumes that Software Testing, Test Driven Development, and Data Science has been added to quote, “I know that you belive…”. Screenshot shows deletion of Software Testing keyword.

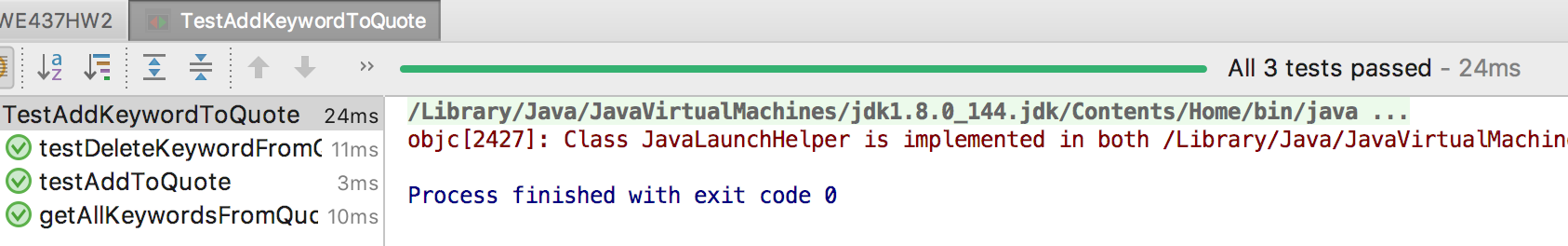


**User story 3**: Users can get all the keywords from a particular quote.

**Test 1**: Check to see if we can get all keywords from a quote. (TestAddKeywordToQuote.java, Quote.java)

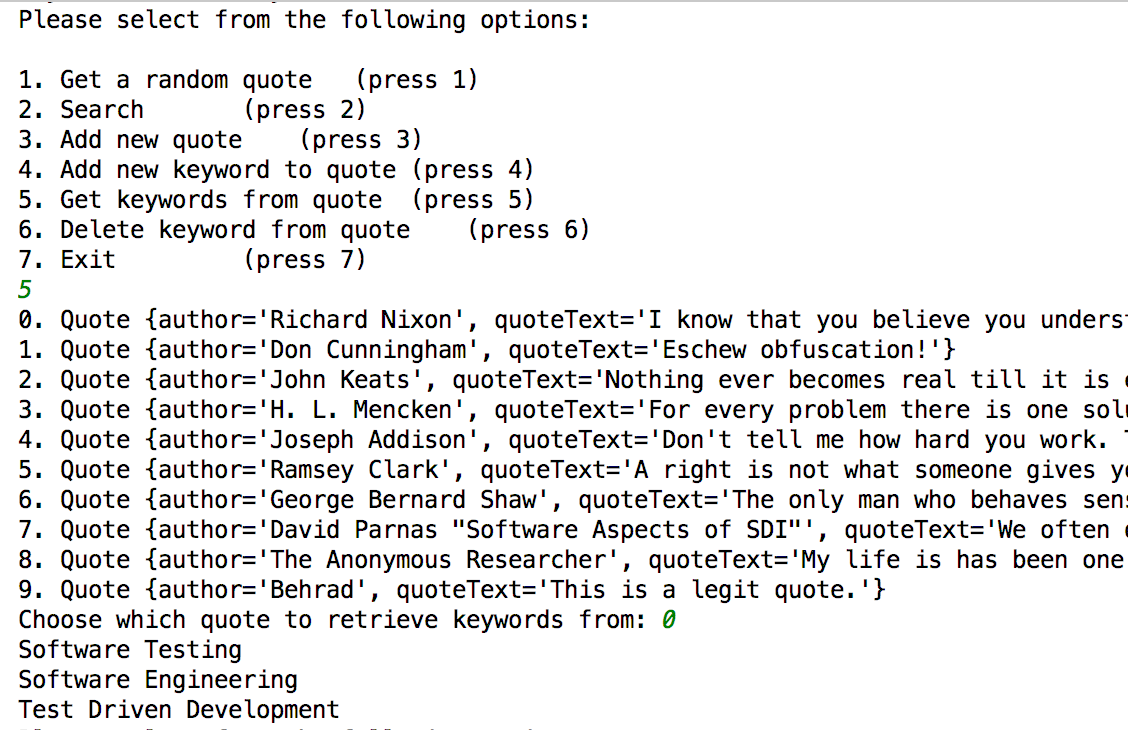
**public class** TestAddKeywordToQuote {  
 Keyword **newKeyword**;  
 Quote **newQuote**;  
 @Before **public void** setUp(){  
 **newKeyword** = **new** Keyword();  
 **newKeyword**.setKeyword(**"New keyword"**);  
 **newQuote** = **new** Quote();  
 **newQuote**.addKeyword(**newKeyword**);  
 }  
  
 @Test **public void** getAllKeywordsFromQuote(){  
 Keyword firstKey = **new** Keyword();  
 firstKey.setKeyword(**"First"**);  
 Keyword secondKey = **new** Keyword();  
 secondKey.setKeyword(**"Second"**);  
 Keyword thirdKey = **new** Keyword();  
 thirdKey.setKeyword(**"Third"**);  
 **newQuote**.addKeyword(firstKey);  
 **newQuote**.addKeyword(secondKey);  
 **newQuote**.addKeyword(thirdKey);  
 ArrayList<Keyword> result = **newQuote**.getAllKeywords();  
 ArrayList<Keyword> testResult = **new** ArrayList<>();  
 testResult.add(**newKeyword**);  
 testResult.add(firstKey);  
 testResult.add(secondKey);  
 testResult.add(thirdKey);  
 *assertEquals*(testResult, result);  
 }  
  
 @After **public void** tearDown(){  
 **newQuote** = **null**;  
 **newKeyword** = **null**;  
 }  
}

**Description**: First we add *getAllKeywords* to test, this will supposedly retrieve all of the keywords from a quote object. For now this method won’t compile, because *getAllKeywords* hasn’t been created yet. We fixed this by adding *getAllKeywords* without any code insideto *quote.java.* The test now compiles but failed. We added logic in getAllKeywords method in quote.java to return related keywords back from quote object. Refactoring part of the test was previously done by tests in user story 1 and user story 2. We also added comments for more readable code. Ran the test after refactoring, all tests passed.



**Screenshot of working user story 3:** Users can get all the keywords from a particular quote.

Note: This test assumes that Software Testing, Software Engineering, and Test Driven Development keywords have been added to quote “I know that you believe…” by Richard Nixon. See User story 1 for adding keywords to a quote.

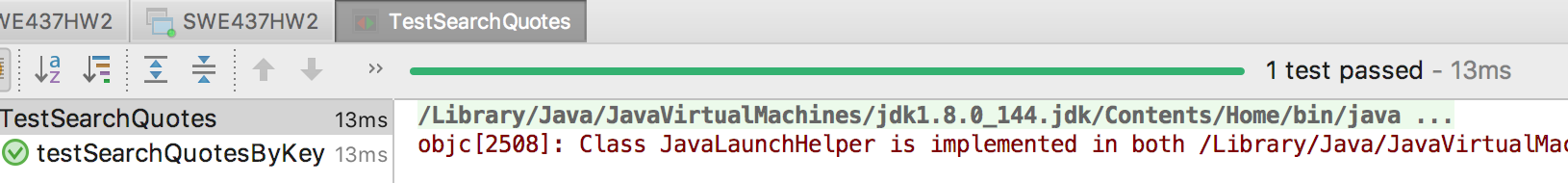


**User story 4**: Users can search for quotes based on given keywords.

**Test 1**: Check to see if we can search quotes by keyword. (TestSearchQuotes.java, QuoteList.java, SWE437HW2.java)

**public class** TestSearchQuotes {  
 QuoteSaxParser **qParser** = **new** QuoteSaxParser (**"src/quotes.xml"**);  
 QuoteList **quoteList** = **qParser**.getQuoteList();  
  
 @Test **public void** testSearchQuotesByKeyword(){  
 Keyword testKeyword = **new** Keyword();  
 testKeyword.setKeyword(**"Test Driven Development"**);  
 QuoteList expectedQuotes = **new** QuoteList();  
 **for**(**int** i = 0; i < **quoteList**.getSize(); i++){  
 String author = **quoteList**.getQuote(i).getAuthor();  
 **if**(author.equalsIgnoreCase(**"Richard Nixon"**)  
 || author.equalsIgnoreCase(**"Don Cunningham"**)  
 || author.equalsIgnoreCase(**"Ramsey Clark"**)){  
 expectedQuotes.setQuote(**quoteList**.getQuote(i));  
 **quoteList**.getQuote(i).addKeyword(testKeyword);  
 }  
 }  
 QuoteList actualQuotes = **quoteList**.searchQuotesByKeyword(**"Test Driven Development"**);  
 *assertEquals*(expectedQuotes.getQuoteArray(), actualQuotes.getQuoteArray());  
  
 }  
}

**Description**: First, this test sets up so that keyword is added to quotes that have certain author name, just for testing purposes. *expectedQuotes* list is also added for testing purposes, by comparing with the *actualList* later in the assertion. *searchQuotesByKeyword* should return all the related quotes that have “Test Driven Development” keyword. At the moment, the test doesn’t compile because *searchQuotesByKeyword* method doesn’t exist. We fixed this by adding *searchQuotesByKeyword* with return null inside the method body. The test now compiles, but failed. We then added our logic of searching for quotes that contain our given string as keyword. Since there were no method to compare QuoteList object, we created a helper method of returning the QuoteList as an ArrayList in order to do the assertion. The test now passed. We refactored our test by adding @Before and @After, so that we can instantiate the necessary variables and methods before each test in the future. After that we ran test again to make sure test still pass.



**Screenshot of new feature user story 4:**

This test assumes that keyword “software testing” has been added to quotes. See user story 1 for adding keyword to a quote. In this example, “software testing” keyword has been added to 3 quotes.

1. I know that you believe … by Richard Nixon
2. Eschew obfuscation, by Don Cunningham
3. Nothing ever becomes… by Jon Keats

