Nicholas Laviano

Software Engineering Project

SE 4560

## Introduction

The goal of this project is to create 10 practical and meaningful test cases that demonstrate different testing techniques learned in class and can be used on real open-source codebases. I selected two different open-source projects to test: assertJ and commons-text. These projects offer clear helper classes with different logic that are worth testing.

For each project, I forked it from the forked ST-Spring-25, which creates a copy of the forked repository onto my personal repository. I also created a branch for each project called “add-test-cases” and I committed each test to the branch. These tests use different testing techniques and target different logical behaviors in each project.

I use Maven to ensure each test passes or acts as it should, to ensure that the tests are properly made, and to test what is needed. I created a pull request for each branch on their project. Lastly, I will summarize my group and I’s weekly check-ins and share the help I gave as well as the help that I got.

## Description of assertj

For my first 6 test cases, I am testing different methods in assertJ. assertJ is a fluent-assertion library for Java tests that replaces the traditional assertEquals checks with better readable and chainable assertions. It provides many features that allow developers to make code clearer and more maintainable while it finds errors and provides informative feedback. Different features that assertj provides are helpful failure messages, fluent API, type-specific assertions, and no external runner needed.

Here is the link for the pull request made for assertj:

https://github.com/ST-Spring-25/assertj/pull/1

**Test Case #1:**

*What is the class and method I am testing:*

File: AbstractIterableAssert.java

- https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/api/AbstractIterableAssert.java
- (~\assertj\assertj-core\src\main\java\org\assertj\core\api\AbstractIterableAssert.java)

Class: The AbstractIterableAssert class is a base assertion class for all iterable types in assertj

Methods: The methods I am testing are filteredOn(Predicate<E>) and doesNotContainNull(). The purpose of filteredOn(Predicate<E>) is to return a new assertion object containing only elements that match the predicate. My test will focus on filtering a list of names with a given letter, and checking if the name results with the given letter. The purpose of doesNotContainNull() is that it will fail if any element in the iterable is *null*. My test will focus on asserting a list with no *null* values.

*What is the test:*

File: AbstractIterableAssertTest.java

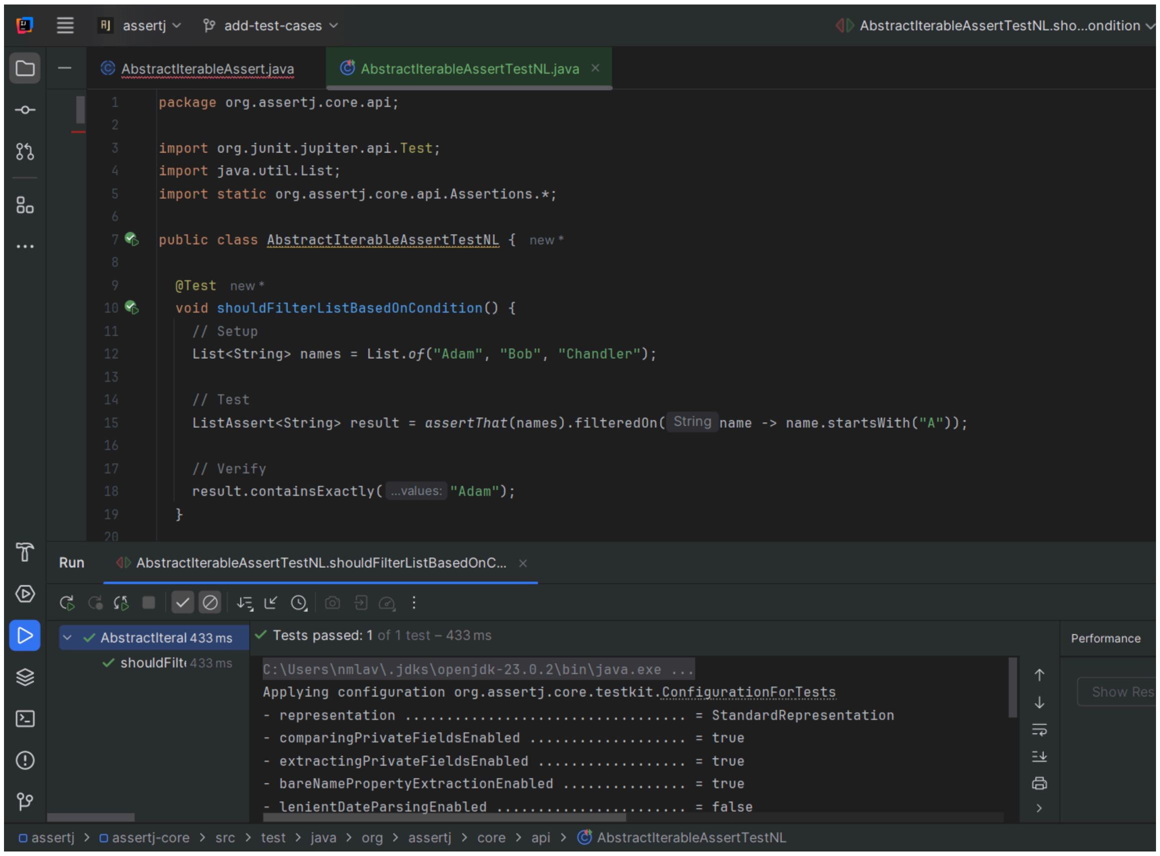
- https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/AbstractIterableAssertTest.java
- (~\assertj\assertj-core\src\test\java\org\assertj\core\api\AbstractIterableAssertTest.java)

Structure: The structure of these 2 tests is basic anatomy of a test (setup → test → verify). It uses JUnit as well.

Technique: The technique is simple black-box testing that tests one functionality for each method. I also

Why: This is a good test because it tests 2 high-level iterable operations that users rely on daily. It also proves positive and safe behavior while using assertj's fluent assertions.

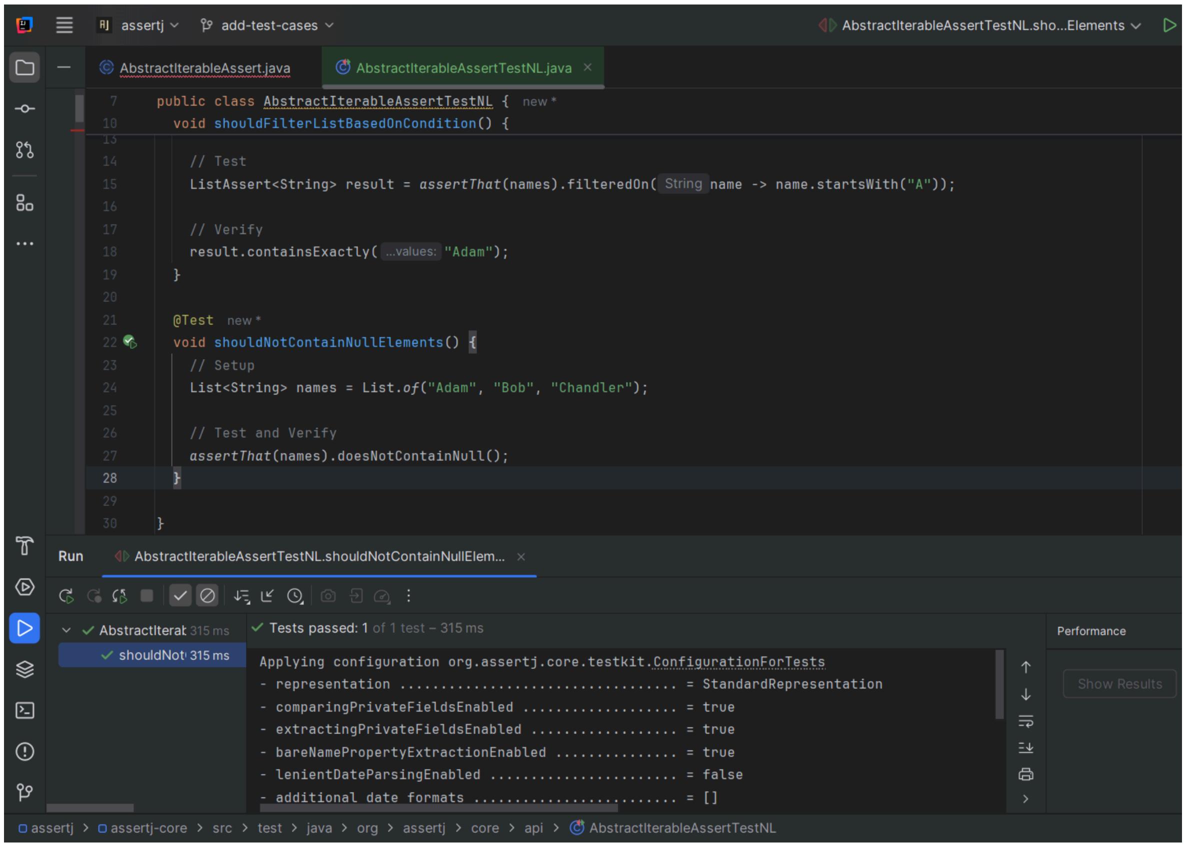
Here are the pictures of the tests passing and evidence of running it through Maven


assertj
add-test-cases
AbstractIterableAssertTestNL.sho...ondition
AbstractIterableAssert.java
AbstractIterableAssertTestNL.java
Run
AbstractIterableAssertTestNL.shouldFilterListBasedOnC...
AbstractIteral 433 ms
shouldFilt 433 ms
Performance
Show Res
assertj › assertj-core › src › test › java › org › assertj › core › api › AbstractIterableAssertTestNL

```java
package org.assertj.core.api;

import org.junit.jupiter.api.Test;
import java.util.List;
import static org.assertj.core.api.Assertions.*;

public class AbstractIterableAssertTestNL {  new *

  @Test  new *
  void shouldFilterListBasedOnCondition() {
    // Setup
    List<String> names = List.of("Adam", "Bob", "Chandler");

    // Test
    ListAssert<String> result = assertThat(names).filteredOn( String name -> name.startsWith("A"));

    // Verify
    result.containsExactly( ...values: "Adam");
  }
```

Tests passed: 1 of 1 test – 433 ms

```
C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...
Applying configuration org.assertj.core.testkit.ConfigurationForTests
- representation ................................ = StandardRepresentation
- comparingPrivateFieldsEnabled .................. = true
- extractingPrivateFieldsEnabled ................. = true
- bareNamePropertyExtractionEnabled .............. = true
- lenientDateParsingEnabled ...................... = false
```

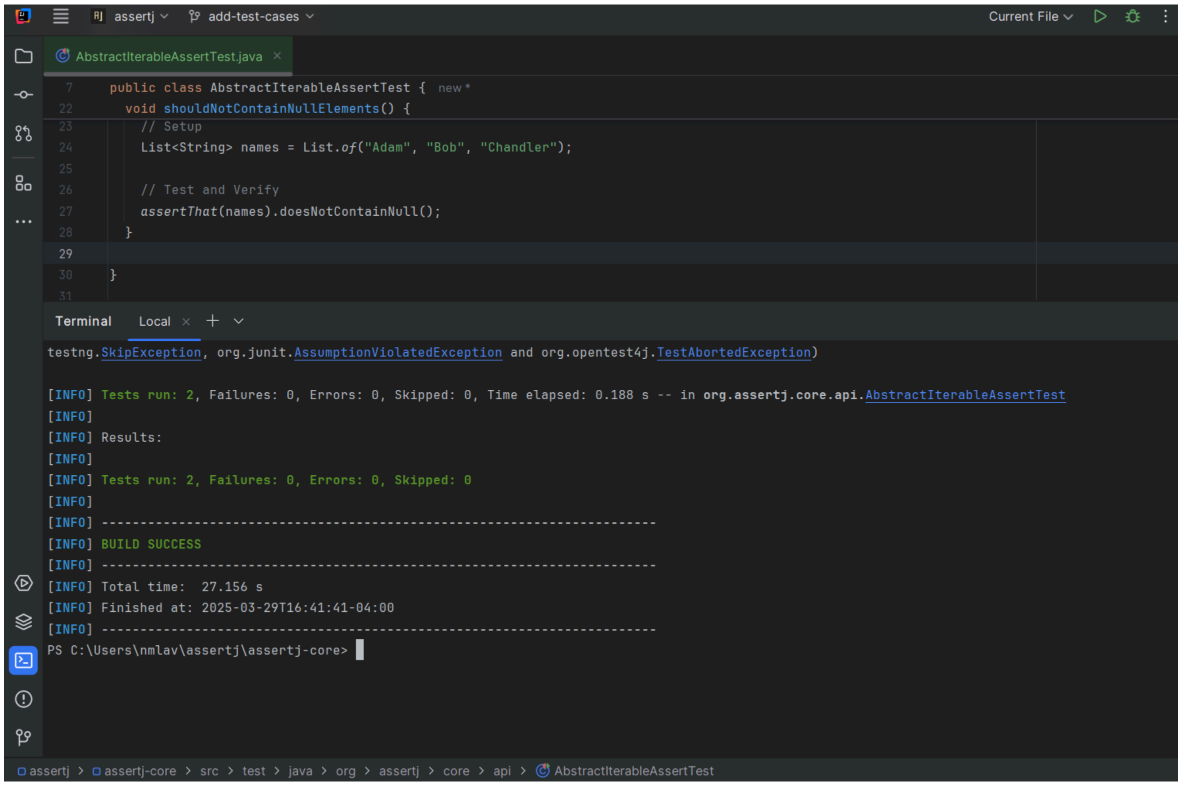


assertj
add-test-cases
AbstractIterableAssertTestNL.sho...Elements
AbstractIterableAssert.java
AbstractIterableAssertTestNL.java

```java
public class AbstractIterableAssertTestNL {  new *
  void shouldFilterListBasedOnCondition() {

    // Test
    ListAssert<String> result = assertThat(names).filteredOn( String name -> name.startsWith("A"));

    // Verify
    result.containsExactly( ...values: "Adam");
  }

  @Test  new *
  void shouldNotContainNullElements() {
    // Setup
    List<String> names = List.of("Adam", "Bob", "Chandler");

    // Test and Verify
    assertThat(names).doesNotContainNull();
  }

}
```

```
Tests passed: 1 of 1 test – 315 ms

Applying configuration org.assertj.core.testkit.ConfigurationForTests
- representation ................................ = StandardRepresentation
- comparingPrivateFieldsEnabled ................. = true
- extractingPrivateFieldsEnabled ................ = true
- bareNamePropertyExtractionEnabled ............. = true
- lenientDateParsingEnabled ..................... = false
- additional date formats ....................... = []
```

assertj > assertj-core > src > test > java > org > assertj > core > api > AbstractIterableAssertTestNL



assertj
add-test-cases
Current File
AbstractIterableAssertTest.java

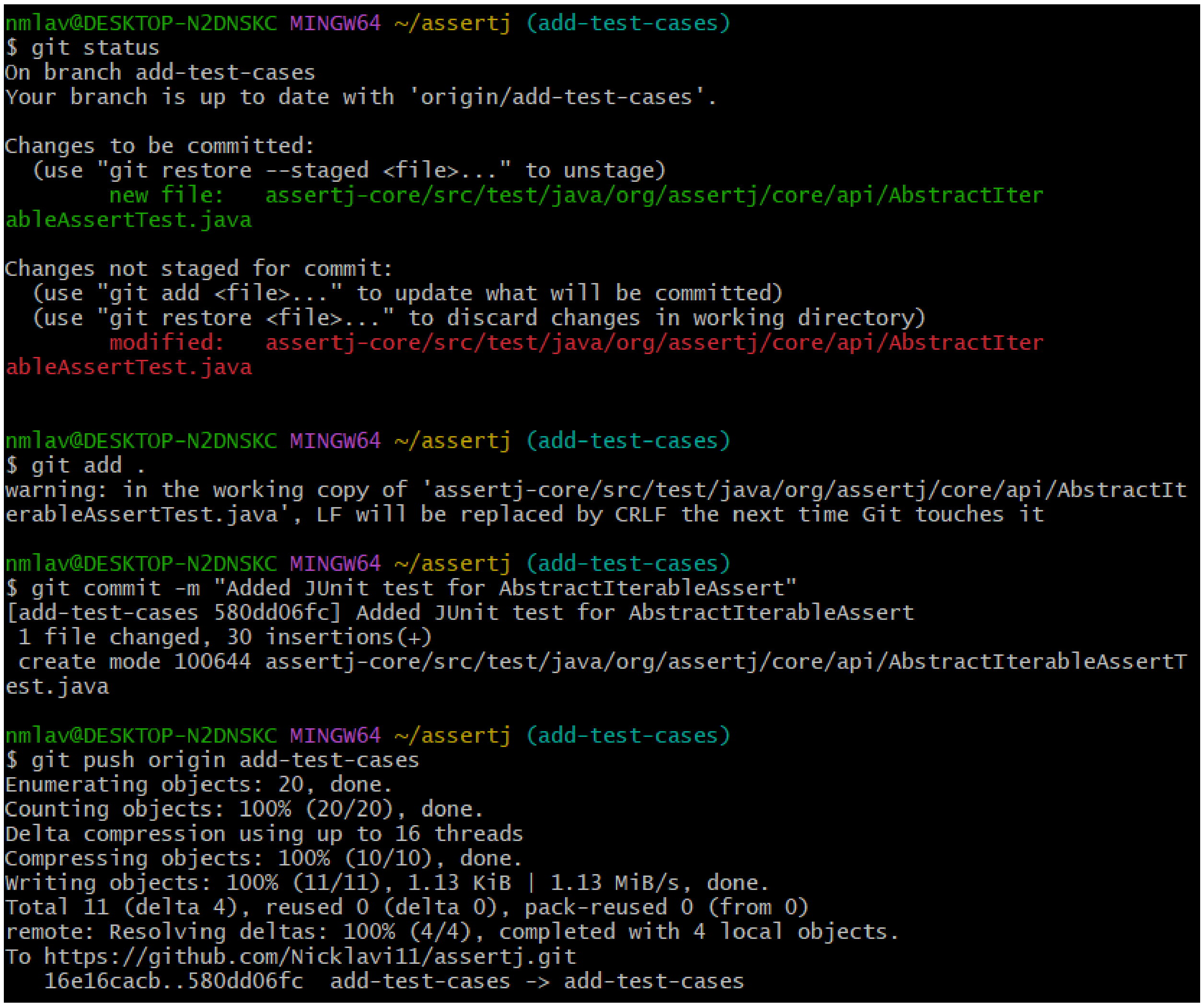
```java
public class AbstractIterableAssertTest {  new *
  void shouldNotContainNullElements() {
    // Setup
    List<String> names = List.of("Adam", "Bob", "Chandler");

    // Test and Verify
    assertThat(names).doesNotContainNull();
  }

}
```

```
testng.SkipException, org.junit.AssumptionViolatedException and org.opentest4j.TestAbortedException)

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.188 s -- in org.assertj.core.api.AbstractIterableAssertTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  27.156 s
[INFO] Finished at: 2025-03-29T16:41:41-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\assertj\assertj-core>
```

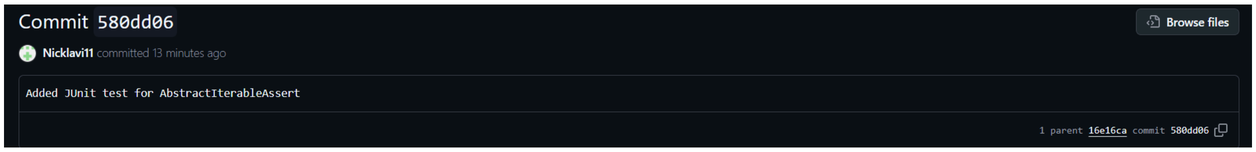
assertj > assertj-core > src > test > java > org > assertj > core > api > AbstractIterableAssertTest

*Commit:*

After the tests passed, I committed it to the repository:



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git status
On branch add-test-cases
Your branch is up to date with 'origin/add-test-cases'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   assertj-core/src/test/java/org/assertj/core/api/AbstractIter
ableAssertTest.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   assertj-core/src/test/java/org/assertj/core/api/AbstractIter
ableAssertTest.java


nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/api/AbstractIt
erableAssertTest.java', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "Added JUnit test for AbstractIterableAssert"
[add-test-cases 580dd06fc] Added JUnit test for AbstractIterableAssert
 1 file changed, 30 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/api/AbstractIterableAssertT
est.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.13 KiB | 1.13 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   16e16cacb..580dd06fc  add-test-cases -> add-test-cases
```



Commit 580dd06

Nicklavi11 committed 13 minutes ago

```
Added JUnit test for AbstractIterableAssert
```

1 parent 16e16ca commit 580dd06

Link for this commit:

https://github.com/ST-Spring-25/assertj/pull/1/commits/580dd06fcb2e8a763b4981ccc04efc58f36fd43f

## **Test Case #2**

*What is the class and method I am testing:*

File: Strings.java

- [https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/util/Strings.java](https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/util/Strings.java)
- (~/assertj/assert-core/src/main/java/org/assertj/core/util/Strings.java)

Class: The Strings class is a utility class that provides helper methods that are used with strings. It helps with formatting strings, normalizing whitespaces, and checking if a string is empty or null. These methods help make assertions simpler in testing frameworks and allow for reusable logic related to strings.

Method: The method I am testing isNullOrEmpty(), which is a method that is used to check if a string is empty or null. My test will focus on making sure it behaves correctly for 3 possible inputs.

*What is the test:*

File: StringEmptyOrNullTest.java

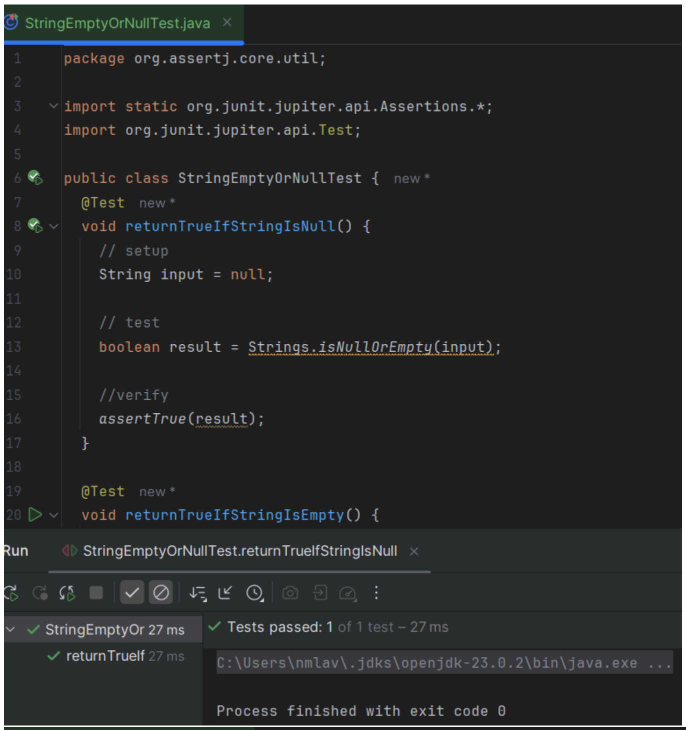
- [https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.java](https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.java)
- (~\assertj\assertj-core\src\test\java\org\assertj\core\util\StringEmptyOrNullTest.java)

Structure: The structure of these 2 tests is basic anatomy of a test (setup → test → verify). It uses JUnit as well.

Technique: The technique I used is equivalence partitioning. Partition 1 is null, partition 2 is an empty string, and partition 3 is any non-empty string.

Why: These are good tests because it is complete but minimal with easy readability. It is also fast and tests good functionality.

Here are the pictures of the tests and evidence of running it through Maven

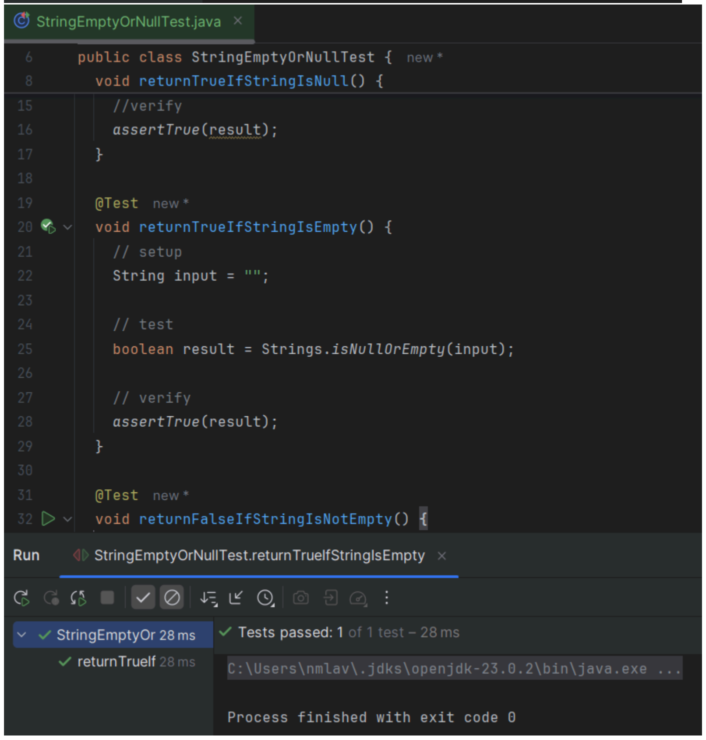


StringEmptyOrNullTest.java

```java
package org.assertj.core.util;

import static org.junit.jupiter.api.Assertions.*;
import org.junit.jupiter.api.Test;

public class StringEmptyOrNullTest {  new *
  @Test  new *
  void returnTrueIfStringIsNull() {
    // setup
    String input = null;

    // test
    boolean result = Strings.isNullOrEmpty(input);

    //verify
    assertTrue(result);
  }

  @Test  new *
  void returnTrueIfStringIsEmpty() {
```

Run  StringEmptyOrNullTest.returnTrueIfStringIsNull

```
StringEmptyOr 27 ms    Tests passed: 1 of 1 test – 27 ms
  returnTrueIf 27 ms   C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

                       Process finished with exit code 0
```

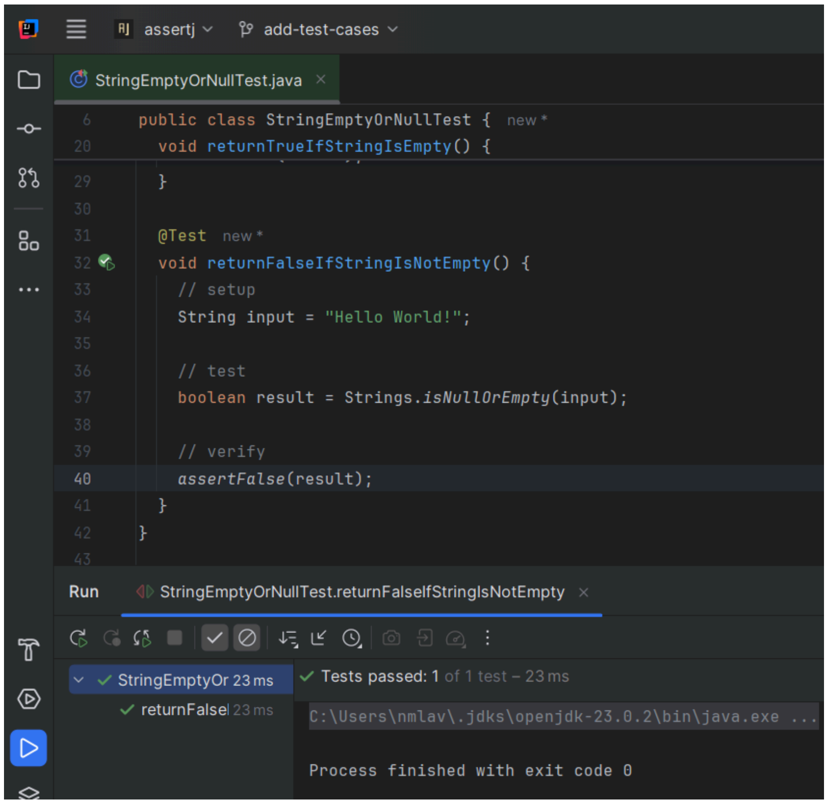


StringEmptyOrNullTest.java

```java
public class StringEmptyOrNullTest {  new *
  void returnTrueIfStringIsNull() {
    //verify
    assertTrue(result);
  }

  @Test  new *
  void returnTrueIfStringIsEmpty() {
    // setup
    String input = "";

    // test
    boolean result = Strings.isNullOrEmpty(input);

    // verify
    assertTrue(result);
  }

  @Test  new *
  void returnFalseIfStringIsNotEmpty() {
```

Run  StringEmptyOrNullTest.returnTrueIfStringIsEmpty

```
StringEmptyOr 28 ms    Tests passed: 1 of 1 test – 28 ms
  returnTrueIf 28 ms   C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

                       Process finished with exit code 0
```



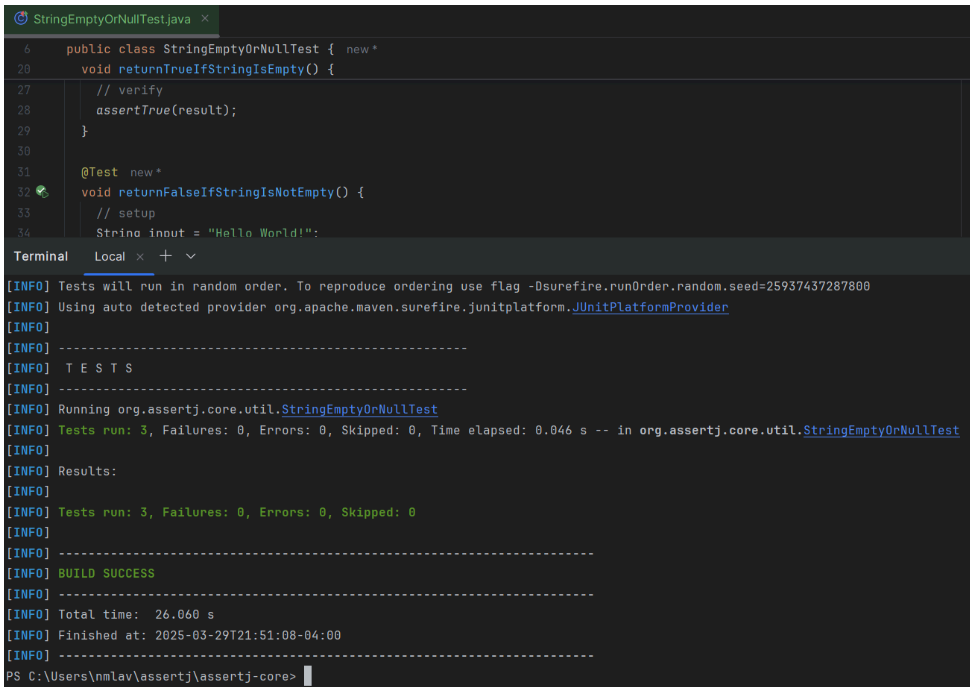
assertj · add-test-cases

StringEmptyOrNullTest.java

```java
public class StringEmptyOrNullTest {  new *
  void returnTrueIfStringIsEmpty() {
  }

  @Test  new *
  void returnFalseIfStringIsNotEmpty() {
    // setup
    String input = "Hello World!";

    // test
    boolean result = Strings.isNullOrEmpty(input);

    // verify
    assertFalse(result);
  }
}
```

Run  StringEmptyOrNullTest.returnFalseIfStringIsNotEmpty

```
StringEmptyOr 23 ms
  returnFalsel 23 ms
Tests passed: 1 of 1 test – 23 ms

C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



StringEmptyOrNullTest.java

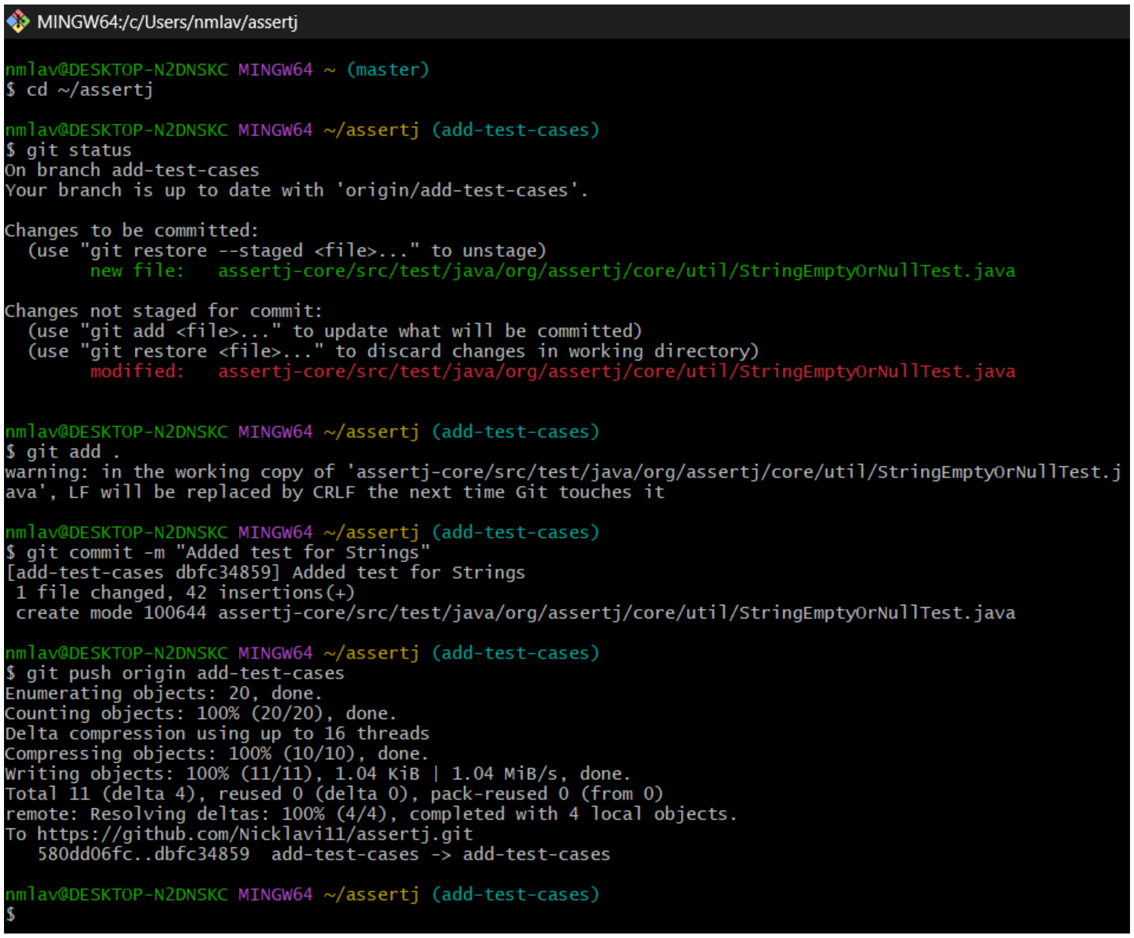
```java
public class StringEmptyOrNullTest {  new *
  void returnTrueIfStringIsEmpty() {
    // verify
    assertTrue(result);
  }

  @Test  new *
  void returnFalseIfStringIsNotEmpty() {
    // setup
    String input = "Hello World!";
```

Terminal  Local

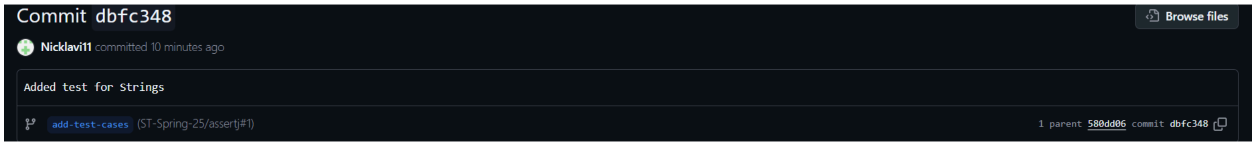
```
[INFO] Tests will run in random order. To reproduce ordering use flag -Dsurefire.runOrder.random.seed=25937437287800
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.assertj.core.util.StringEmptyOrNullTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.046 s -- in org.assertj.core.util.StringEmptyOrNullTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  26.060 s
[INFO] Finished at: 2025-03-29T21:51:08-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\assertj\assertj-core>
```

*Commit:*

After the tests passed, I committed it to the repository



```
MINGW64:/c/Users/nmlav/assertj

nmlav@DESKTOP-N2DNSKC MINGW64 ~ (master)
$ cd ~/assertj

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git status
On branch add-test-cases
Your branch is up to date with 'origin/add-test-cases'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.java


nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.j
ava', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "Added test for Strings"
[add-test-cases dbfc34859] Added test for Strings
 1 file changed, 42 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/util/StringEmptyOrNullTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.04 KiB | 1.04 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   580dd06fc..dbfc34859  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$
```



```
Commit dbfc348                                                          Browse files
Nicklavi11 committed 10 minutes ago

Added test for Strings

add-test-cases (ST-Spring-25/assertj#1)                1 parent 580dd06 commit dbfc348
```

Here is the link to the commit:

[https://github.com/Nicklavi11/assertj/commit/dbfc34859a6785b740819a42422c5613c148e2ed](https://github.com/Nicklavi11/assertj/commit/dbfc34859a6785b740819a42422c5613c148e2ed)

## Test Case #3

*What is the class and method I am testing:*

File: NumberAssert.java

- [https://github.com/ST-Spring-25/assertj/blob/main/assertj-core/src/main/java/org/assertj/core/api/NumberAssert.java](https://github.com/ST-Spring-25/assertj/blob/main/assertj-core/src/main/java/org/assertj/core/api/NumberAssert.java)
- (~assertj/assertj-core/src/main/java/org/assertj/core/api/NumberAssert.java)

Class: NumberAssert is a class that is for all numeric types, such as Integer, Long, Float, etc, and it adds readable methods such as isPositive(), isZero(), and isBetween(), which verify numeric ranges while generating clear failure messages.

Method: The method I am testing is isBetween(start, end), and it passes when start <= actual <= end and fails otherwise. It is inclusive on both ends and throws an AssertionError with a descriptive message when the actual value is out of range.

*What is the test:*

File: NumberRangeTest.java

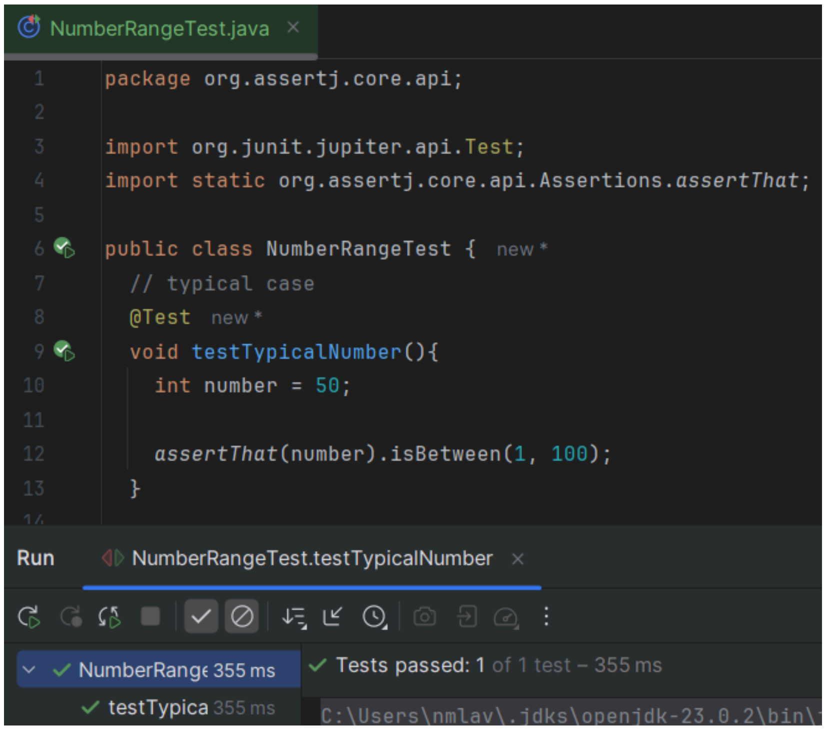
- [https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java](https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java)
- (~/assertj/assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java)

Structure: The structure of these tests is basic anatomy of a test. It also uses JUnit.

Technique: The testing technique is boundary value testing (min, min + 1, typical, max - 1, max). I am testing between numbers 1 and 100. Input for min is 1, min + 1 is 2, typical is 50, max - 1 is 99, and max is 100.

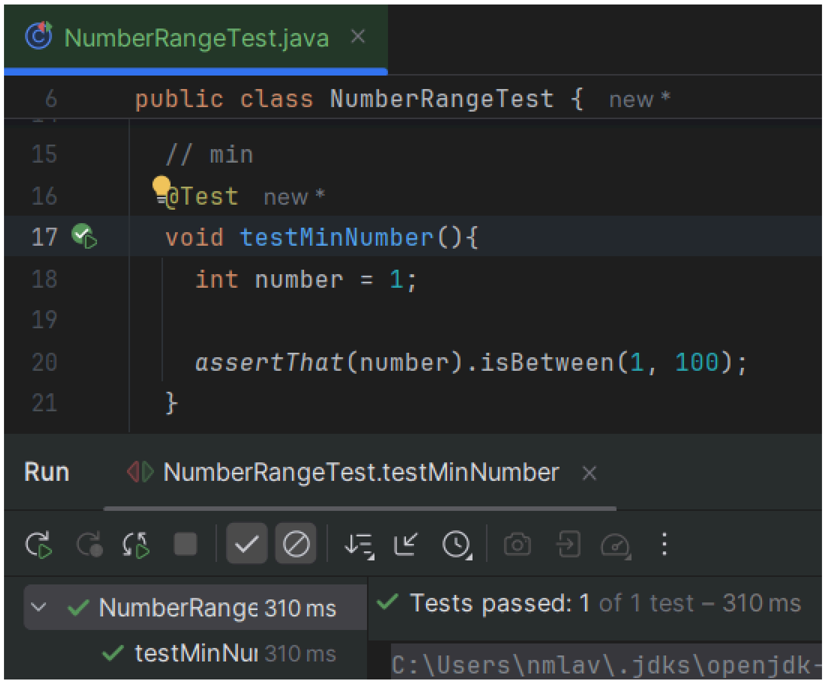
Why: This is a great candidate for Boundary tests because it can test critical points and catch errors one by one. It uses assertj's assertion so intent and failure messages are clear, and each case is independent, so there is no randomness. It gives full confidence in isBetween for integer input without over-testing inputs.

Here are the pictures of the tests and evidence of running it through Maven



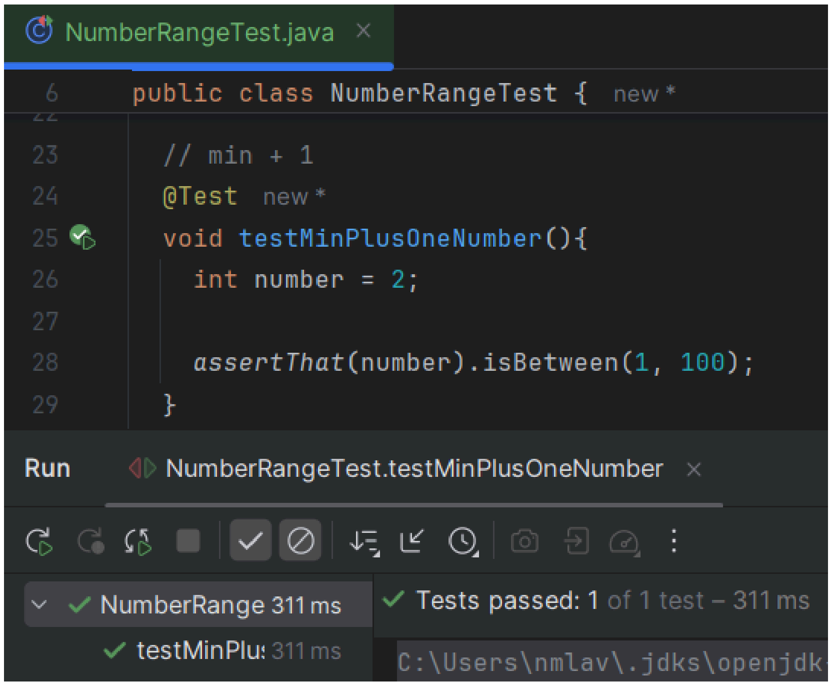
```
package org.assertj.core.api;

import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class NumberRangeTest {  new *
  // typical case
  @Test  new *
  void testTypicalNumber(){
    int number = 50;

    assertThat(number).isBetween(1, 100);
  }
```

```
NumberRang€ 355 ms     Tests passed: 1 of 1 test – 355 ms
  testTypica 355 ms    C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\
```



```
public class NumberRangeTest {  new *
  // min
  @Test  new *
  void testMinNumber(){
    int number = 1;

    assertThat(number).isBetween(1, 100);
  }
```

```
NumberRang€ 310 ms     Tests passed: 1 of 1 test – 310 ms
  testMinNu 310 ms     C:\Users\nmlav\.jdks\openjdk-
```

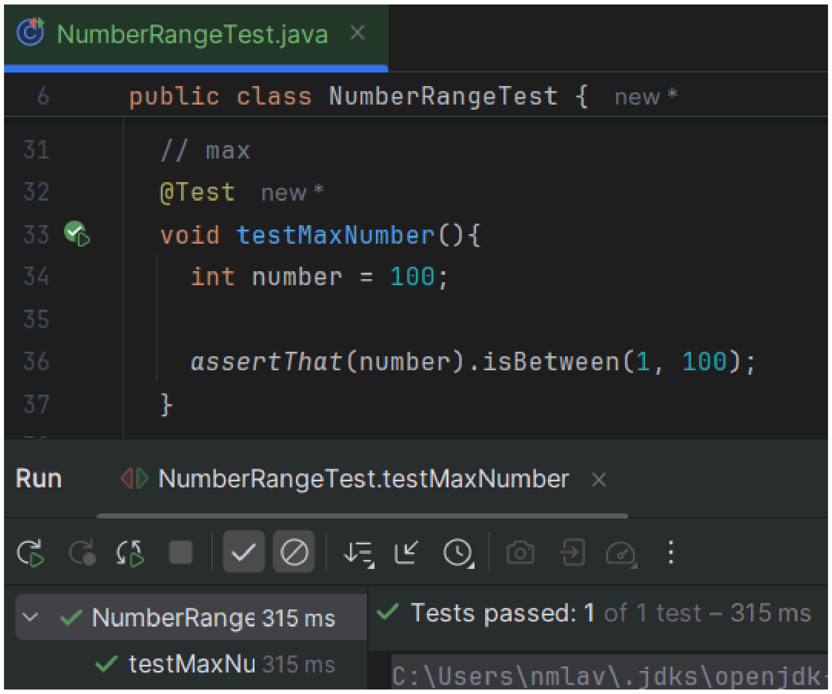


NumberRangeTest.java

```
public class NumberRangeTest {  new *

  // min + 1
  @Test  new *
  void testMinPlusOneNumber(){
    int number = 2;

    assertThat(number).isBetween(1, 100);
  }
```

Run NumberRangeTest.testMinPlusOneNumber

```
NumberRange 311 ms        Tests passed: 1 of 1 test – 311 ms
  testMinPlu: 311 ms      C:\Users\nmlav\.jdks\openjdk
```

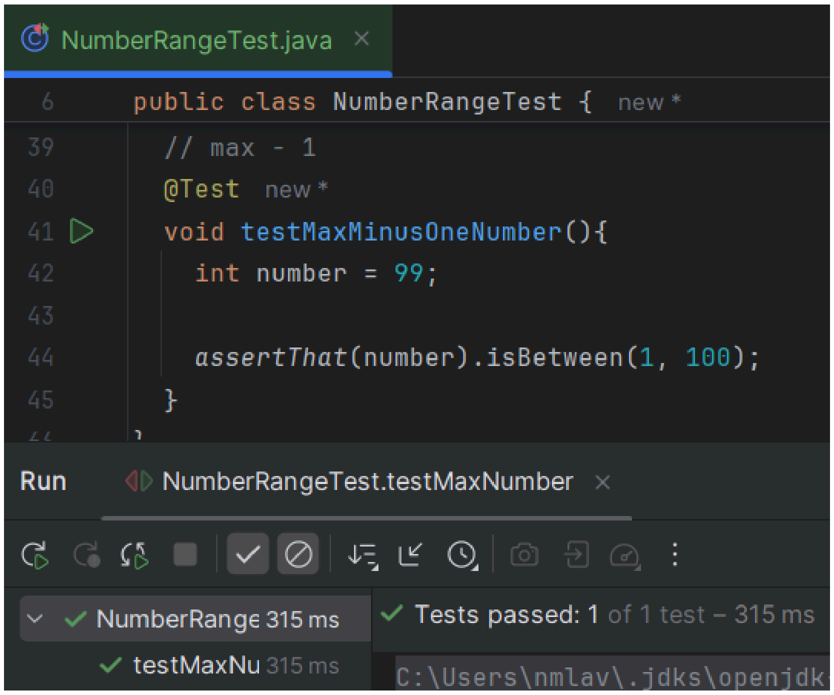


NumberRangeTest.java

```
public class NumberRangeTest {  new *

  // max
  @Test  new *
  void testMaxNumber(){
    int number = 100;

    assertThat(number).isBetween(1, 100);
  }
```

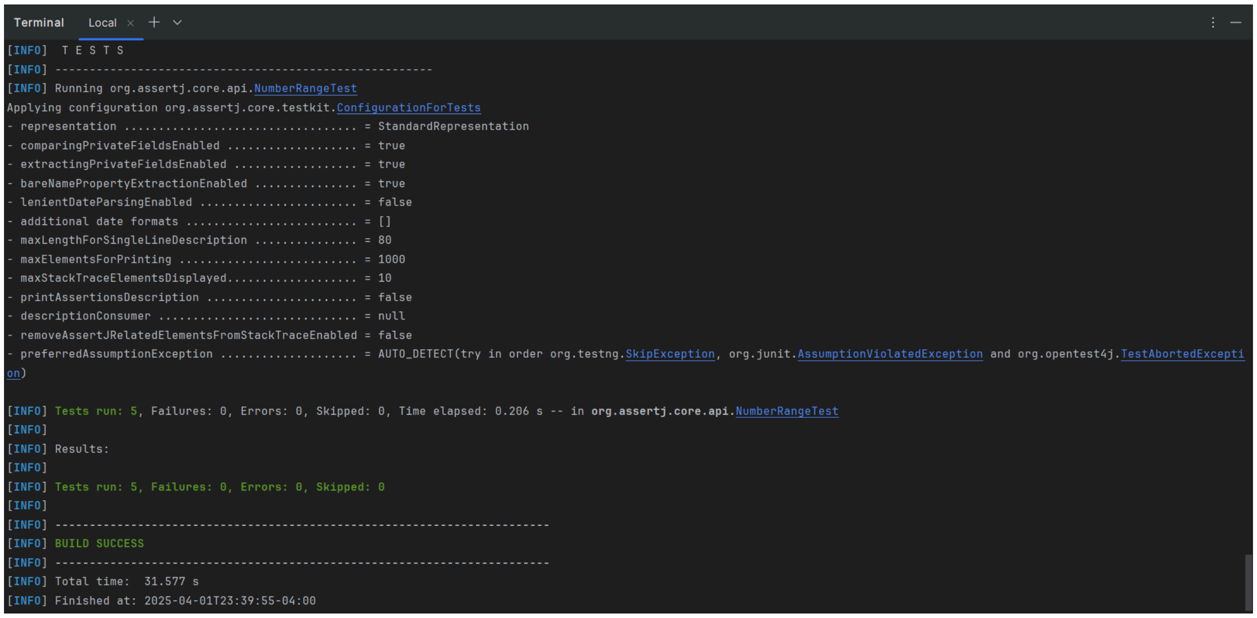
Run NumberRangeTest.testMaxNumber

```
NumberRange 315 ms        Tests passed: 1 of 1 test – 315 ms
  testMaxNu 315 ms        C:\Users\nmlav\.jdks\openjdk
```



```
public class NumberRangeTest {  new *
    // max - 1
    @Test  new *
    void testMaxMinusOneNumber(){
        int number = 99;

        assertThat(number).isBetween(1, 100);
    }
```

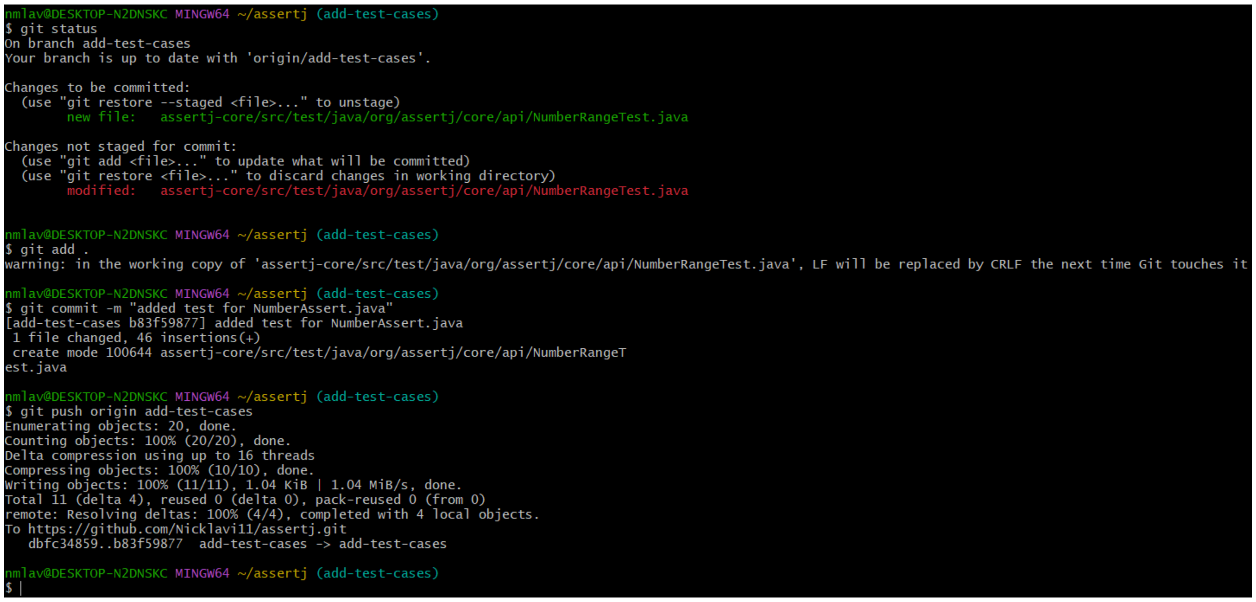
```
Tests passed: 1 of 1 test – 315 ms
C:\Users\nmlav\.jdks\openjdk
```



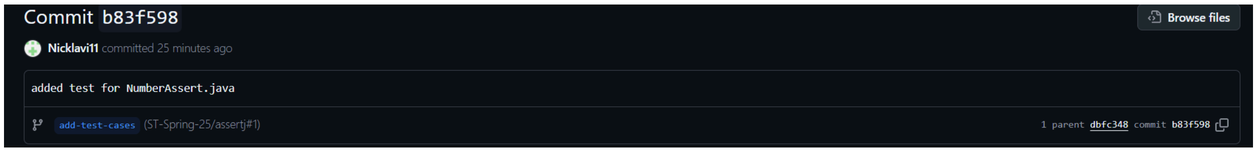
```
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.assertj.core.api.NumberRangeTest
Applying configuration org.assertj.core.testkit.ConfigurationForTests
- representation ................................. = StandardRepresentation
- comparingPrivateFieldsEnabled .................. = true
- extractingPrivateFieldsEnabled ................. = true
- bareNamePropertyExtractionEnabled .............. = true
- lenientDateParsingEnabled ...................... = false
- additional date formats ........................ = []
- maxLengthForSingleLineDescription .............. = 80
- maxElementsForPrinting ......................... = 1000
- maxStackTraceElementsDisplayed.................. = 10
- printAssertionsDescription ..................... = false
- descriptionConsumer ............................ = null
- removeAssertJRelatedElementsFromStackTraceEnabled = false
- preferredAssumptionException ................... = AUTO_DETECT(try in order org.testng.SkipException, org.junit.AssumptionViolatedException and org.opentest4j.TestAbortedExcepti
on)

[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.206 s -- in org.assertj.core.api.NumberRangeTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  31.577 s
[INFO] Finished at: 2025-04-01T23:39:55-04:00
```

*Commit:*

After the tests passed, I committed it to the repository



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git status
On branch add-test-cases
Your branch is up to date with 'origin/add-test-cases'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java


nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/api/NumberRangeTest.java', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "added test for NumberAssert.java"
[add-test-cases b83f59877] added test for NumberAssert.java
 1 file changed, 46 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/api/NumberRangeT
est.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.04 KiB | 1.04 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   dbfc34859..b83f59877  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ |
```



Commit b83f598

Nicklavi11 committed 25 minutes ago

```
added test for NumberAssert.java
```

add-test-cases (ST-Spring-25/assertj#1)

1 parent dbfc348 commit b83f598

Here is the link to the commit:

https://github.com/ST-Spring-25/assertj/commit/b83f5987744687cca825e3ebee3eacf8d9477c40

## Test Case #4

*What is the class and method I am testing:*

File: AbstractCharSequenceAssert.java

- https://github.com/Nicklavi11/assertj/blob/main/assertj-core/src/main/java/org/assertj/core/api/AbstractCharSequenceAssert.java
- (~/assertj/assertj-core/src/main/java/org/assertj/core/api/AbstractCharSequenceAssert.java)

Class: The AbstractCharSequenceAssert is a base class that powers assertj assertion for charSequence types. It has high-level text that checks and returns SELF for good chaining.

Method: The method I am testing is isNotBlank(), which is only successful when the actual value is not null, not empty, and has at least one non-whitespace character. If it is not successful, it throws an AssertionError with a meaningful message.

*What is the test:*

File: StringValidationTest.java

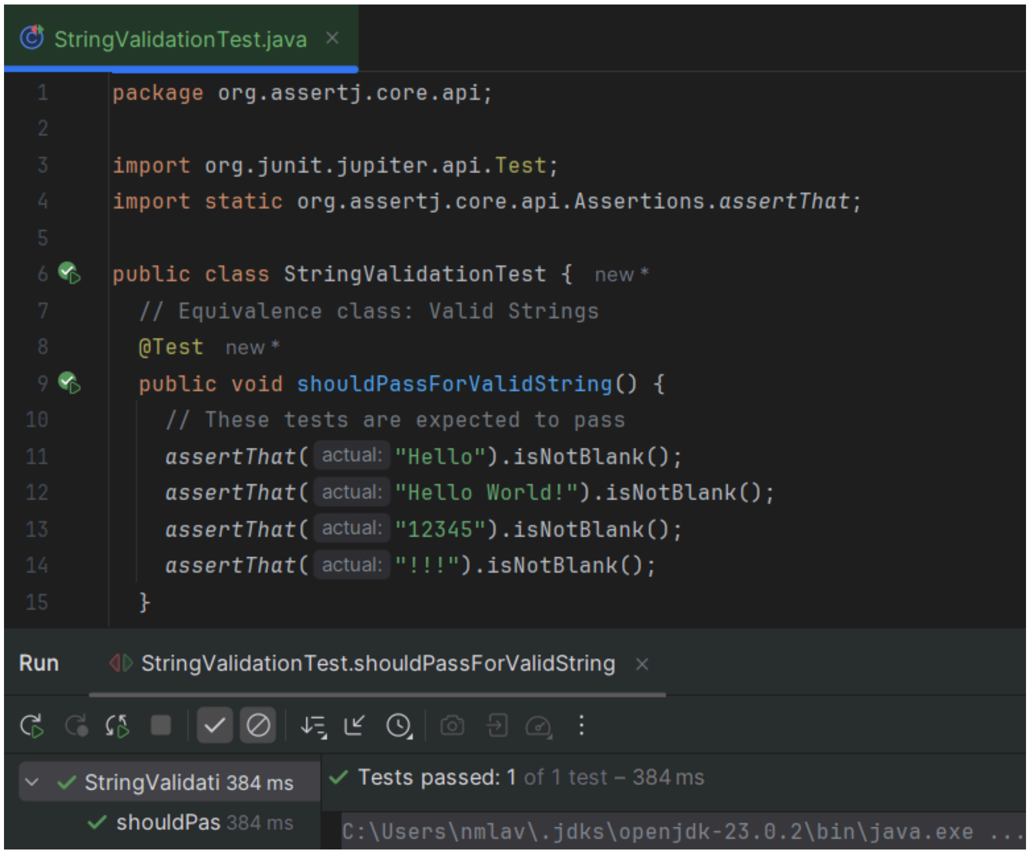
- https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java
- (~/assertj/assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java)

Structure: The test structure is the anatomy of a test. It also uses JUnit and the @Disabled annotation to make sure the Maven build is successful. The tests are designed to fail and use assertThatThrownBy for checking it correctly, which is successful for this test.

Technique: The technique for these tests is Equivalence Partitioning.

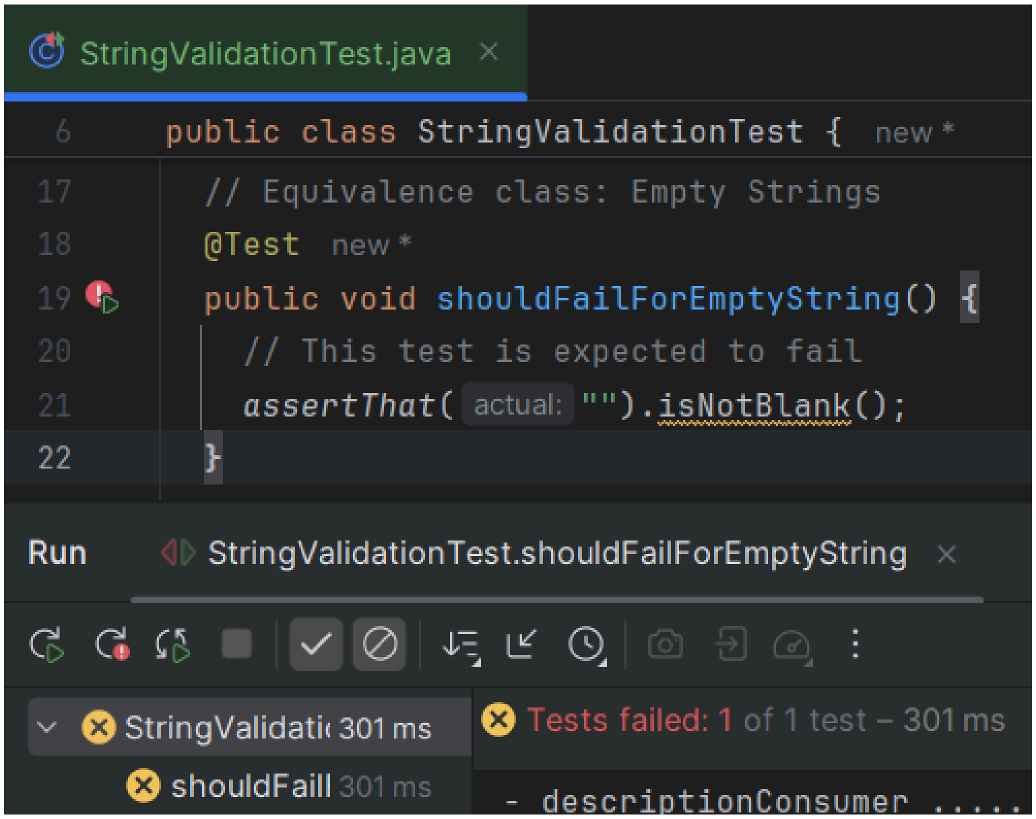
Why: These tests are good because they trigger every internal branch, such as valid, empty, whitespace, and null. It is also very readable, and it has failure-expected tests that are present but are disabled, so it does not break the build.

Here are the pictures of the tests and evidence of running through Maven. The tests are captured before the @Disabled annotation, and it fails through Maven, which is evidence of success in this test:


StringValidationTest.java
Run
StringValidationTest.shouldPassForValidString

```
package org.assertj.core.api;

import org.junit.jupiter.api.Test;
import static org.assertj.core.api.Assertions.assertThat;

public class StringValidationTest {  new *
  // Equivalence class: Valid Strings
  @Test  new *
  public void shouldPassForValidString() {
    // These tests are expected to pass
    assertThat( actual: "Hello").isNotBlank();
    assertThat( actual: "Hello World!").isNotBlank();
    assertThat( actual: "12345").isNotBlank();
    assertThat( actual: "!!!").isNotBlank();
  }
```

```
✓ StringValidati 384 ms     ✓ Tests passed: 1 of 1 test – 384 ms
  ✓ shouldPas 384 ms        C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...
```


StringValidationTest.java

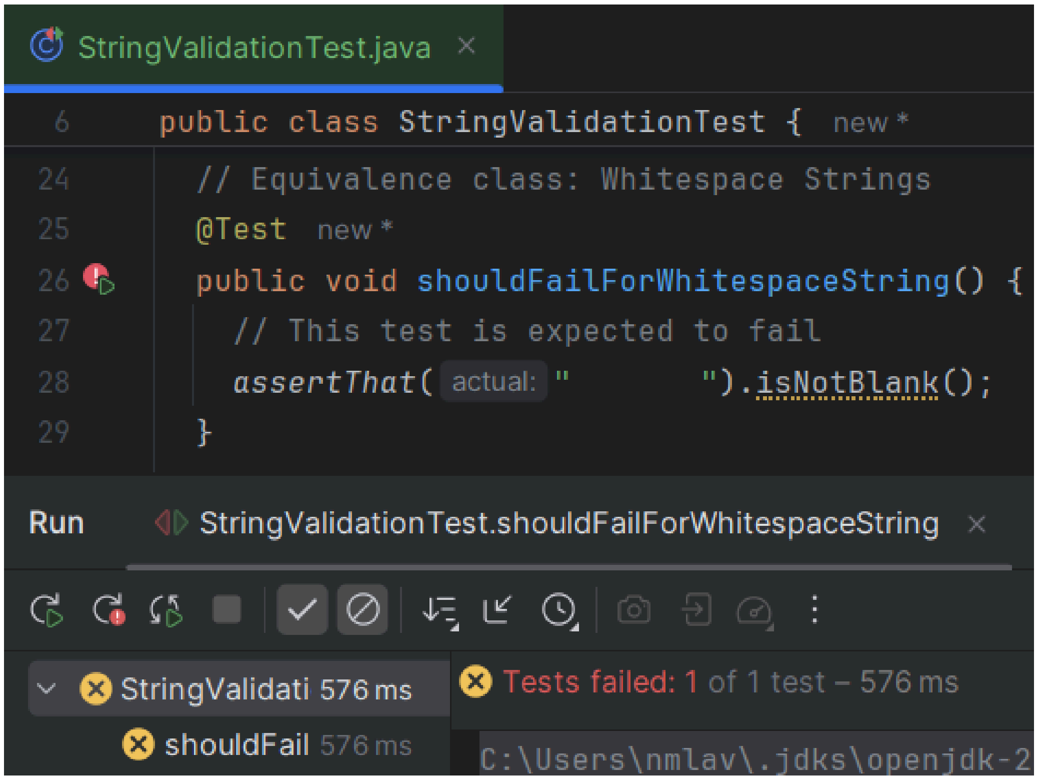
```
public class StringValidationTest {  new *
    // Equivalence class: Empty Strings
    @Test  new *
    public void shouldFailForEmptyString() {
        // This test is expected to fail
        assertThat( actual: "").isNotBlank();
    }
```

Run  StringValidationTest.shouldFailForEmptyString

StringValidati 301 ms
shouldFaill 301 ms

Tests failed: 1 of 1 test – 301 ms

```
- descriptionConsumer .....
```


StringValidationTest.java

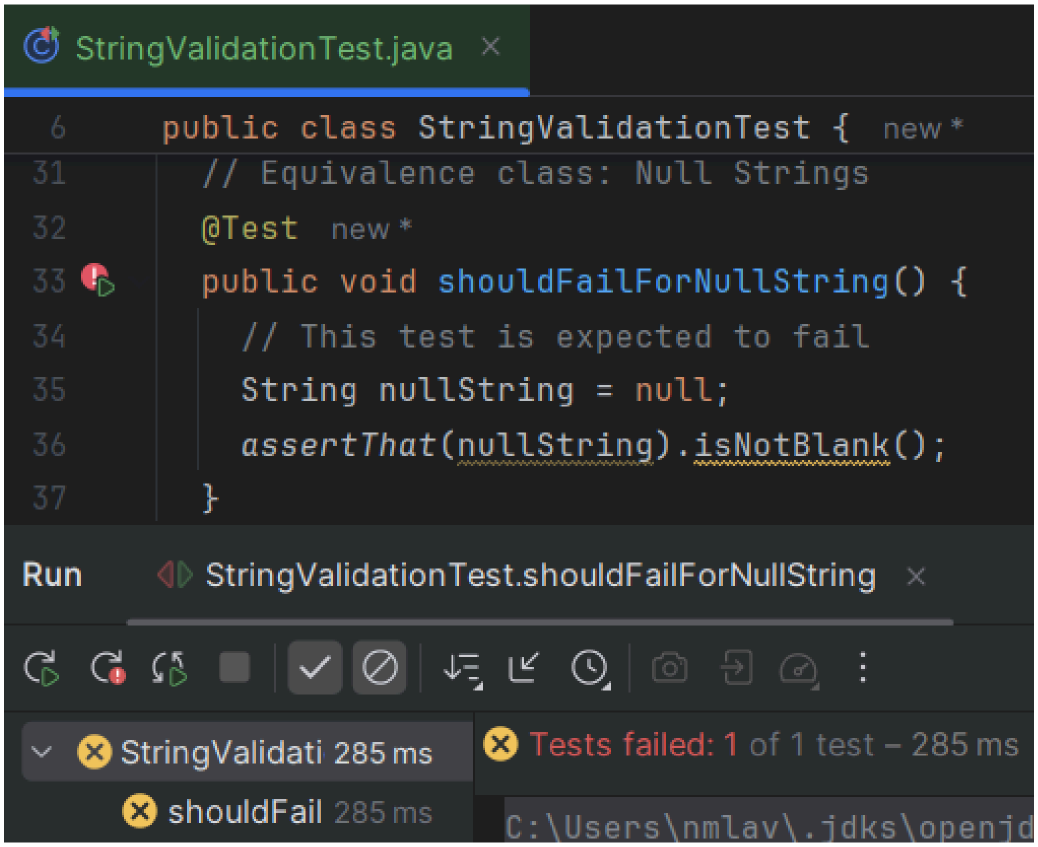
```
public class StringValidationTest {  new *
    // Equivalence class: Whitespace Strings
    @Test  new *
    public void shouldFailForWhitespaceString() {
        // This test is expected to fail
        assertThat( actual: "       ").isNotBlank();
    }
```

Run  StringValidationTest.shouldFailForWhitespaceString

StringValidati 576 ms
shouldFail 576 ms

Tests failed: 1 of 1 test – 576 ms

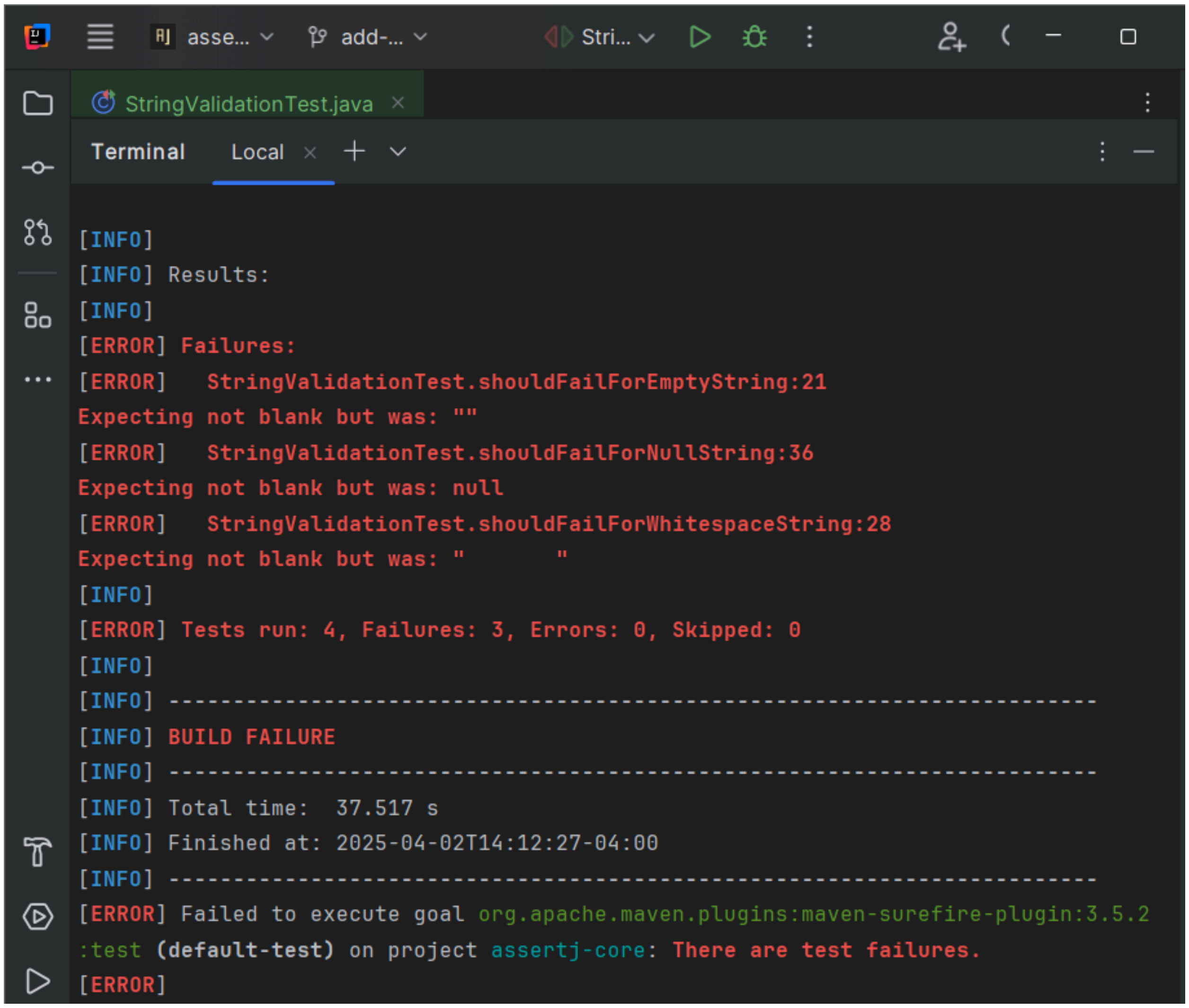
```
C:\Users\nmlav\.jdks\openjdk-2
```


StringValidationTest.java

```
public class StringValidationTest {  new *
  // Equivalence class: Null Strings
  @Test  new *
  public void shouldFailForNullString() {
    // This test is expected to fail
    String nullString = null;
    assertThat(nullString).isNotBlank();
  }
```

Run
StringValidationTest.shouldFailForNullString
StringValidati 285 ms
shouldFail 285 ms

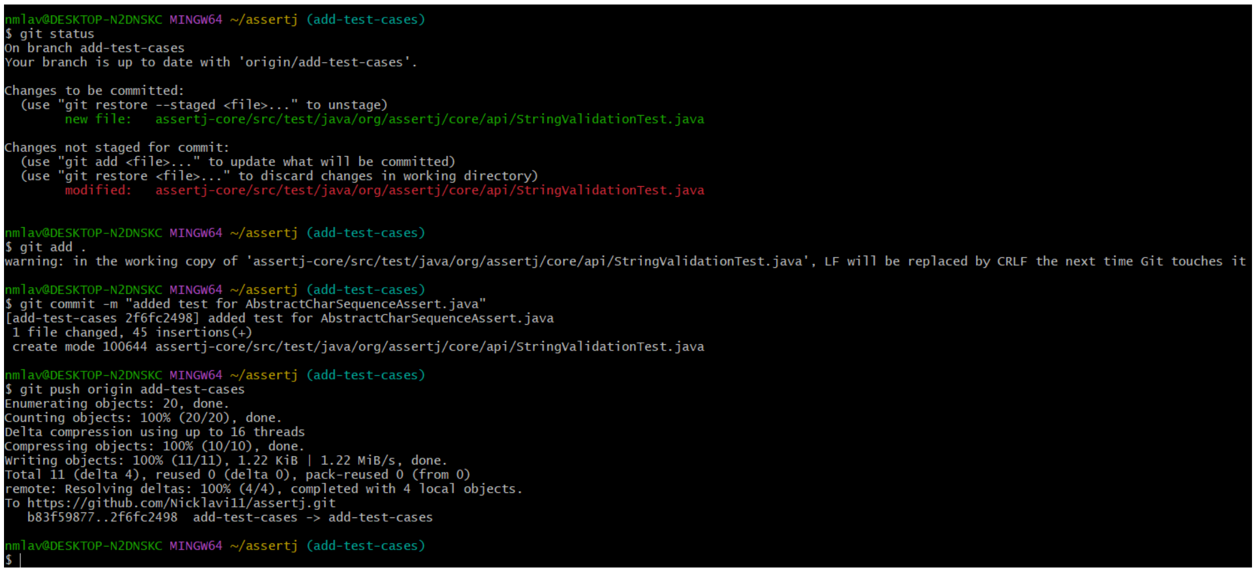
```
Tests failed: 1 of 1 test – 285 ms
C:\Users\nmlav\.jdks\openjd
```



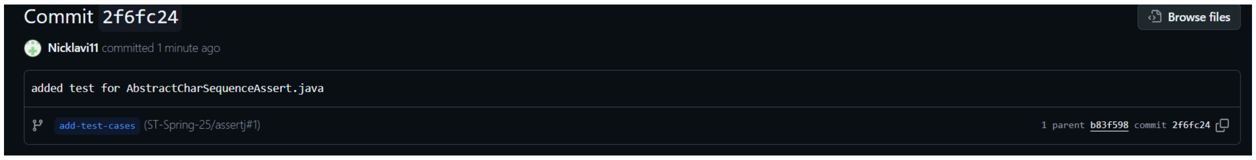
```
[INFO]
[INFO] Results:
[INFO]
[ERROR] Failures:
[ERROR]   StringValidationTest.shouldFailForEmptyString:21
Expecting not blank but was: ""
[ERROR]   StringValidationTest.shouldFailForNullString:36
Expecting not blank but was: null
[ERROR]   StringValidationTest.shouldFailForWhitespaceString:28
Expecting not blank but was: "        "
[INFO]
[ERROR] Tests run: 4, Failures: 3, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD FAILURE
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  37.517 s
[INFO] Finished at: 2025-04-02T14:12:27-04:00
[INFO] ------------------------------------------------------------------------
[ERROR] Failed to execute goal org.apache.maven.plugins:maven-surefire-plugin:3.5.2
:test (default-test) on project assertj-core: There are test failures.
[ERROR]
```

*Commit:*

After the tests passed (with the @Disabled annotation), I committed it to the repository



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git status
On branch add-test-cases
Your branch is up to date with 'origin/add-test-cases'.

Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java


nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "added test for AbstractCharSequenceAssert.java"
[add-test-cases 2f6fc2498] added test for AbstractCharSequenceAssert.java
 1 file changed, 45 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/api/StringValidationTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.22 KiB | 1.22 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   b83f59877..2f6fc2498  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ |
```



Commit 2f6fc24

Nicklavi11 committed 1 minute ago

```
added test for AbstractCharSequenceAssert.java
```

add-test-cases (ST-Spring-25/assertj#1)    1 parent b83f598 commit 2f6fc24

Here is the link to the commit:

https://github.com/ST-Spring-25/assertj/commit/2f6fc249846d916be63b50dee93f92483f
f977ba

**Test Case #5**

*What is the class and method I am testing:*

File: Lists.java

- https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/util/Lists.java
- (~/assertj/assertj-core/src/main/java/org/assertj/core/util/Lists.java)

Class: The Lists class is a small helper class in assertj that offers methods for ArrayList creation.

Method: The method I am testing is newArrayList(T… elements), which creates new ArrayLists. It returns an empty ArrayList when no arguments are there or a list with pre-populated elements that are provided in the correct order.

*What is the test:*

File: ListsTest.java

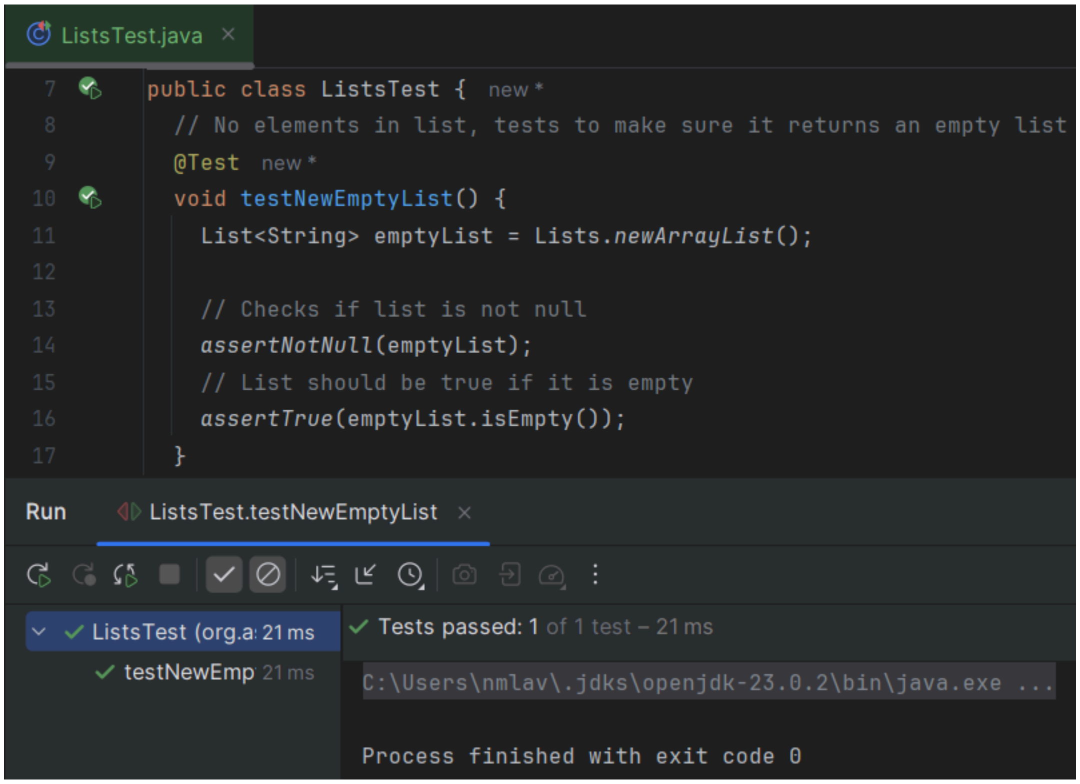
- https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/util/ListsTest.java
- (~/assertj/assertj-core/src/test/java/org/assertj/core/util/ListsTest.java)

Structure: The structure is the basic anatomy of a test. It also uses JUnit.

Technique: The technique is Equivalence Partitioning. Partition 1 is 0 arguments, partition 2 is exactly 1 argument, and partition 3 is 2 or more arguments.

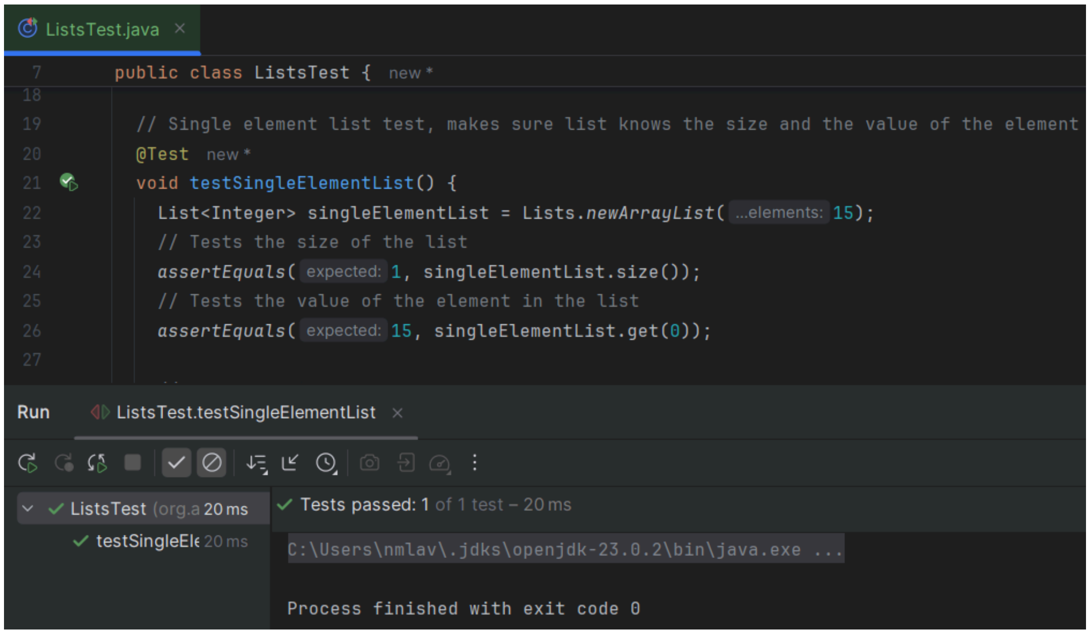
Why: These tests work because they cover all logical paths with only 3 tests and ensure both size and content correctness. It uses the helper to check these tests, and if it returns null, it immediately fails these tests and protects calls in assertJ.

Here are the pictures of the tests and evidence of running through Maven:



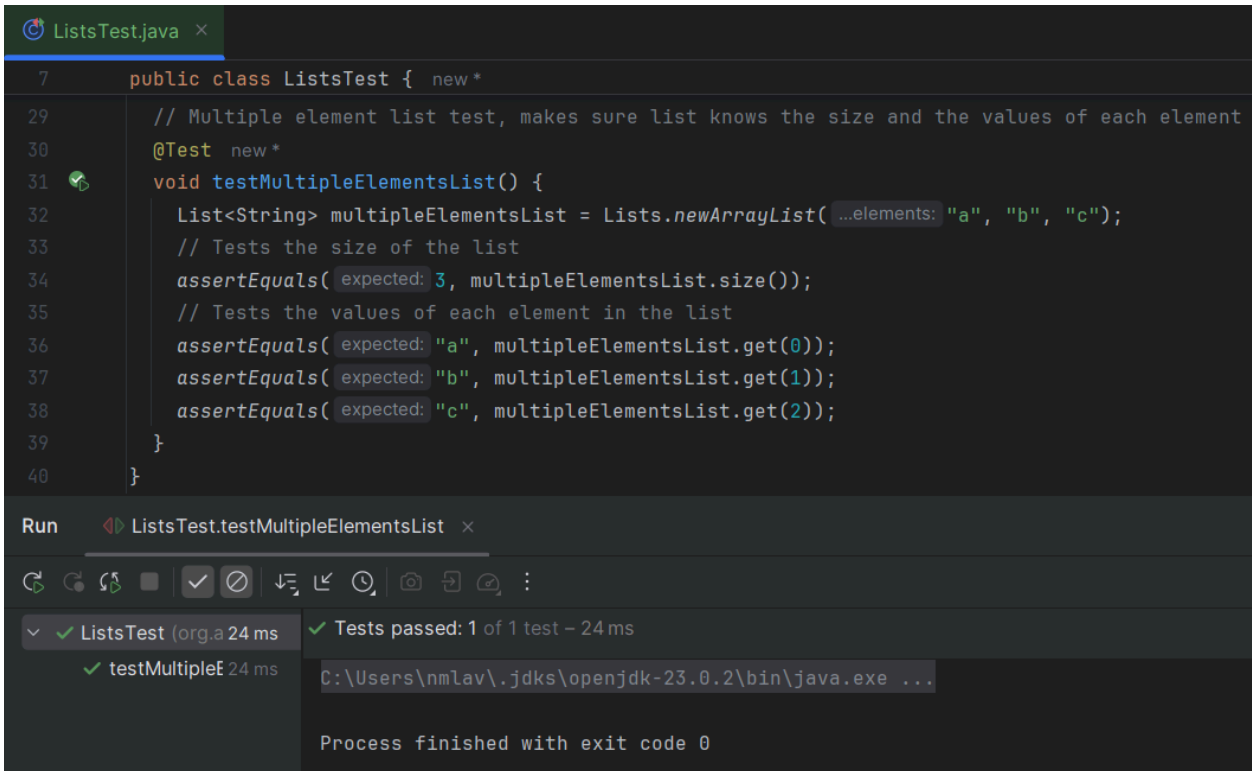
```java
public class ListsTest {  new *
  // No elements in list, tests to make sure it returns an empty list
  @Test  new *
  void testNewEmptyList() {
    List<String> emptyList = Lists.newArrayList();

    // Checks if list is not null
    assertNotNull(emptyList);
    // List should be true if it is empty
    assertTrue(emptyList.isEmpty());
  }
```

```
✓ Tests passed: 1 of 1 test – 21 ms

C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



```java
public class ListsTest {  new *

  // Single element list test, makes sure list knows the size and the value of the element
  @Test  new *
  void testSingleElementList() {
    List<Integer> singleElementList = Lists.newArrayList( ...elements: 15);
    // Tests the size of the list
    assertEquals( expected: 1, singleElementList.size());
    // Tests the value of the element in the list
    assertEquals( expected: 15, singleElementList.get(0));

```

```
✓ Tests passed: 1 of 1 test – 20 ms

C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



ListsTest.java

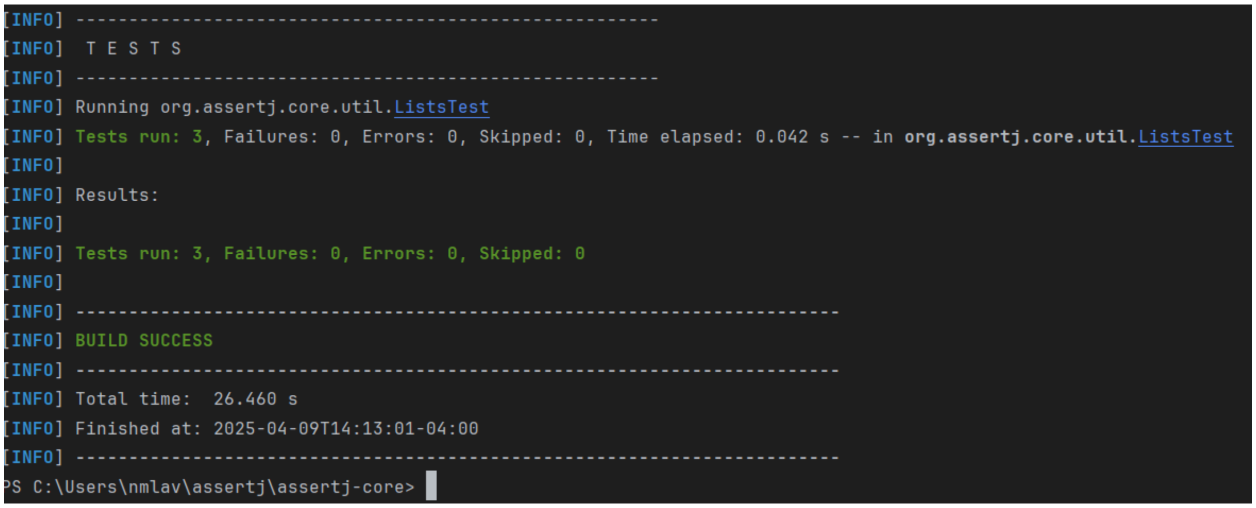
```
    public class ListsTest {  new *
      // Multiple element list test, makes sure list knows the size and the values of each element
      @Test  new *
      void testMultipleElementsList() {
        List<String> multipleElementsList = Lists.newArrayList( ...elements: "a", "b", "c");
        // Tests the size of the list
        assertEquals( expected: 3, multipleElementsList.size());
        // Tests the values of each element in the list
        assertEquals( expected: "a", multipleElementsList.get(0));
        assertEquals( expected: "b", multipleElementsList.get(1));
        assertEquals( expected: "c", multipleElementsList.get(2));
      }
    }
```

Run   ListsTest.testMultipleElementsList

ListsTest (org.a 24 ms
testMultipleE 24 ms

Tests passed: 1 of 1 test – 24 ms

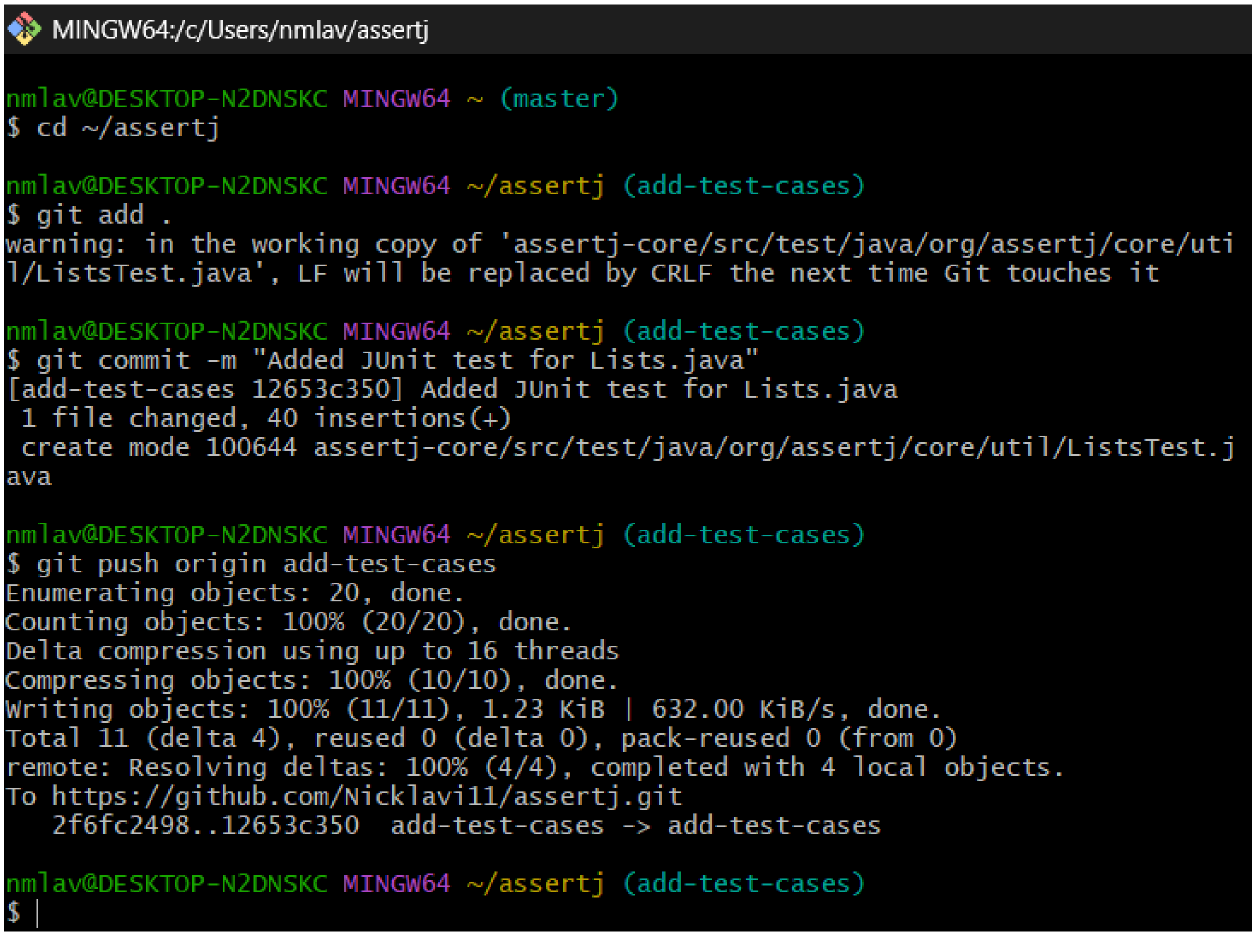
```
C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.assertj.core.util.ListsTest
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.042 s -- in org.assertj.core.util.ListsTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 3, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  26.460 s
[INFO] Finished at: 2025-04-09T14:13:01-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\assertj\assertj-core>
```

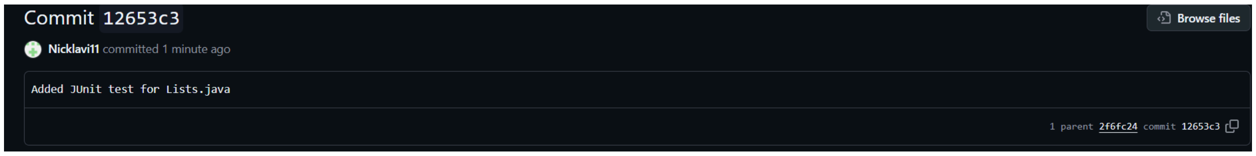
*Commit:*

After the tests passed, I committed it to the repository



MINGW64:/c/Users/nmlav/assertj

```
nmlav@DESKTOP-N2DNSKC MINGW64 ~ (master)
$ cd ~/assertj

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/uti
l/ListsTest.java', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "Added JUnit test for Lists.java"
[add-test-cases 12653c350] Added JUnit test for Lists.java
 1 file changed, 40 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/util/ListsTest.j
ava

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.23 KiB | 632.00 KiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   2f6fc2498..12653c350  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ |
```



Commit 12653c3

Nicklavi11 committed 1 minute ago

```
Added JUnit test for Lists.java
```

1 parent 2f6fc24 commit 12653c3

Here is the link to the commit:

[https://github.com/ST-Spring-25/assertj/commit/12653c350b12e00fe1e46b44e78ed95e95d51d91](https://github.com/ST-Spring-25/assertj/commit/12653c350b12e00fe1e46b44e78ed95e95d51d91)

**Test Case #6**

*What is the class and method I am testing:*

File: Condition.java

- [https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/api/Condition.java](https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/main/java/org/assertj/core/api/Condition.java)
- (~/assertj/assertj-core/src/main/java/org/assertj/core/api/Condition.java)

Class: The Condition class is used to define custom conditions for assertions.

Method: The Method I am testing is a subclass, and its functionality is to return true if the value satisfies the condition and false when it doesn't.

*What is the test:*

File: ConditionTest.java

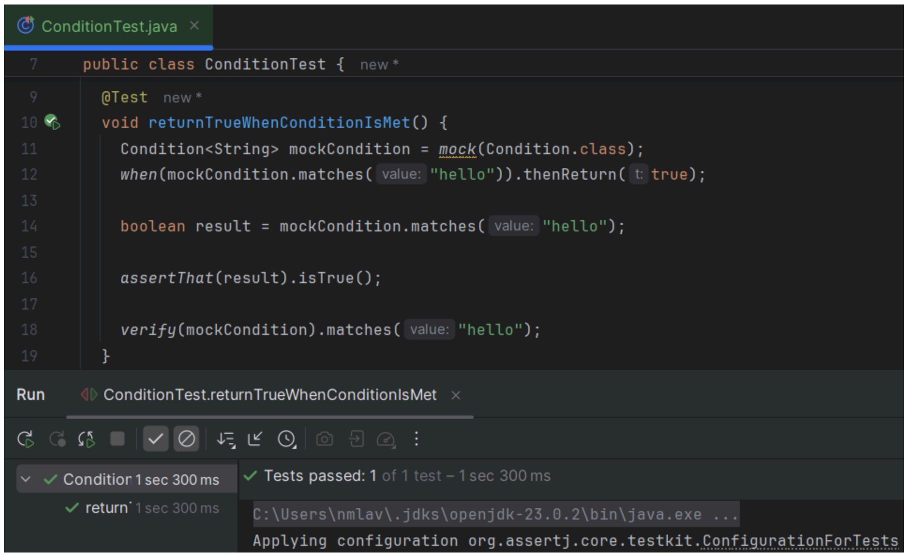
- [https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/ConditionTest.java](https://github.com/Nicklavi11/assertj/blob/add-test-cases/assertj-core/src/test/java/org/assertj/core/api/ConditionTest.java)
- (~/assertj/assertj-core/src/test/java/org/assertj/core/api/ConditionTest.java)

Structure: The structure of the test is mock-based, and it uses Mockito.

Technique: The technique is Mock testing. It replaces the Condition with a mock, and explicitly controls its matched return value, and verifies the method is used with the expected argument.

Why: This test works because it uses a Condition to make sure the external code can rely on the interaction. It shows both positive and negative branches with simplicity. It is fast, deterministic, and does not need external resources, which is good for tests.

Here are the pictures of the tests and evidence of running through Maven:



ConditionTest.java

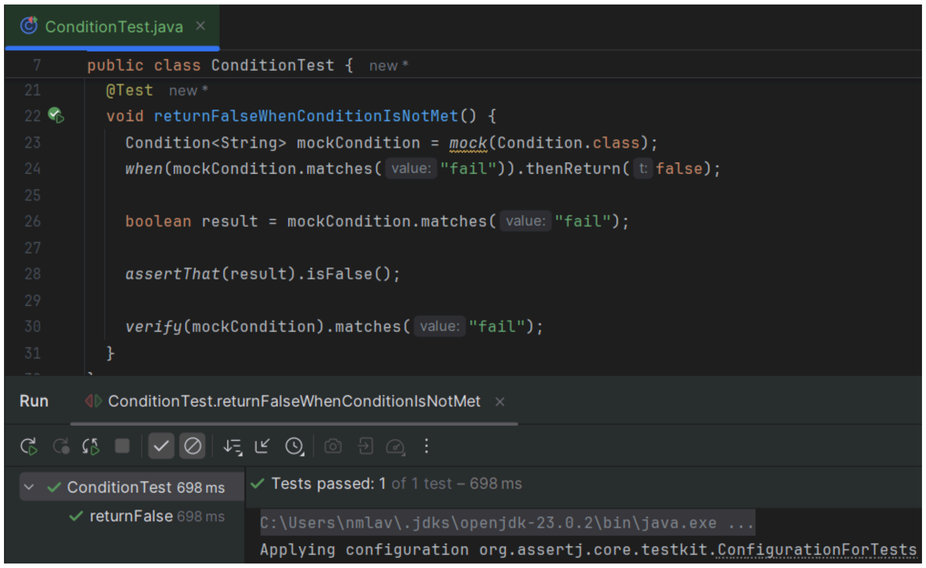
```java
public class ConditionTest {

  @Test
  void returnTrueWhenConditionIsMet() {
    Condition<String> mockCondition = mock(Condition.class);
    when(mockCondition.matches("hello")).thenReturn(true);

    boolean result = mockCondition.matches("hello");

    assertThat(result).isTrue();

    verify(mockCondition).matches("hello");
  }
```

Run  ConditionTest.returnTrueWhenConditionIsMet

Condition 1 sec 300 ms
return 1 sec 300 ms

Tests passed: 1 of 1 test – 1 sec 300 ms

```
C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...
Applying configuration org.assertj.core.testkit.ConfigurationForTests
```



ConditionTest.java

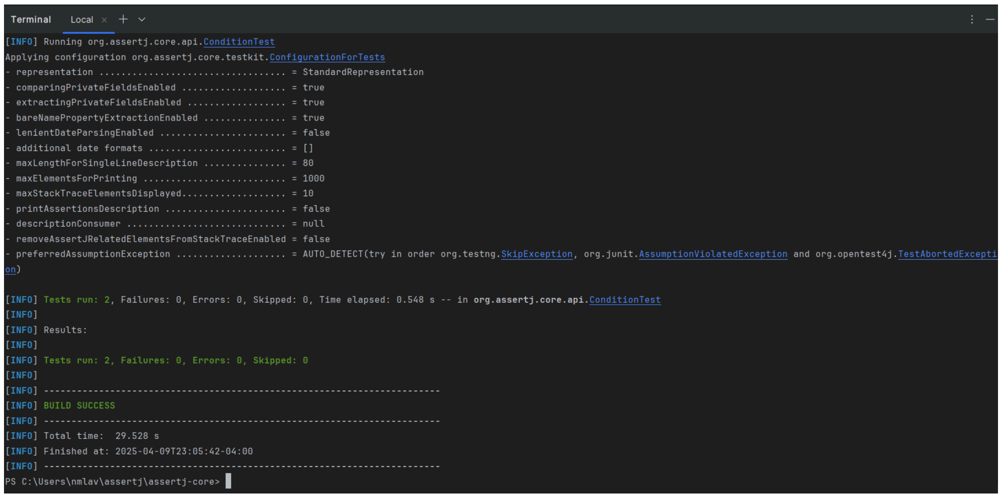
```java
public class ConditionTest {
  @Test
  void returnFalseWhenConditionIsNotMet() {
    Condition<String> mockCondition = mock(Condition.class);
    when(mockCondition.matches("fail")).thenReturn(false);

    boolean result = mockCondition.matches("fail");

    assertThat(result).isFalse();

    verify(mockCondition).matches("fail");
  }
```

Run  ConditionTest.returnFalseWhenConditionIsNotMet

ConditionTest 698 ms
returnFalse 698 ms

Tests passed: 1 of 1 test – 698 ms

```
C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...
Applying configuration org.assertj.core.testkit.ConfigurationForTests
```

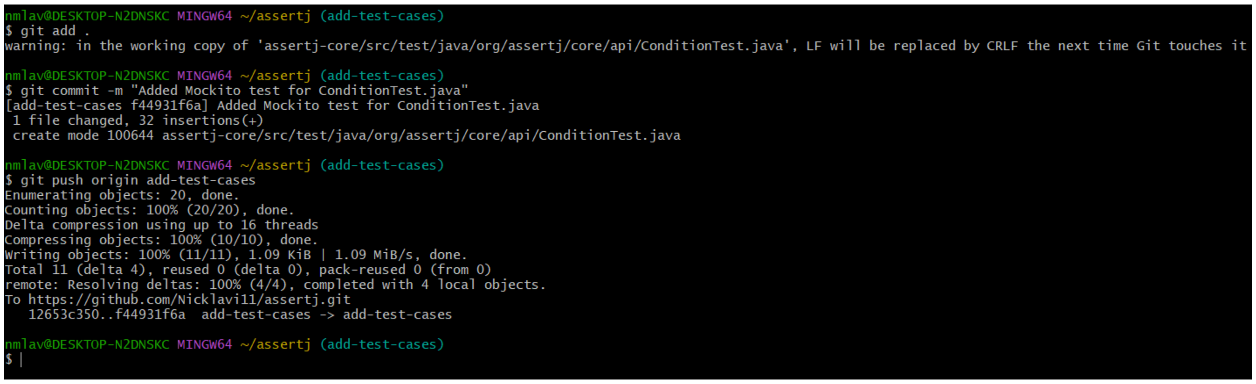


Terminal  Local

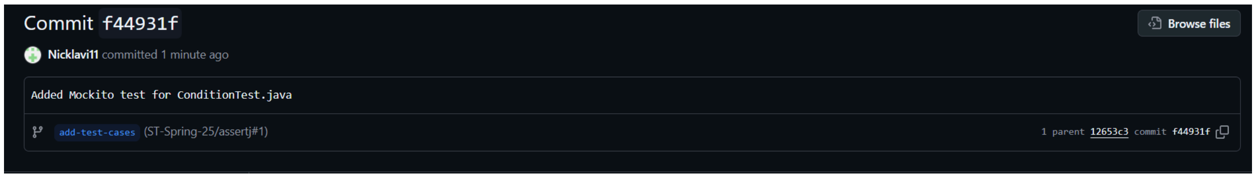
```
[INFO] Running org.assertj.core.api.ConditionTest
Applying configuration org.assertj.core.testkit.ConfigurationForTests
- representation ................................ = StandardRepresentation
- comparingPrivateFieldsEnabled .................. = true
- extractingPrivateFieldsEnabled ................. = true
- bareNamePropertyExtractionEnabled .............. = true
- lenientDateParsingEnabled ...................... = false
- additional date formats ........................ = []
- maxLengthForSingleLineDescription .............. = 80
- maxElementsForPrinting ......................... = 1000
- maxStackTraceElementsDisplayed.................. = 10
- printAssertionsDescription ..................... = false
- descriptionConsumer ............................ = null
- removeAssertJRelatedElementsFromStackTraceEnabled = false
- preferredAssumptionException ................... = AUTO_DETECT(try in order org.testng.SkipException, org.junit.AssumptionViolatedException and org.opentest4j.TestAbortedExcepti
on)

[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.548 s -- in org.assertj.core.api.ConditionTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 2, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  29.528 s
[INFO] Finished at: 2025-04-09T23:05:42-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\assertj\assertj-core>
```

*Commit:*

After the tests passed, I committed it to the repository



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git add .
warning: in the working copy of 'assertj-core/src/test/java/org/assertj/core/api/ConditionTest.java', LF will be replaced by CRLF the next time Git touches it

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git commit -m "Added Mockito test for ConditionTest.java"
[add-test-cases f44931f6a] Added Mockito test for ConditionTest.java
 1 file changed, 32 insertions(+)
 create mode 100644 assertj-core/src/test/java/org/assertj/core/api/ConditionTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 20, done.
Counting objects: 100% (20/20), done.
Delta compression using up to 16 threads
Compressing objects: 100% (10/10), done.
Writing objects: 100% (11/11), 1.09 KiB | 1.09 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/assertj.git
   12653c350..f44931f6a  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/assertj (add-test-cases)
$ |
```



Commit f44931f

Nicklavi11 committed 1 minute ago

```
Added Mockito test for ConditionTest.java
```

add-test-cases (ST-Spring-25/assertj#1)    1 parent 12653c3 commit f44931f

Here is the link to the commit:

[https://github.com/ST-Spring-25/assertj/commit/f44931f6a0275f9c0c1f750aa369fd5296bcca88](https://github.com/ST-Spring-25/assertj/commit/f44931f6a0275f9c0c1f750aa369fd5296bcca88)

## Commons-text:

Apache Commons Text is a Java library that gives developers utilities for working with and manipulating strings. This improves the Java String class by giving it more flexibility, and it is a part of other Apache Commons libraries. Commons text focuses on advanced string operations, and its features are string similarity metrics, random string generators, escaping/unescaping, string substitution, etc. This is a great candidate for creating tests because it has great utility classes with clear input and output, and it also supports many black-box testing techniques.

Here is the link for the pull request made for commons-text:

https://github.com/ST-Spring-25/commons-text/pull/3

## Test Case #7

*What is the class and method I am testing:*

File: WordUtils.java

- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/main/java/org/apache/commons/text/WordUtils.java
- (~/commons-text/src/main/java/org/apache/commons/text/WordUtils.java)

Class: The WordUtils class has utility methods that have several different operations in strings.

Method: The capitalize(String str) method takes the input string and capitalizes the first character of each word. It returns the converted uppercase words, and if the input is null, it returns null. If it is empty, it returns an empty string. The method does not change any whitespace and the case of other characters.

*What is the test:*

File: CapitalizeTest.java

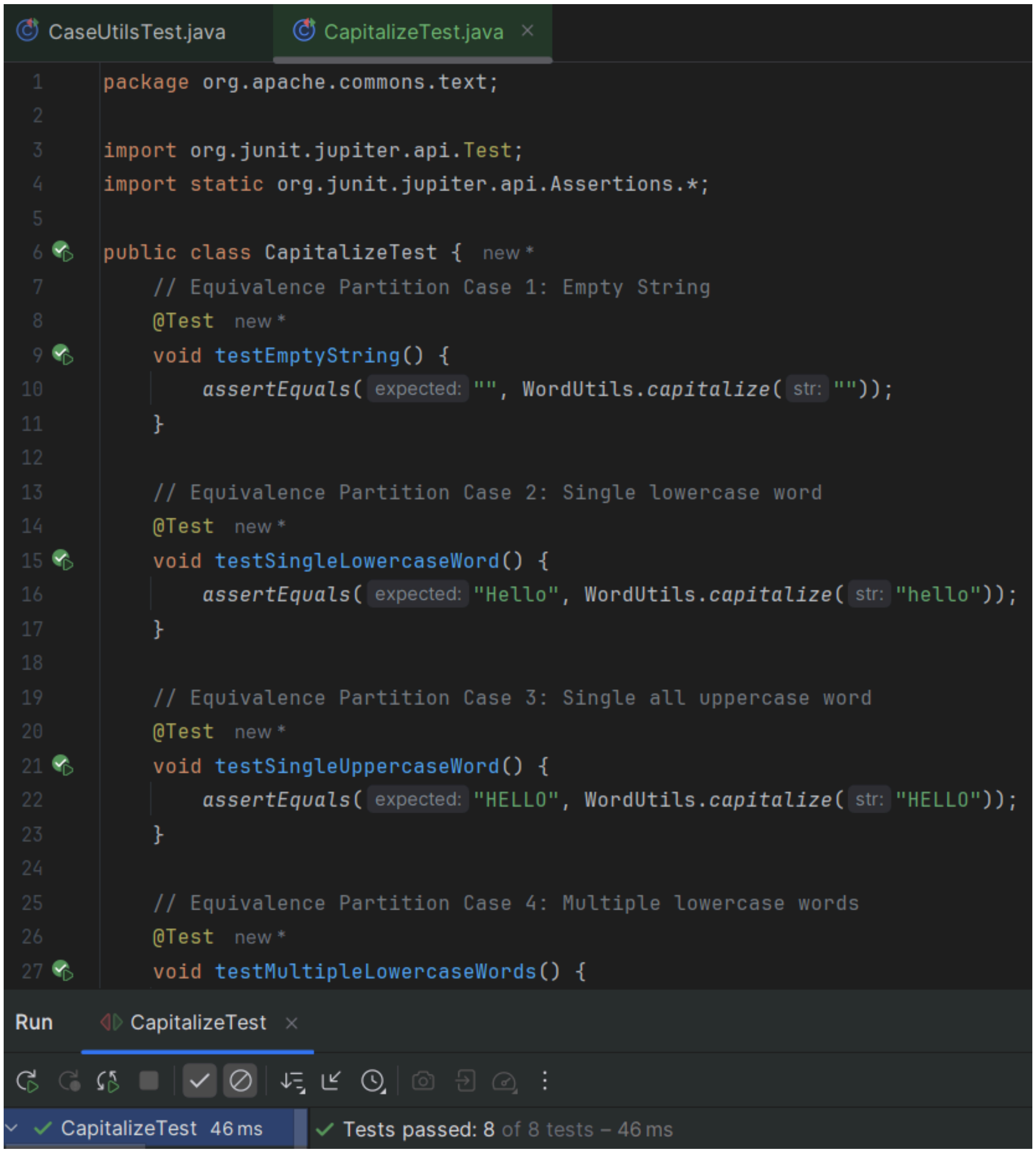
- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/test/java/org/apache/commons/text/CapitalizeTest.java
- (~/commons-text/src/test/java/org/apache/commons/text/CapitalizeTest.java)

Structure: I have made 8 different test cases that cover a variety of different types of inputs. It also uses JUnit.

Technique: The technique used is Equivalence Partitioning. It tests inputs that represent categories of behavior, such as valid simple inputs, null, empty string, mixed or complex formatting, to see how the method behaves around several different input types.

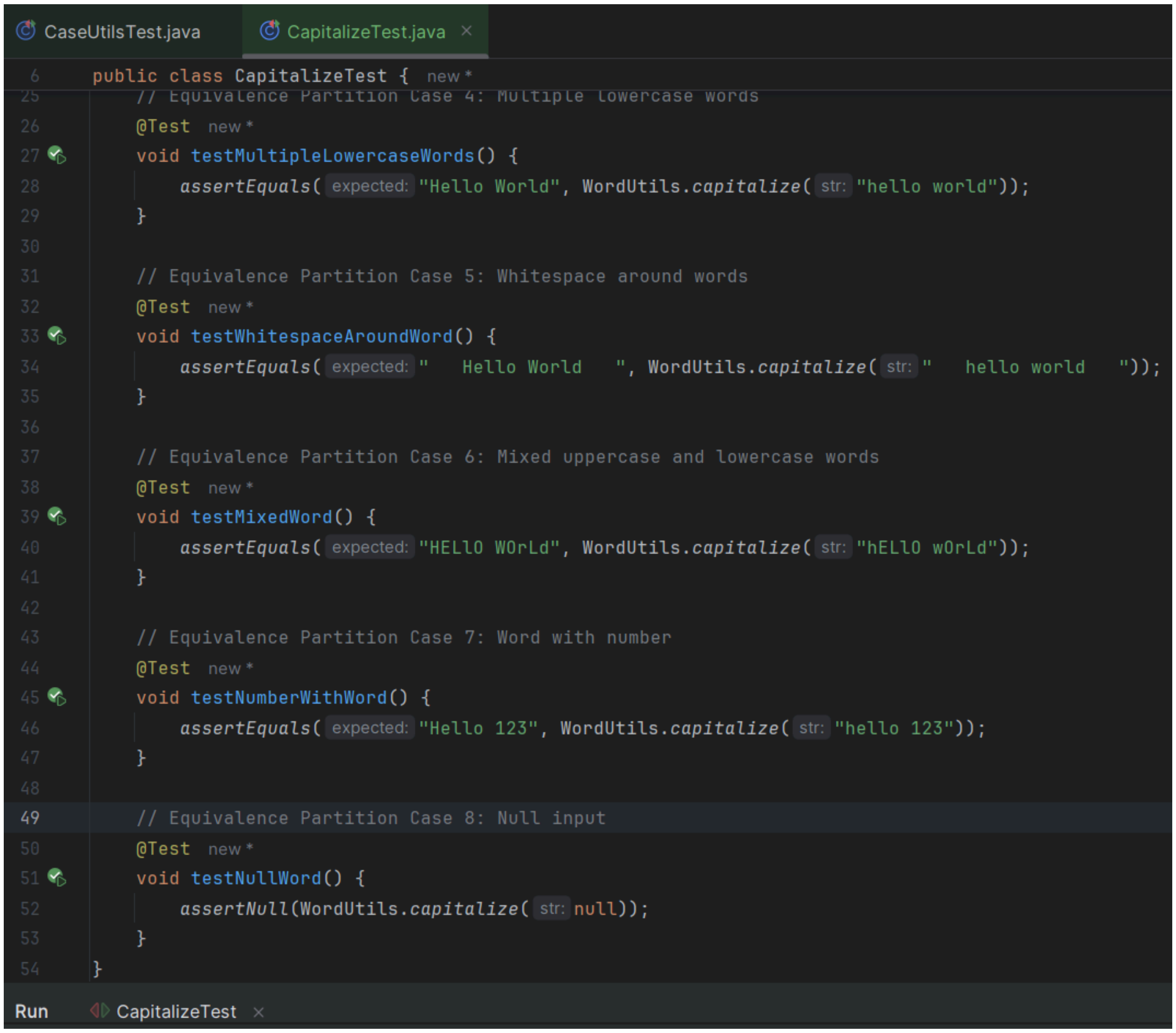
Why: These are good tests because they cover a variety of major use cases in the method. It checks many common input types while also checking boundary and edge cases. Each test is very simple, clear, and fast to run, which makes them good test cases to make sure the method in this class works properly.

Here are the pictures of the tests passing and evidence of running it through Maven

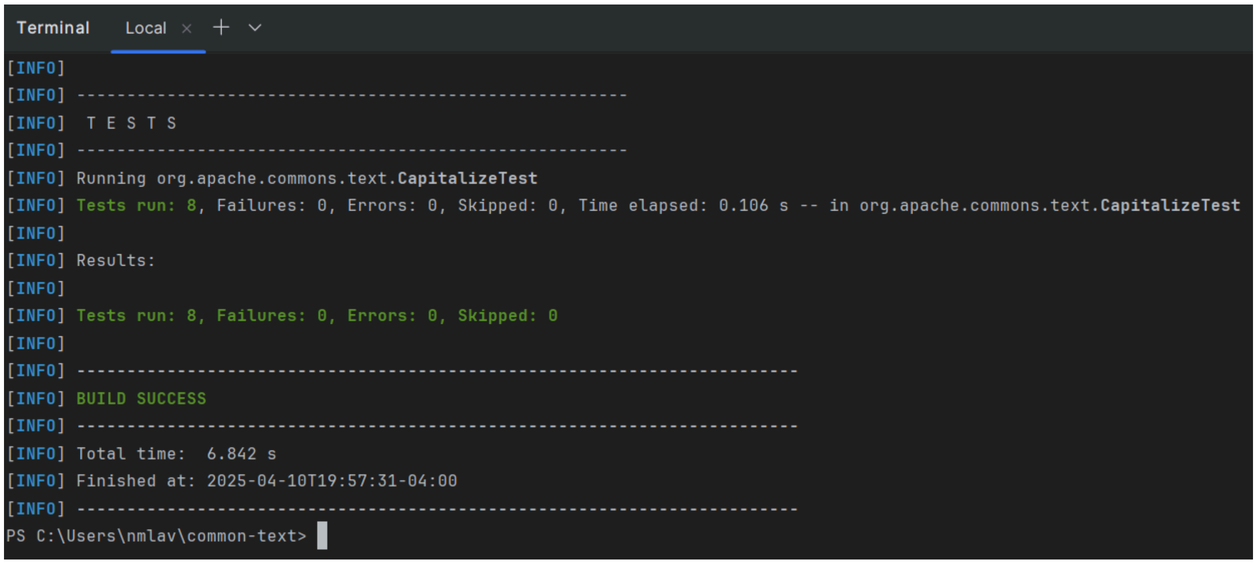


```java
package org.apache.commons.text;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

public class CapitalizeTest {  new *
    // Equivalence Partition Case 1: Empty String
    @Test  new *
    void testEmptyString() {
        assertEquals( expected: "", WordUtils.capitalize( str: ""));
    }

    // Equivalence Partition Case 2: Single lowercase word
    @Test  new *
    void testSingleLowercaseWord() {
        assertEquals( expected: "Hello", WordUtils.capitalize( str: "hello"));
    }

    // Equivalence Partition Case 3: Single all uppercase word
    @Test  new *
    void testSingleUppercaseWord() {
        assertEquals( expected: "HELLO", WordUtils.capitalize( str: "HELLO"));
    }

    // Equivalence Partition Case 4: Multiple lowercase words
    @Test  new *
    void testMultipleLowercaseWords() {
```

```
✓ CapitalizeTest 46 ms    ✓ Tests passed: 8 of 8 tests – 46 ms
```



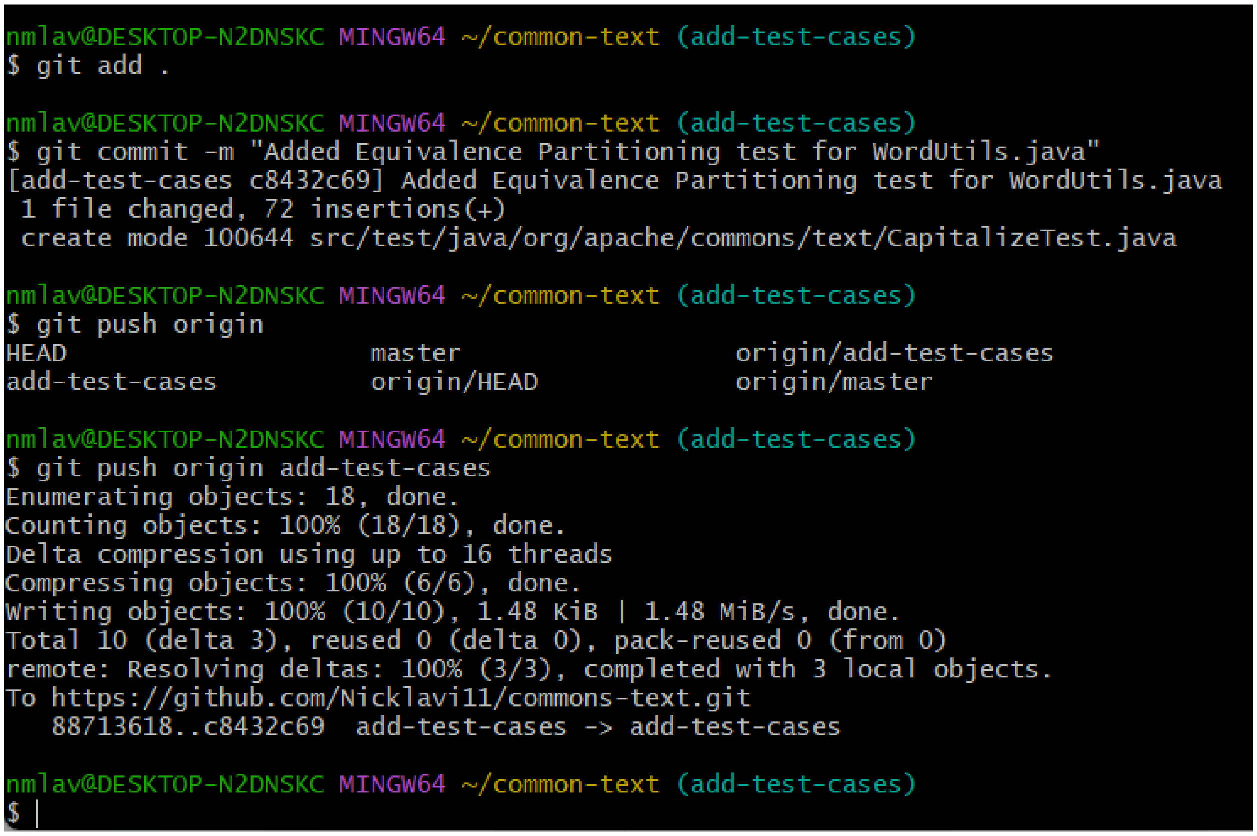
```java
public class CapitalizeTest {  new *
    // Equivalence Partition Case 4: Multiple lowercase words
    @Test  new *
    void testMultipleLowercaseWords() {
        assertEquals( expected: "Hello World", WordUtils.capitalize( str: "hello world"));
    }

    // Equivalence Partition Case 5: Whitespace around words
    @Test  new *
    void testWhitespaceAroundWord() {
        assertEquals( expected: "   Hello World    ", WordUtils.capitalize( str: "   hello world    "));
    }

    // Equivalence Partition Case 6: Mixed uppercase and lowercase words
    @Test  new *
    void testMixedWord() {
        assertEquals( expected: "HELlO WOrLd", WordUtils.capitalize( str: "hELlO wOrLd"));
    }

    // Equivalence Partition Case 7: Word with number
    @Test  new *
    void testNumberWithWord() {
        assertEquals( expected: "Hello 123", WordUtils.capitalize( str: "hello 123"));
    }

    // Equivalence Partition Case 8: Null input
    @Test  new *
    void testNullWord() {
        assertNull(WordUtils.capitalize( str: null));
    }
}
```



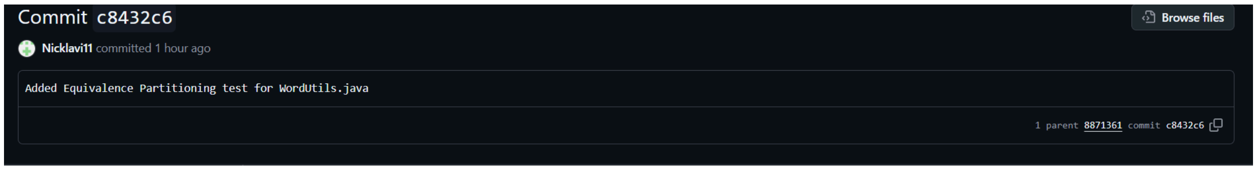
```
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.apache.commons.text.CapitalizeTest
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.106 s -- in org.apache.commons.text.CapitalizeTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 8, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  6.842 s
[INFO] Finished at: 2025-04-10T19:57:31-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\common-text>
```

*Commit:*

After the tests passed, I committed it to the repository:



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/common-text (add-test-cases)
$ git add .

nmlav@DESKTOP-N2DNSKC MINGW64 ~/common-text (add-test-cases)
$ git commit -m "Added Equivalence Partitioning test for WordUtils.java"
[add-test-cases c8432c69] Added Equivalence Partitioning test for WordUtils.java
 1 file changed, 72 insertions(+)
 create mode 100644 src/test/java/org/apache/commons/text/CapitalizeTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/common-text (add-test-cases)
$ git push origin
HEAD                    master                  origin/add-test-cases
add-test-cases          origin/HEAD             origin/master

nmlav@DESKTOP-N2DNSKC MINGW64 ~/common-text (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 1.48 KiB | 1.48 MiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Nicklavi11/commons-text.git
   88713618..c8432c69  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/common-text (add-test-cases)
$ |
```



Commit c8432c6

Nicklavi11 committed 1 hour ago

```
Added Equivalence Partitioning test for WordUtils.java
```

1 parent 8871361 commit c8432c6

Link for this commit:

[https://github.com/ST-Spring-25/commons-text/pull/3/commits/c8432c69c0408b6e0abd4f3874431f969b129874](https://github.com/ST-Spring-25/commons-text/pull/3/commits/c8432c69c0408b6e0abd4f3874431f969b129874)

## Test Case #8

*What is the class and method I am testing:*

File: JaroWinklerDistance.java

- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/main/java/org/apache/commons/text/similarity/JaroWinklerDistance.java
- (~/commons-text/src/main/java/org/apache/commons/text/similarity/JaroWinklerDistance.java)

Class: The JaroWinklerDistance class provides an implementation of the Jaro-Winkler distance algorithm. This algorithm measures the similarity between two character sequences. It applies a distance to the measurement; a 1.0 is not similar at all, and 0.0 is perfectly similar.

Method: The method I am testing is apply(CharSequence left, CharSequence right). It returns a Double representing how different the 2 strings are and throws an IllegalArgumentException if one of the inputs is null. This method is useful for checking the spelling of words, and it measures the distance between strings.

*What is the test:*

File: JaroWinklerDistanceScoreTest.java

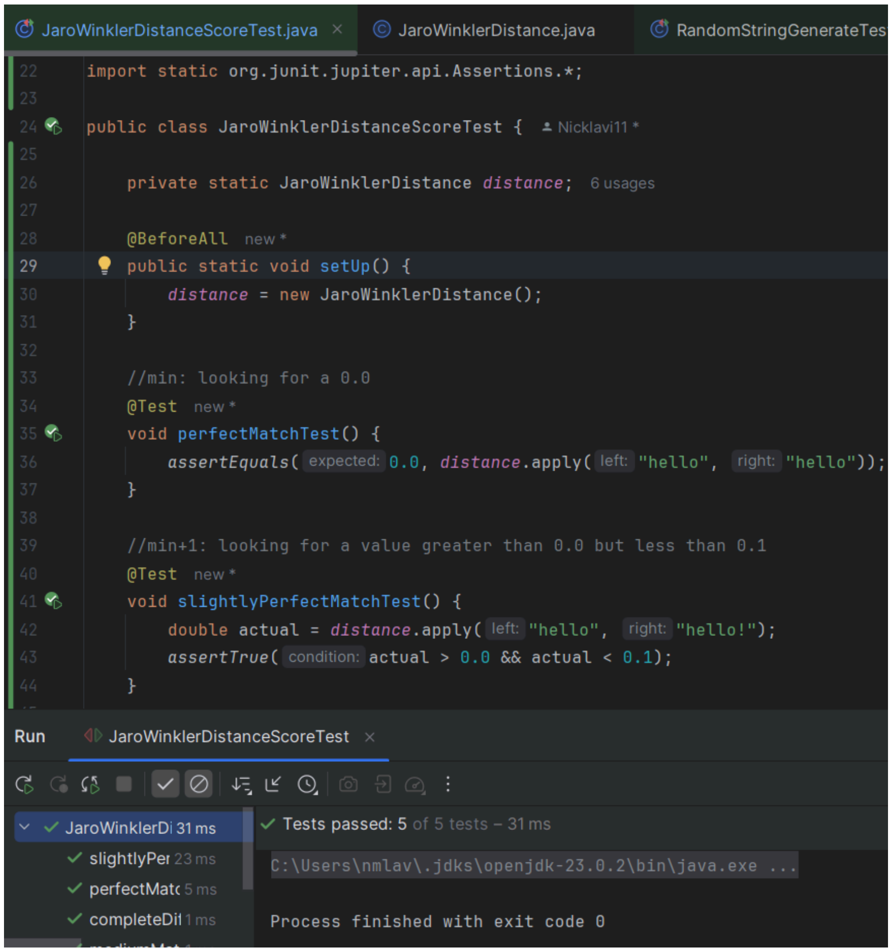
- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScoreTest.java
- (~/commons-text/src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScoreTest.java)

Structure: The structure of the test uses a common test case, which is the word "hello". This keeps the testing consistent with a similar case for each test. It also uses JUnit.

Technique: The technique used is Boundary Value testing, which covers both ends of the score range and a typical case in between. The min is a perfect match (0.0), the min+1 is a slightly different match (greater than 0.0, less than 0.1), the typical is a moderate similarity (greater than 0.1, less than 0.5), and max-1 is an almost completely different match (greater than 0.5, less than 1.0), and the max is a completely different match(1.0). I rounded each score to the nearest hundredth, and I double checked this algorithm with an online Jaro-Winkler distance calculator (Link to website: [https://tilores.io/jaro-winkler-distance-algorithm-online-tool](https://tilores.io/jaro-winkler-distance-algorithm-online-tool)).

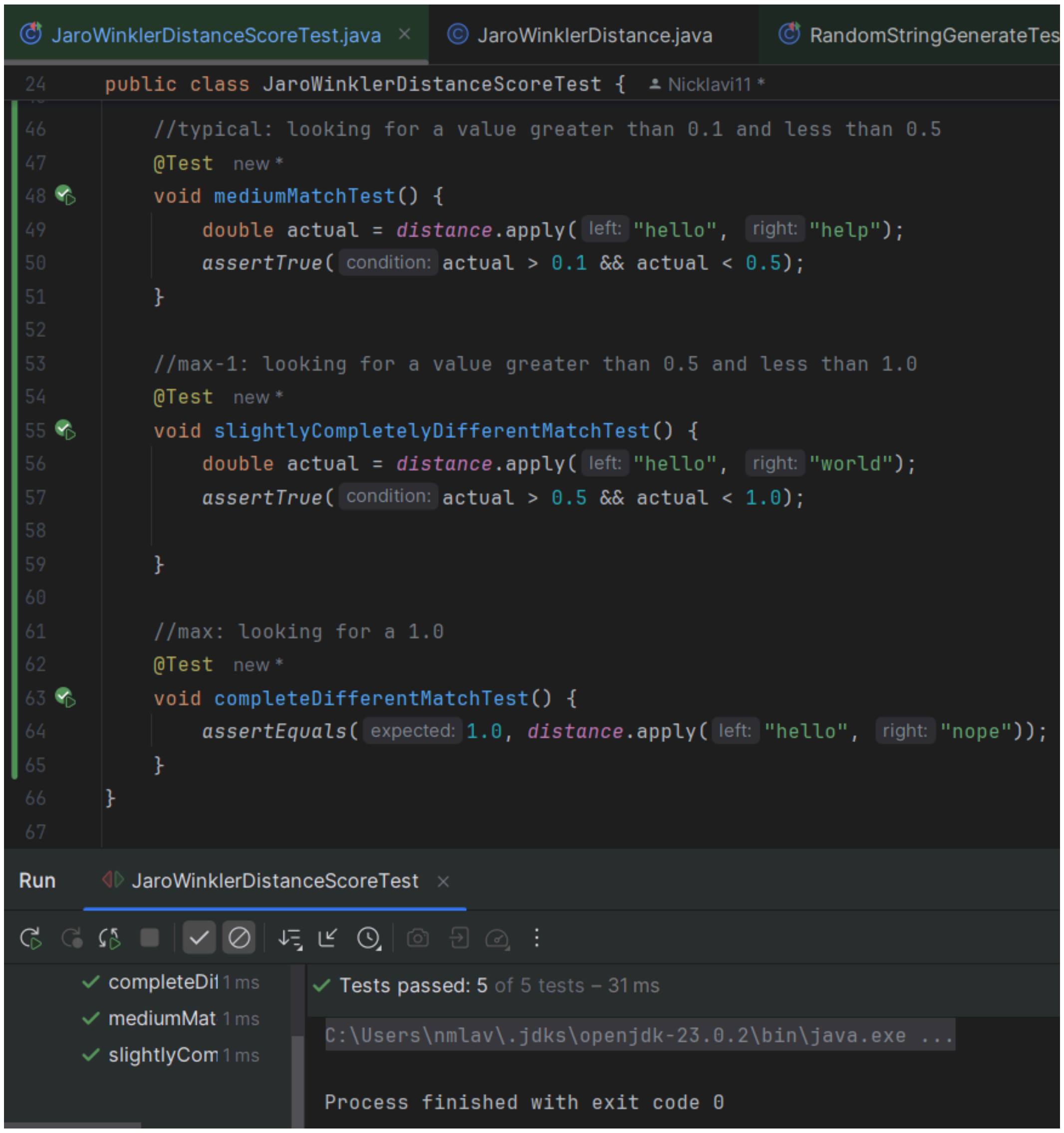
Why: This is a good test because it covers the full range of behavior for the method and tests exact matches, partial similarity, and completely different cases. It uses readable and clear assertions with measurable ranges, and the tests are double checks with an outside source to confirm that the tests are valid.

Here are the pictures of the tests passing and evidence of running it through Maven


JaroWinklerDistanceScoreTest.java
JaroWinklerDistance.java
RandomStringGenerateTes
Run
JaroWinklerDistanceScoreTest

```
import static org.junit.jupiter.api.Assertions.*;

public class JaroWinklerDistanceScoreTest {  Nicklavi11 *

    private static JaroWinklerDistance distance;  6 usages

    @BeforeAll  new *
    public static void setUp() {
        distance = new JaroWinklerDistance();
    }

    //min: looking for a 0.0
    @Test  new *
    void perfectMatchTest() {
        assertEquals( expected: 0.0, distance.apply( left: "hello",  right: "hello"));
    }

    //min+1: looking for a value greater than 0.0 but less than 0.1
    @Test  new *
    void slightlyPerfectMatchTest() {
        double actual = distance.apply( left: "hello",  right: "hello!");
        assertTrue( condition: actual > 0.0 && actual < 0.1);
    }
```

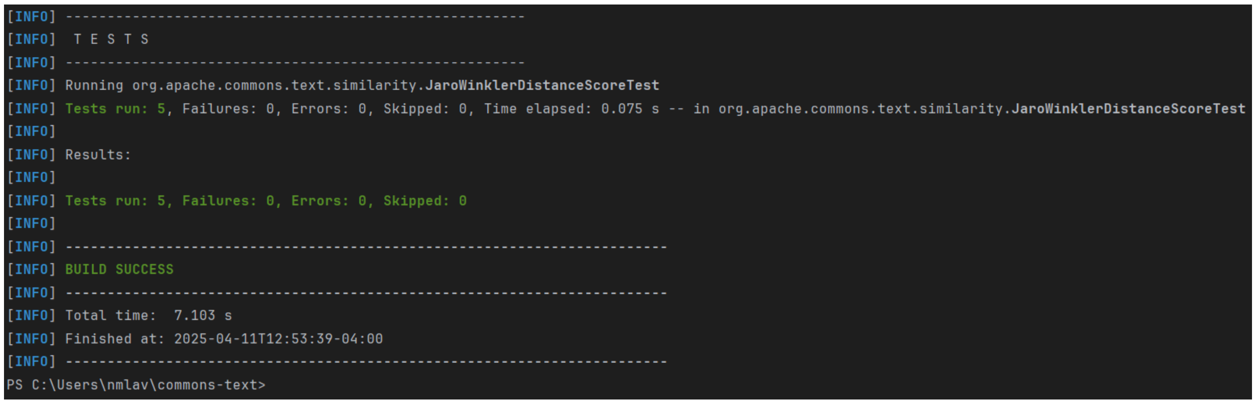
```
JaroWinklerDi 31 ms
  slightlyPer 23 ms
  perfectMatc 5 ms
  completeDi 1 ms

Tests passed: 5 of 5 tests – 31 ms

C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```


JaroWinklerDistanceScoreTest.java × | JaroWinklerDistance.java | RandomStringGenerateTes

```
public class JaroWinklerDistanceScoreTest {  Nicklavi11 *
    //typical: looking for a value greater than 0.1 and less than 0.5
    @Test  new *
    void mediumMatchTest() {
        double actual = distance.apply( left: "hello",  right: "help");
        assertTrue( condition: actual > 0.1 && actual < 0.5);
    }

    //max-1: looking for a value greater than 0.5 and less than 1.0
    @Test  new *
    void slightlyCompletelyDifferentMatchTest() {
        double actual = distance.apply( left: "hello",  right: "world");
        assertTrue( condition: actual > 0.5 && actual < 1.0);

    }

    //max: looking for a 1.0
    @Test  new *
    void completeDifferentMatchTest() {
        assertEquals( expected: 1.0, distance.apply( left: "hello",  right: "nope"));
    }
}
```

Run  JaroWinklerDistanceScoreTest ×

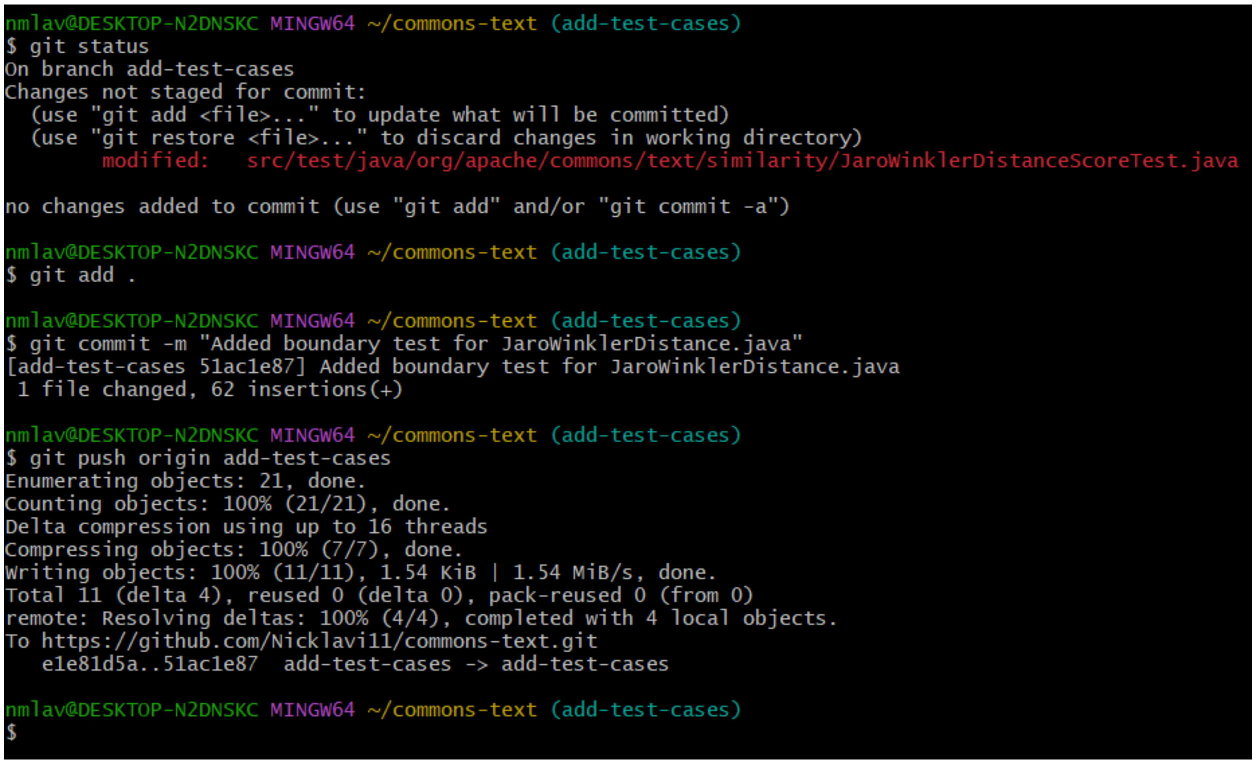
```
✓ completeDi 1 ms      ✓ Tests passed: 5 of 5 tests – 31 ms
✓ mediumMat 1 ms       C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...
✓ slightlyCom 1 ms
                       Process finished with exit code 0
```



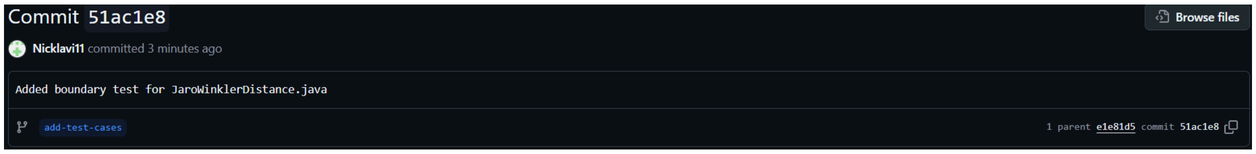
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.apache.commons.text.similarity.JaroWinklerDistanceScoreTest
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.075 s -- in org.apache.commons.text.similarity.JaroWinklerDistanceScoreTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 5, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  7.103 s
[INFO] Finished at: 2025-04-11T12:53:39-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\commons-text>
```

*Commit:*

After the tests passed, I committed it to the repository:



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git status
On branch add-test-cases
Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScoreTest.java

no changes added to commit (use "git add" and/or "git commit -a")

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git add .

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git commit -m "Added boundary test for JaroWinklerDistance.java"
[add-test-cases 51ac1e87] Added boundary test for JaroWinklerDistance.java
 1 file changed, 62 insertions(+)

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 16 threads
Compressing objects: 100% (7/7), done.
Writing objects: 100% (11/11), 1.54 KiB | 1.54 MiB/s, done.
Total 11 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/commons-text.git
   e1e81d5a..51ac1e87  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$
```



Commit 51ac1e8
Browse files
Nicklavi11 committed 3 minutes ago

```
Added boundary test for JaroWinklerDistance.java
```

add-test-cases
1 parent e1e81d5 commit 51ac1e8

Link for this commit:

https://github.com/ST-Spring-25/commons-text/pull/3/commits/51ac1e874a2b972c46e2

9aac0bcd47d89b56c047

**Test Case #9**

*What is the class and method I am testing:*

File: RandomStringGenerator.java

- https://github.com/Nicklavi11/commons-text/blob/master/src/main/java/org/apache/commons/text/RandomStringGenerator.java
- (~/commons-text/src/main/java/org/apache/commons/text/RandomStringGenerator.java)

Class: The RandomStringGenerator class is a utility class that is used to generate random strings composed of characters from a specified range. It is a useful class in testing, password creation, data generation, and several other scenarios where random character sequences are needed.

Method: The method I am testing is generate(int length), which creates a random string based on the given length in the input using the character rand defined in the generator's builder. In this test, the generator that was built uses characters from 'a' to 'a', and the method is supposed to return a string of 10 lowercase letters.

*What is the test:*

File: RandomStringGenerateTest.java

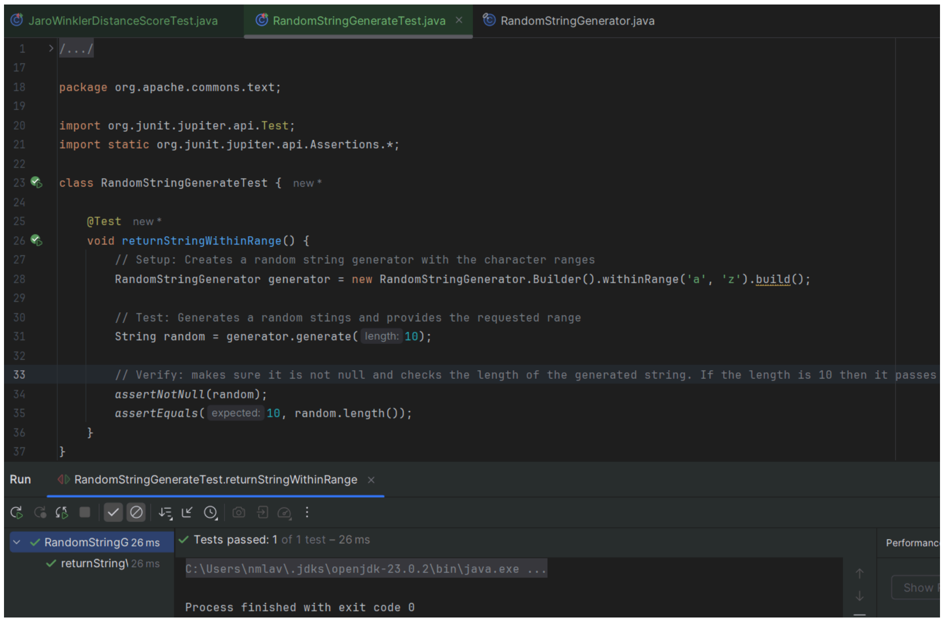
- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/test/java/org/apache/commons/text/RandomStringGenerateTest.java
- (~/commons-text/src/test/java/org/apache/commons/text/RandomStringGenerateTest.java)

Structure: The structure of the test is to generate a random string with any lowercase letter of length 10. It uses the anatomy of a test, and it checks if it is null and that the length is correct. It uses JUnit as well.

Technique: The technique used is black-box functional testing. It focuses on the output and makes sure that it is not null and exactly the given length (10 characters). It does not check how the string is built, which is normal for testing random behavior.

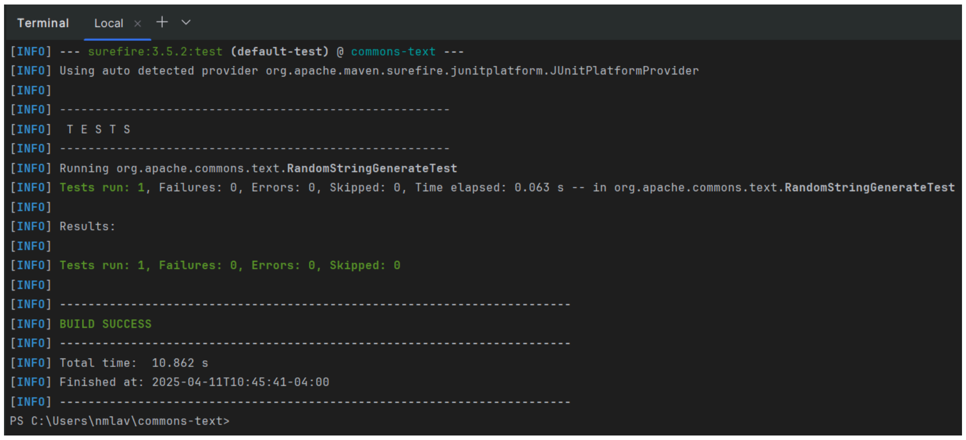
Why: This is a good test because it checks the generate() method for its proper length and makes sure it is not null. It also makes sure the random string was built correctly in the given range. It has clear and fast assertions, and is also easy to read, which is great for automated testing.

Here are the pictures of the tests passing and evidence of running it through Maven


JaroWinklerDistanceScoreTest.java
RandomStringGenerateTest.java
RandomStringGenerator.java

```java
/.../

package org.apache.commons.text;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class RandomStringGenerateTest {  new *

    @Test  new *
    void returnStringWithinRange() {
        // Setup: Creates a random string generator with the character ranges
        RandomStringGenerator generator = new RandomStringGenerator.Builder().withinRange('a', 'z').build();

        // Test: Generates a random stings and provides the requested range
        String random = generator.generate( length: 10);

        // Verify: makes sure it is not null and checks the length of the generated string. If the length is 10 then it passes
        assertNotNull(random);
        assertEquals( expected: 10, random.length());
    }
}
```

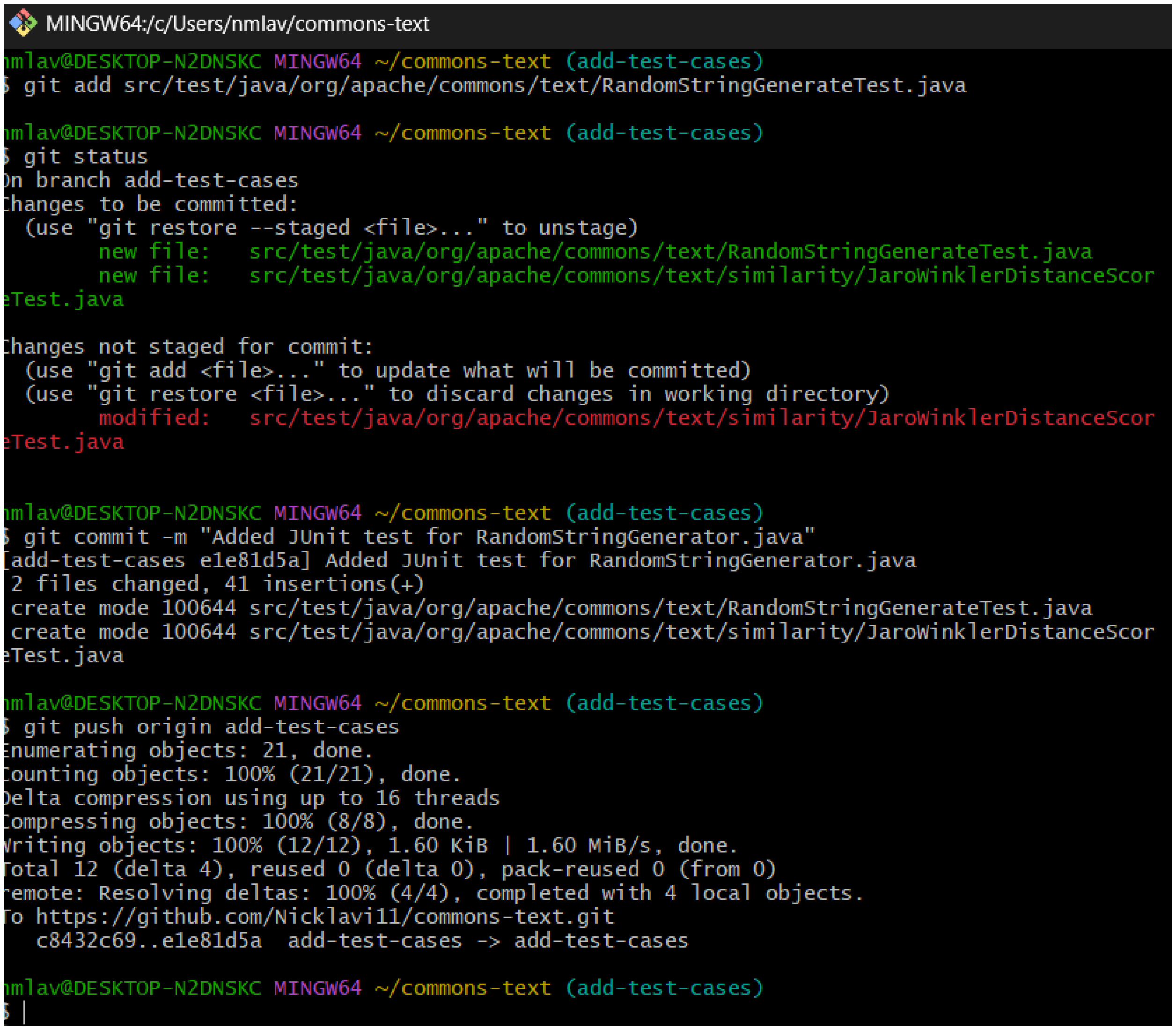
```
✓ Tests passed: 1 of 1 test – 26 ms
C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



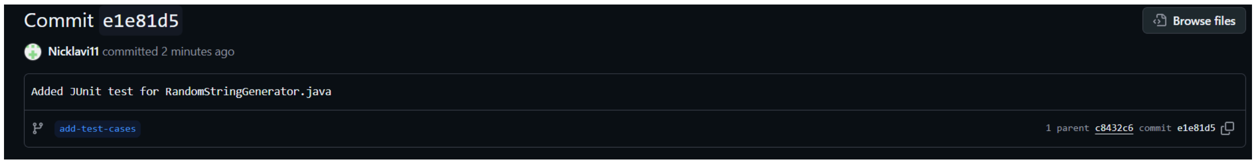
```
[INFO] --- surefire:3.5.2:test (default-test) @ commons-text ---
[INFO] Using auto detected provider org.apache.maven.surefire.junitplatform.JUnitPlatformProvider
[INFO]
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.apache.commons.text.RandomStringGenerateTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.063 s -- in org.apache.commons.text.RandomStringGenerateTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  10.862 s
[INFO] Finished at: 2025-04-11T10:45:41-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\commons-text>
```

*Commit:*

After the tests passed, I committed it to the repository:



```
MINGW64:/c/Users/nmlav/commons-text
nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git add src/test/java/org/apache/commons/text/RandomStringGenerateTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git status
On branch add-test-cases
Changes to be committed:
  (use "git restore --staged <file>..." to unstage)
        new file:   src/test/java/org/apache/commons/text/RandomStringGenerateTest.java
        new file:   src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScor
eTest.java

Changes not staged for commit:
  (use "git add <file>..." to update what will be committed)
  (use "git restore <file>..." to discard changes in working directory)
        modified:   src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScor
eTest.java


nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git commit -m "Added JUnit test for RandomStringGenerator.java"
[add-test-cases e1e81d5a] Added JUnit test for RandomStringGenerator.java
 2 files changed, 41 insertions(+)
 create mode 100644 src/test/java/org/apache/commons/text/RandomStringGenerateTest.java
 create mode 100644 src/test/java/org/apache/commons/text/similarity/JaroWinklerDistanceScor
eTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 21, done.
Counting objects: 100% (21/21), done.
Delta compression using up to 16 threads
Compressing objects: 100% (8/8), done.
Writing objects: 100% (12/12), 1.60 KiB | 1.60 MiB/s, done.
Total 12 (delta 4), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (4/4), completed with 4 local objects.
To https://github.com/Nicklavi11/commons-text.git
   c8432c69..e1e81d5a  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ |
```



Commit e1e81d5

Nicklavi11 committed 2 minutes ago

Added JUnit test for RandomStringGenerator.java

add-test-cases

1 parent c8432c6 commit e1e81d5

Link for this commit:

[https://github.com/ST-Spring-25/commons-text/pull/3/commits/e1e81d5aaf97e05b8e928
28f4bba69a97b1fd9ae](https://github.com/ST-Spring-25/commons-text/pull/3/commits/e1e81d5aaf97e05b8e92828f4bba69a97b1fd9ae)

**Test Case #10**

*What is the class and method I am testing:*

File: StringEscapeUtils.java

- https://github.com/ST-Spring-25/commons-text/blob/master/src/main/java/org/apache/commons/text/StringEscapeUtils.java
- (~/commons-text/src/main/java/org/apache/commons/text/StringEscapeUtils.java)

Class: The StringEscapeUtils class is a utility class that has methods to escape and unescape strings for many different contexts, such as Java, HTML, XML, and JSON. These methods are helpful when dealing with strings that have special characters that need to be encoded safely.

Method: The method I am testing is escapeJava(String str). This method escapes special characters, so it is safe to have in Java source code. So if there is a (“) it becomes a (\”), and a backslash (\) becomes (\\).

*What is the test:*

File: StringEscapeUtilsTopDownTest.java

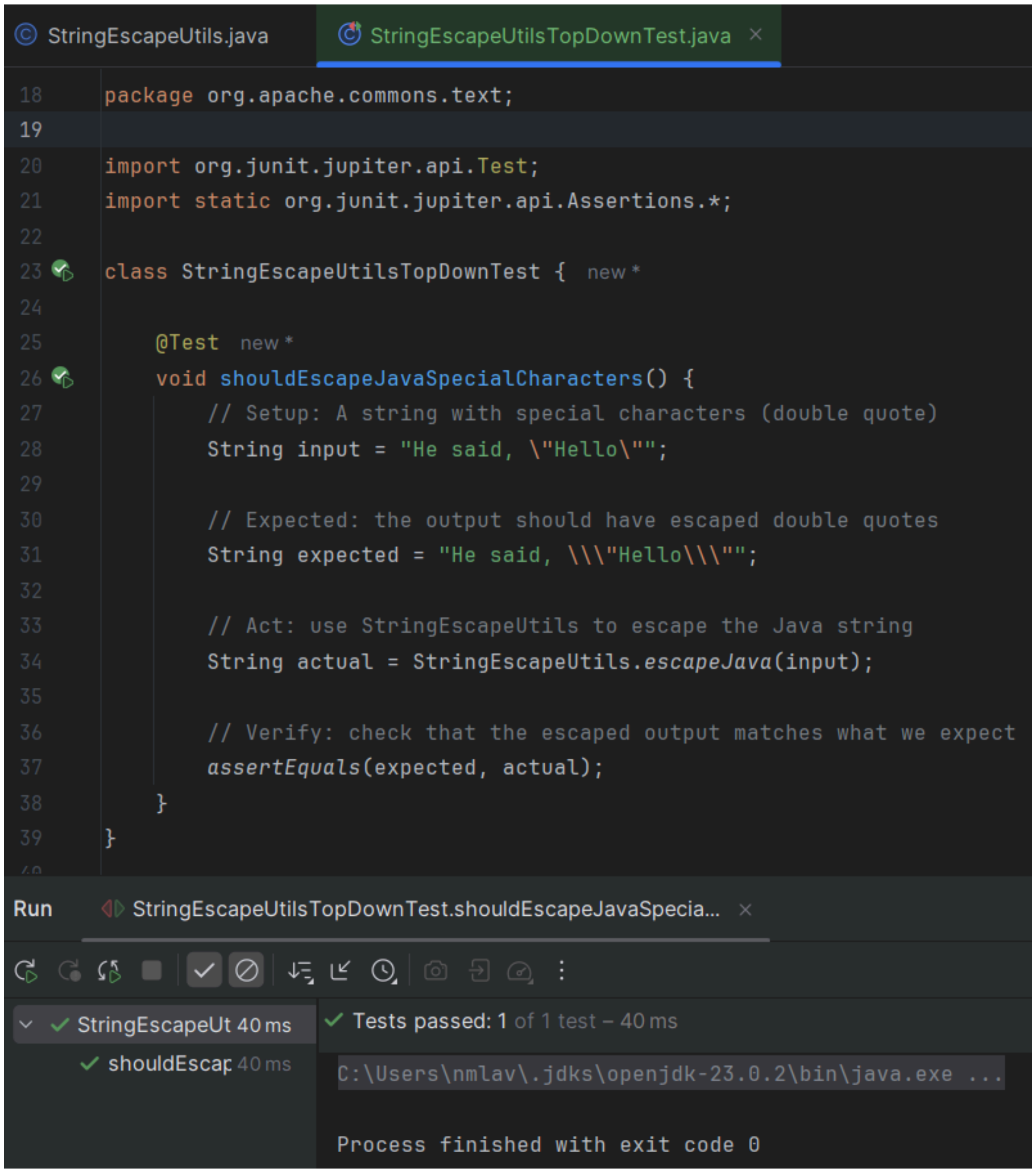
- https://github.com/Nicklavi11/commons-text/blob/add-test-cases/src/test/java/org/apache/commons/text/StringEscapeUtilsTopDownTest.java
- (~/commons-text/src/test/java/org/apache/commons/text/StringEscapeUtilsTopDownTest.java)

Structure: The structure of this test takes the input of a string that has a double quote and has an expected output that it verifies with the same quote but escapes. It uses JUnit as well.

Technique: The technique it uses is top-down testing. It takes a high-level method with an input that is known and compares the result against the known correct output. This test does not check how escaping is implemented, but it confirms the behavior is correct.

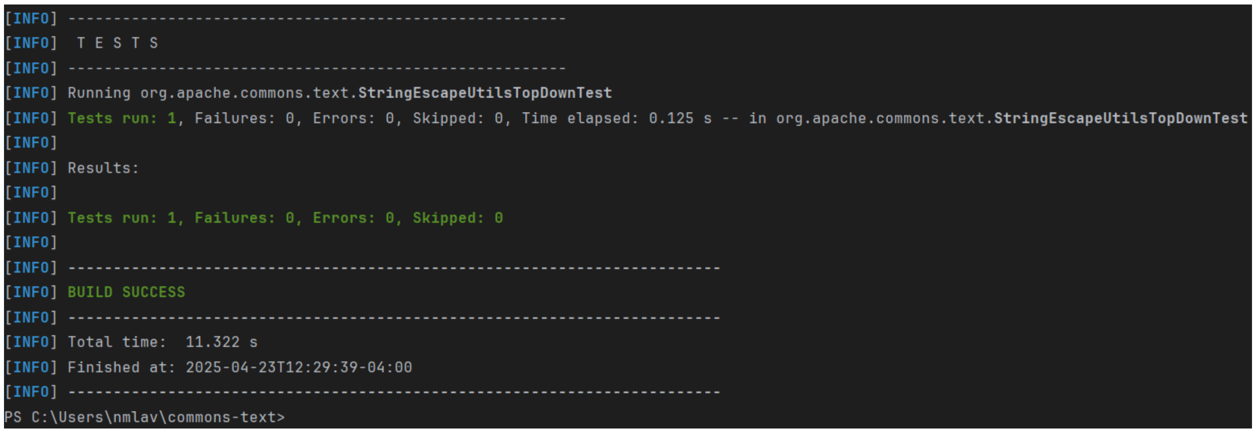
Why: This is a good test because it targets a high-level utility function that is commonly used in Java code generation, and it has a realistic and easy-to-understand example that has special characters. It checks and verifies if it is correct, and the format of the outpu,t and it is short, readable, and very self-contained.

Here are the pictures of the tests passing and evidence of running it through Maven


StringEscapeUtils.java
StringEscapeUtilsTopDownTest.java
Run
StringEscapeUtilsTopDownTest.shouldEscapeJavaSpecia...
StringEscapeUt 40 ms
shouldEscap 40 ms

```java
package org.apache.commons.text;

import org.junit.jupiter.api.Test;
import static org.junit.jupiter.api.Assertions.*;

class StringEscapeUtilsTopDownTest {  new *

    @Test  new *
    void shouldEscapeJavaSpecialCharacters() {
        // Setup: A string with special characters (double quote)
        String input = "He said, \"Hello\"";

        // Expected: the output should have escaped double quotes
        String expected = "He said, \\\"Hello\\\"";

        // Act: use StringEscapeUtils to escape the Java string
        String actual = StringEscapeUtils.escapeJava(input);

        // Verify: check that the escaped output matches what we expect
        assertEquals(expected, actual);
    }
}
```

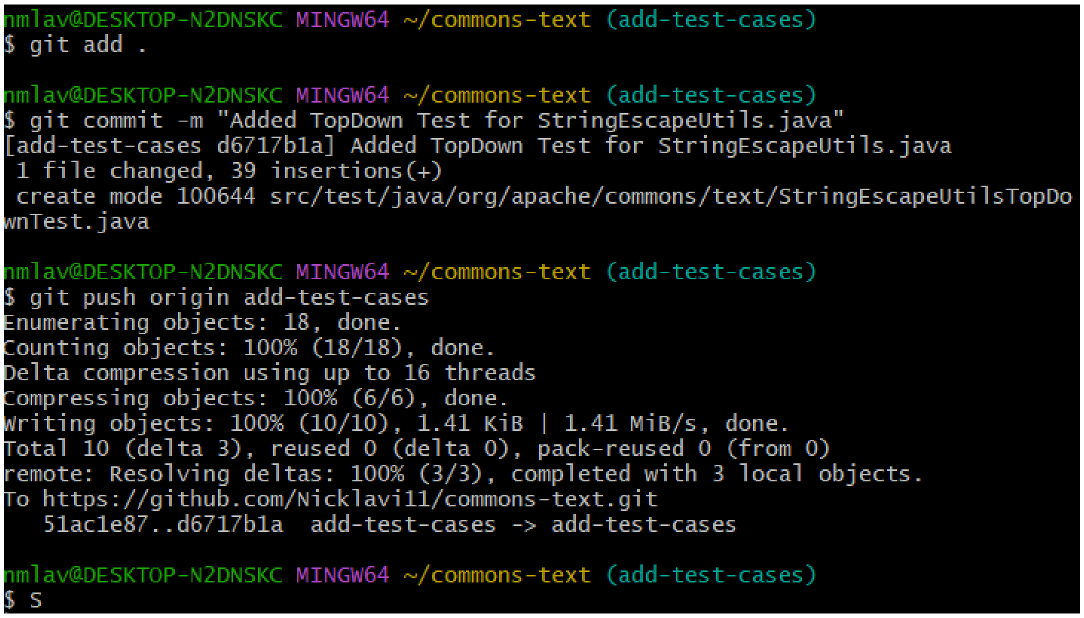
```
Tests passed: 1 of 1 test – 40 ms

C:\Users\nmlav\.jdks\openjdk-23.0.2\bin\java.exe ...

Process finished with exit code 0
```



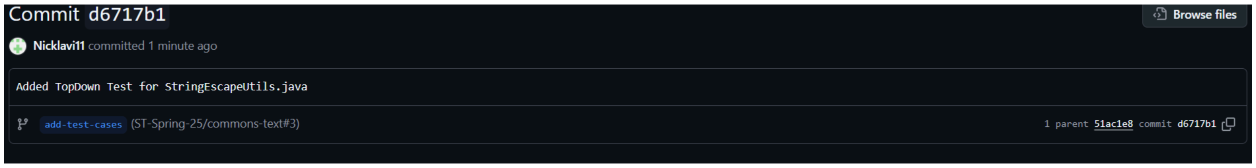
```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running org.apache.commons.text.StringEscapeUtilsTopDownTest
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.125 s -- in org.apache.commons.text.StringEscapeUtilsTopDownTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 1, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  11.322 s
[INFO] Finished at: 2025-04-23T12:29:39-04:00
[INFO] ------------------------------------------------------------------------
PS C:\Users\nmlav\commons-text>
```

*Commit:*

After the tests passed, I committed it to the repository:



```
nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git add .

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git commit -m "Added TopDown Test for StringEscapeUtils.java"
[add-test-cases d6717b1a] Added TopDown Test for StringEscapeUtils.java
 1 file changed, 39 insertions(+)
 create mode 100644 src/test/java/org/apache/commons/text/StringEscapeUtilsTopDo
wnTest.java

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ git push origin add-test-cases
Enumerating objects: 18, done.
Counting objects: 100% (18/18), done.
Delta compression using up to 16 threads
Compressing objects: 100% (6/6), done.
Writing objects: 100% (10/10), 1.41 KiB | 1.41 MiB/s, done.
Total 10 (delta 3), reused 0 (delta 0), pack-reused 0 (from 0)
remote: Resolving deltas: 100% (3/3), completed with 3 local objects.
To https://github.com/Nicklavi11/commons-text.git
   51ac1e87..d6717b1a  add-test-cases -> add-test-cases

nmlav@DESKTOP-N2DNSKC MINGW64 ~/commons-text (add-test-cases)
$ S
```



Commit d6717b1

Nicklavi11 committed 1 minute ago

```
Added TopDown Test for StringEscapeUtils.java
```

add-test-cases (ST-Spring-25/commons-text#3)    1 parent 51ac1e8 commit d6717b1

Link for this commit:

https://github.com/ST-Spring-25/commons-text/pull/3/commits/d6717b1ab6c3fc9146c95abd9aa5c17a0f4df01f

## Group contribution:

I have met with my group several times. My group members are Prabhjot Kaur, Alexander Leali, and Robert Ventura. Each week since this project has been released, we have met through Discord and in class meetings to discuss where we are while working on this project. I have been given a ton of help from Prabhjot Kaur as she has helped me format and create my project report while helping me navigate through many of the projects and find testable methods. Alexander Leali helped me with making my tests better and answered many of my questions that I have had. All of my group members have talked and discussed our report and compared and contrasted them, as well as helped each other find where to get testable methods in many different classes and what are the best open-source projects to test. I helped everyone with many different things such as helping her add required items to make the report better like adding direct links to pull requests and commits while also teaching everyone how to make the pull requests. These weekly group meetings have helped me make my report amazing, and they have been very vital to creating this project report.

## Conclusion

This project has helped me understand how to create real, meaningful test cases for open-source Java libraries. I have learned a lot about reading and analyzing different open-source projects and understanding what real code bases look like, and how I can apply my skills to these. It has also shown me how to navigate and use GitHub better, which is a vital skill that I can forever use. Throughout this project, I have gained a very deep understanding of how to target specific behavior in different classes and methods.

In this project, I have made 10 different test cases, 6 in assertj and 4 in commons-text, and I have used several different testing techniques, such as equivalence partitioning, boundary value testing, mock testing, black-box testing, and top-down testing, which have demonstrated what I have learned throughout this course. I have also practiced with GitHub and worked with forks, branches, commits, and pull requests while verifying each of my tests in Maven.