📍

Junit Tutorial Notes

https://github.com/newtonhobbs/JunitTutorial28Minutes

Date: @February 20, 2023

Topic: Introduction

Recall

Notes

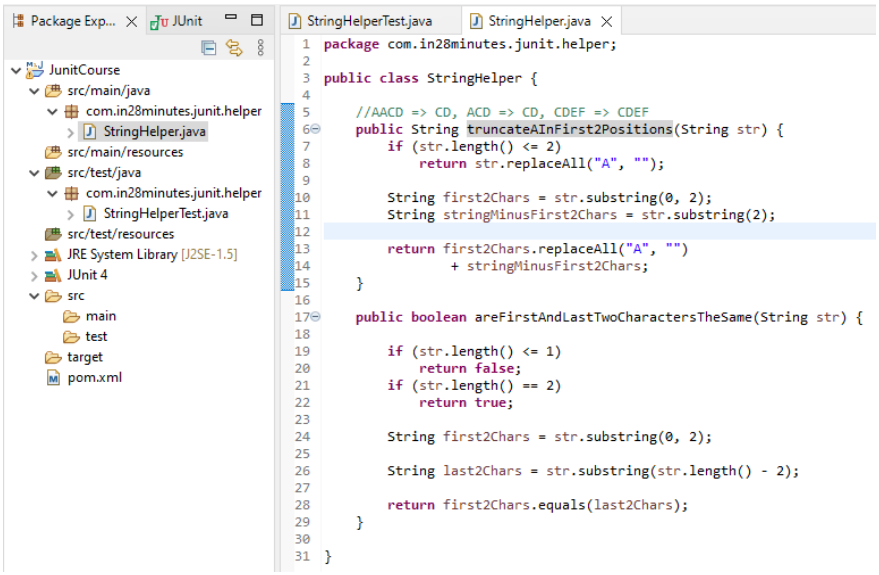
- In src/main/java all the java packages/class are there
- In src/test/java all the packages/class to write the JUnit test case.
- @Test - annotation is use to tells JUnit that the public void method to which it is attached can be run as a test case.

> 💡 **method after the @Test annotation should be public and void**

- ***assertEquals*(expected, actual)**;
  eg: *assertEquals*(”ABC”, “ABCD”); - return failed unit test

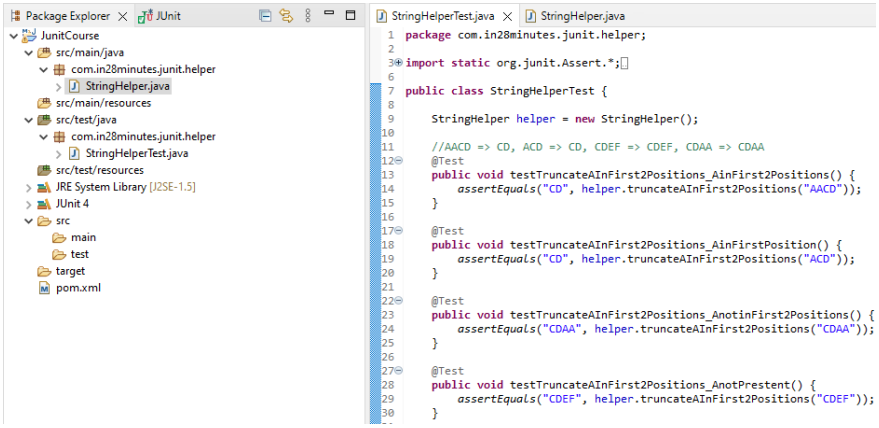
  ***assertFalse*("Message to print after if method return true", method());**
  eg: *assertFalse*("The condition failed", areFirstAndLastTwoCharactersTheSame("ABCD")); - return failed unit test with message “The condition failed”

  ***assertTrue*("Message to print after if method return false", method());**
  eg: *assertTrue*("The condition failed", areFirstAndLastTwoCharactersTheSame("ABCD")); - return failed unit test with message “The condition failed”
- Example:

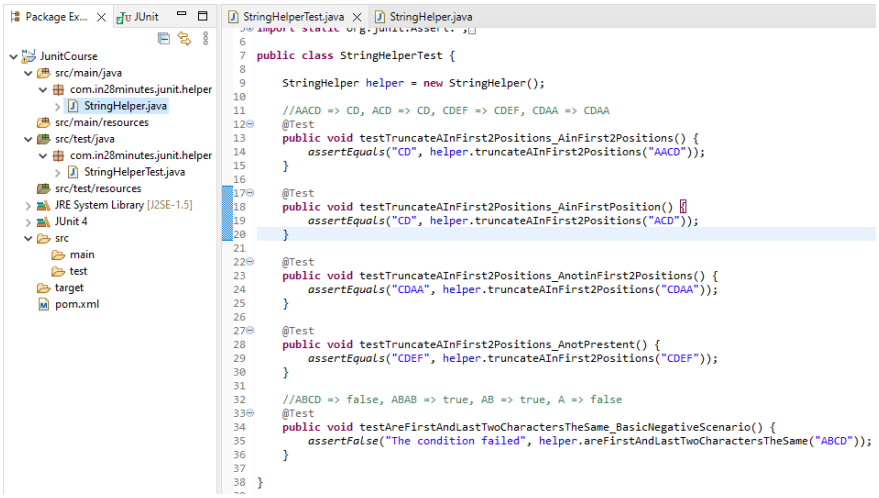


Packages and method for which we have to write the unit test case

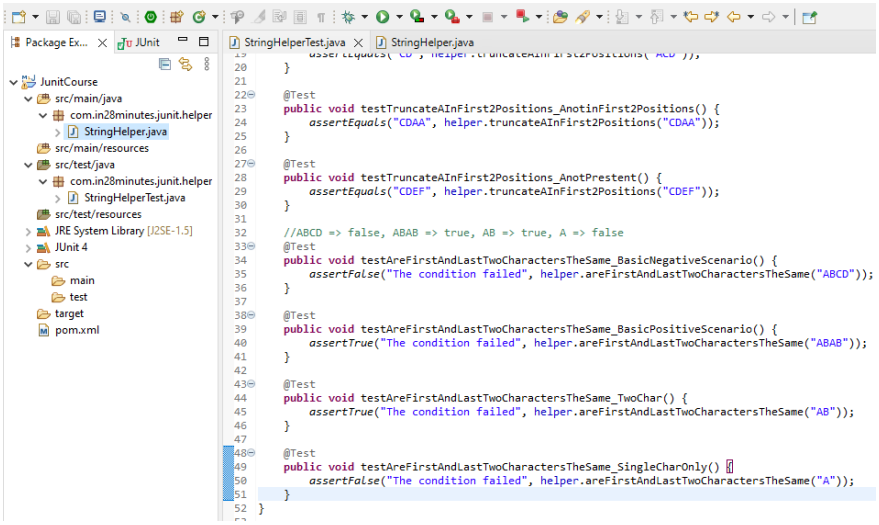
Junit test for above method : public String truncateAInFirst2Positions(String str) (Version:1)



Junit test for above method : public String truncateAInFirst2Positions(String str) (Version:2) || improvement in naming



Junit test for above method : **public boolean** areFirstAndLastTwoCharactersTheSame(String str) (Version:1) || using of *assertFalse*



Junit test for above method : **public boolean** areFirstAndLastTwoCharactersTheSame(String str) (Version:2) || using of *assertTrue*

- Code

```
package com.in28minutes.junit.helper;

public class StringHelper {

  //AACD => CD, ACD => CD, CDEF => CDEF, CDAA => CDAA
  public String truncateAInFirst2Positions(String str) {
    if (str.length() <= 2)
      return str.replaceAll("A", "");

    String first2Chars = str.substring(0, 2);
    String stringMinusFirst2Chars = str.substring(2);

    return first2Chars.replaceAll("A", "")
        + stringMinusFirst2Chars;
  }

  //ABCD => false, ABAB => true, AB => true, A => false
  public boolean areFirstAndLastTwoCharactersTheSame(String str) {
```

```
    if (str.length() <= 1)
      return false;
    if (str.length() == 2)
      return true;

    String first2Chars = str.substring(0, 2);

    String last2Chars = str.substring(str.length() - 2);

    return first2Chars.equals(last2Chars);
  }

}
```

```
package com.in28minutes.junit.helper;

import static org.junit.Assert.*;

import org.junit.Test;

public class StringHelperTest {

  StringHelper helper = new StringHelper();

  //AACD => CD, ACD => CD, CDEF => CDEF, CDAA => CDAA
  @Test
  public void testTruncateAInFirst2Positions_AinFirst2Positions() {
    assertEquals("CD", helper.truncateAInFirst2Positions("AACD"));
  }

  @Test
  public void testTruncateAInFirst2Positions_AinFirstPosition() {
    assertEquals("CD", helper.truncateAInFirst2Positions("ACD"));
  }

  @Test
  public void testTruncateAInFirst2Positions_AnotinFirst2Positions() {
    assertEquals("CDAA", helper.truncateAInFirst2Positions("CDAA"));
  }

  @Test
  public void testTruncateAInFirst2Positions_AnotPrestent() {
    assertEquals("CDEF", helper.truncateAInFirst2Positions("CDEF"));
  }

  //ABCD => false, ABAB => true, AB => true, A => false
  @Test
  public void testAreFirstAndLastTwoCharactersTheSame_BasicNegativeScenario() {
    assertFalse("The condition failed", helper.areFirstAndLastTwoCharactersTheSame("ABCD"));
  }

  @Test
  public void testAreFirstAndLastTwoCharactersTheSame_BasicPositiveScenario() {
    assertTrue("The condition failed", helper.areFirstAndLastTwoCharactersTheSame("ABAB"));
  }

  @Test
  public void testAreFirstAndLastTwoCharactersTheSame_TwoChar() {
    assertTrue("The condition failed", helper.areFirstAndLastTwoCharactersTheSame("AB"));
  }

  @Test
  public void testAreFirstAndLastTwoCharactersTheSame_SingleCharOnly() {
    assertFalse("The condition failed", helper.areFirstAndLastTwoCharactersTheSame("A"));
  }
}
```

📌 **SUMMARY:**

Date: @February 20, 2023

Topic: Junit @Before and @After

Recall

Notes

- @Before - Methods annotated with the *@Before* annotation are run before each test.
- @After - Methods annotated with the *@After* annotation are run after each test.

> 💡 Method under @Before and @After should be **public and void**

- @BeforeClass - (Once-only setup) It runs only once for the entire test class before any of the tests are executed, and prior to any @Before method(s).
- @AfterClass - (Once-only tear down) It runs only once after all test case methods and @After annotations have been executed.

> 💡 Method under @BeforeClass and @AfterClass should be **public, void and static.**

📌 **SUMMARY:**

Date: @February 20, 2023

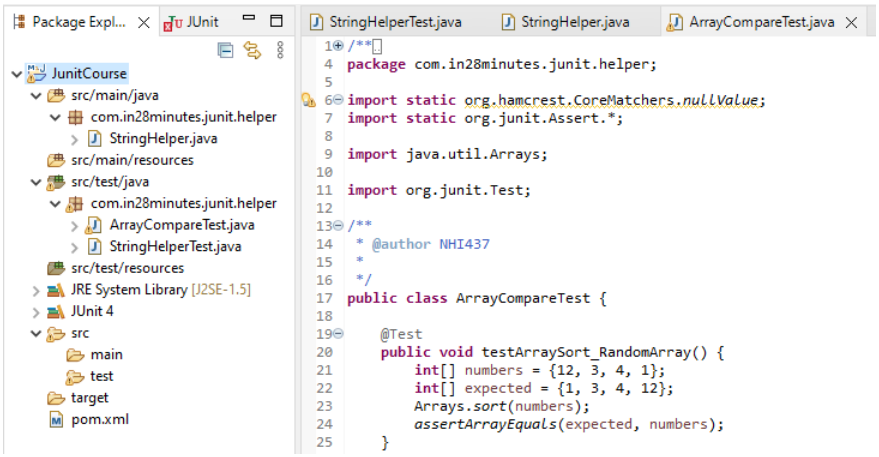
Topic: Junit @AssertArrayEquals And Handling Exception

Recall

Notes

- @AssertArrayEquals(expectedArray, actualArray);
  Check expectedArray is equal to actual or not if it equal than test pass otherwise fail
  Example:

```
@Test
public void testArraySort_RandomArray() {
    int[] numbers = {12, 3, 4, 1};
    int[] expected = {1, 3, 4, 12};
    Arrays.sort(numbers);
    assertArrayEquals(expected, numbers);
}
```



- Handling Exception

1. Using Try-Catch

```java
//Method:1 for Exception
@Test
public void testArraySort_NullArray1() {
    int[] numbers = null;
    try {
        Arrays.sort(numbers);
    } catch (NullPointerException e) {
        // TODO: handle exception
    }
}
```

2. Using @Test(expected = —————)

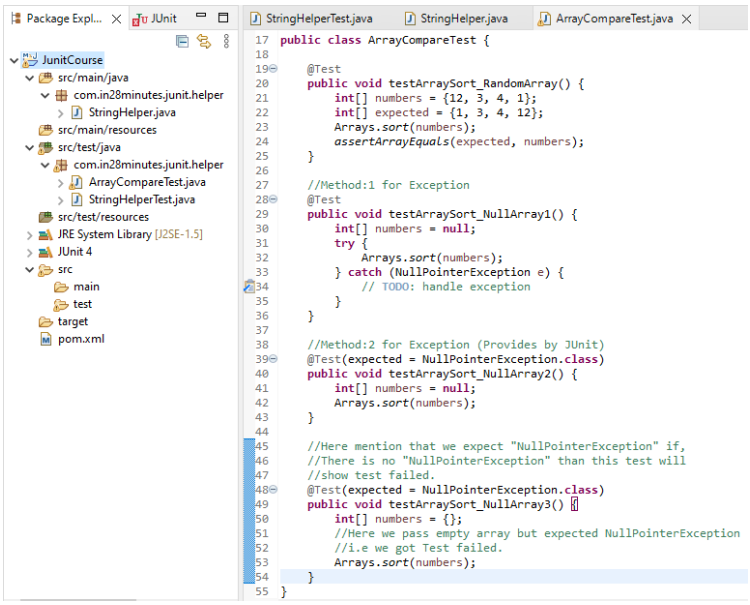
```java
//Method:2 for Exception (Provides by JUnit)
@Test(expected = NullPointerException.class)
public void testArraySort_NullArray2() {
    int[] numbers = null;
    Arrays.sort(numbers);
}
```

> 💡 if we use @Test(expected = ~~Exception1~~) means we expected Excepting will occur and if there is no or other exception will occur than this test will give failed Test.

Example:

```java
//Here mention that we expect "NullPointerException" if,
//There is no "NullPointerException" than this test will
//show test failed.
@Test(expected = NullPointerException.class)
public void testArraySort_NullArray3() {
    int[] numbers = {};
    //Here we pass empty array but expected NullPointerException
    //i.e we got Test failed.
    Arrays.sort(numbers);
}
```

Example:



Code:

```
package com.in28minutes.junit.helper;

import static org.hamcrest.CoreMatchers.nullValue;
import static org.junit.Assert.*;

import java.util.Arrays;

import org.junit.Test;

public class ArrayCompareTest {

  @Test
  public void testArraySort_RandomArray() {
    int[] numbers = {12, 3, 4, 1};
    int[] expected = {1, 3, 4, 12};
    Arrays.sort(numbers);
    assertArrayEquals(expected, numbers);
  }

  //Method:1 for Exception
  @Test
  public void testArraySort_NullArray1() {
    int[] numbers = null;
    try {
      Arrays.sort(numbers);
    } catch (NullPointerException e) {
      // TODO: handle exception
    }
  }

  //Method:2 for Exception (Provides by JUnit)
  @Test(expected = NullPointerException.class)
  public void testArraySort_NullArray2() {
    int[] numbers = null;
    Arrays.sort(numbers);
  }

  //Here mention that we expect "NullPointerException" if,
  //There is no "NullPointerException" than this test will
  //show test failed.
  @Test(expected = NullPointerException.class)
  public void testArraySort_NullArray3() {
    int[] numbers = {};
    //Here we pass empty array but expected NullPointerException
    //i.e we got Test failed.
```

```
    Arrays.sort(numbers);
  }
}
```

📌 **SUMMARY:**

Date: @February 20, 2023

Topic: Performance Test (@Test(timeout = —— ~~milisec~~)

Recall

Notes

Code

```
//Performance Test
@Test(timeout = 100)
public void testSort_performance() {
  int array[] = {12, 23, 4};
  for(int i=1; i<=1000000; i++) {
    array[0] = i;
    Arrays.sort(array);
  }
}
```

📌 **SUMMARY:**

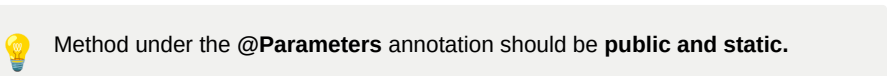
Date: @Today

Topic: Parametrized Test (@RunWith(Parameterized.class)) and Junit Test Suite

Recall

Notes

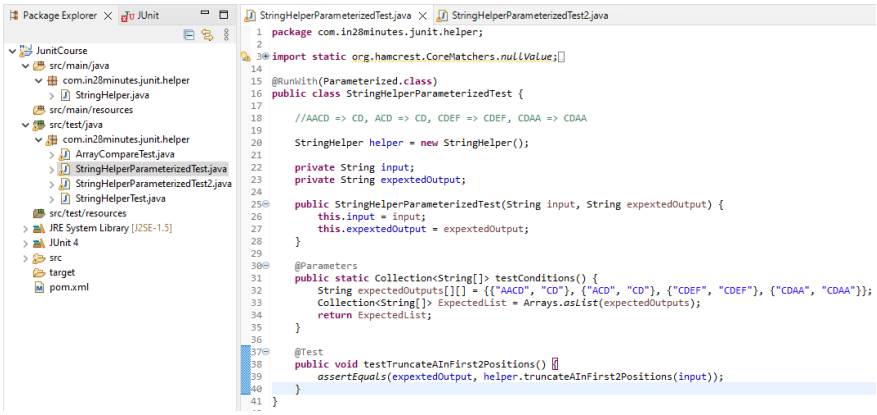
- We can't have two parameterized test in same class we have to make two different class for it. Because we have have two constructor in same class.
  example:
  1. StringHelperParameterizedTest class for - {{"AACD", "CD"}, {"ACD", "CD"}, {"CDEF", "CDEF"}, {"CDAA", "CDAA"}}
  2. StringHelperParameterizedTest2 class for -
- Parameterized class should be under the annotation - **@RunWith(Parameterized.class)**
  Parameter method should be under the annotation - **@Parameters**



Method under the **@Parameters** annotation should be **public and static.**

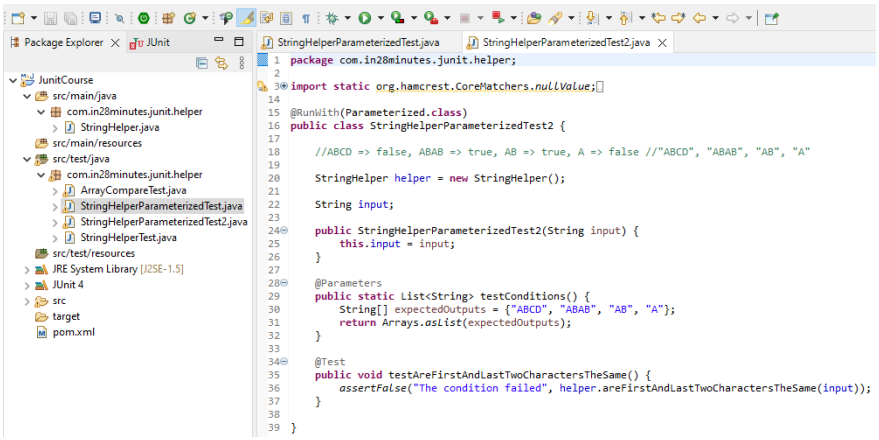
- Example Code

  1. AACD => CD, ACD => CD, CDEF => CDEF, CDAA => CDAA



StringHelperParameterizedTest.java

```
package com.in28minutes.junit.helper;

import static org.hamcrest.CoreMatchers.nullValue;
import static org.junit.Assert.*;

import java.util.Arrays;
import java.util.Collection;
import java.util.List;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class StringHelperParameterizedTest {

  //AACD => CD, ACD => CD, CDEF => CDEF, CDAA => CDAA

  StringHelper helper = new StringHelper();

  private String input;
  private String expextedOutput;

  public StringHelperParameterizedTest(String input, String expextedOutput) {
    this.input = input;
    this.expextedOutput = expextedOutput;
  }

  @Parameters
  public static Collection<String[]> testConditions() {
    String expectedOutputs[][] = {{"AACD", "CD"}, {"ACD", "CD"}, {"CDEF", "CDEF"}, {"CDAA",
    Collection<String[]> ExpectedList = Arrays.asList(expectedOutputs);
    return ExpectedList;
  }

  @Test
  public void testTruncateAInFirst2Positions() {
    assertEquals(expextedOutput, helper.truncateAInFirst2Positions(input));
  }
}
```



StringHelperParameterizedTest2.java

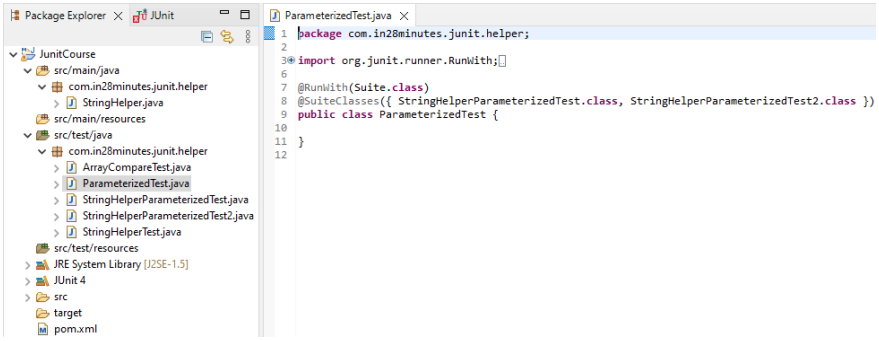
2. ABCD => false, ABAB => true, AB => true, A => false

```
package com.in28minutes.junit.helper;

import static org.hamcrest.CoreMatchers.nullValue;
import static org.junit.Assert.*;

import java.util.Arrays;
import java.util.Collection;
import java.util.List;

import org.junit.Test;
import org.junit.runner.RunWith;
import org.junit.runners.Parameterized;
import org.junit.runners.Parameterized.Parameters;

@RunWith(Parameterized.class)
public class StringHelperParameterizedTest2 {

  //ABCD => false, ABAB => true, AB => true, A => false

  StringHelper helper = new StringHelper();

  String input;

  public StringHelperParameterizedTest2(String input) {
    this.input = input;
  }

  @Parameters
  public static List<String> testConditions() {
    String[] expectedOutputs = {"ABCD", "ABAB", "AB", "A"};
    return Arrays.asList(expectedOutputs);
  }

  @Test
  public void testAreFirstAndLastTwoCharactersTheSame() {
    assertFalse("The condition failed", helper.areFirstAndLastTwoCharactersTheSame(input));
  }

}
```

- **Junit Test Suite** - To Combine the unit test as one and run this file will run all the class that are in that test suite.

  How to create Junit test suite?

Click New Project → Search Junit suite → select the test classes that you want to include in test suite → Name it and click on create.
example:
@SuiteClasses({ StringHelperParameterizedTest.**class**, StringHelperParameterizedTest2.**class, ….**add all the class that you want to combine…. })



📌 **SUMMARY:**