Lab 2. Understanding Asserts

Challenge

1. Explain what the purpose of each method added in the step 1 is
   `whenAssertingEquality_thenEqual(): Tests if strings are equal`
   `whenAssertingArraysEquality_thenEqual(): Tests if arrays are equal`
   `givenNullArrays_whenAssertingArraysEquality_thenEqual(): Tests if arrays are equal with empty arrays`
   `whenAssertingNull_thenTrue(): Tests if value is Null`
   `whenAssertingNotSameObject_thenDifferent(): Tests if two variables don't refer to the same object.`
   `whenAssertingConditions_thenVerified(): Tests Boolean conditions`

2. Modify the method `whenAssertingEquality_thenEqual` changing the actual string to "Not today", run and see the results, then update the method to use the commented assert instead of assertEquals. What is the difference of both asserts?
   Test fails now, AssertEquals with a string displays a message.

3. Create `whenAssertingNull_thenFalse` method based on `whenAssertingNull_thenTrue` and make test pass

```
@Test
public void whenAssertingNull_thenFalse() {
	Object longclaw = new Object();
	assertNotNull(longclaw, "The longclaw should not be null");
	}
```

4. Create `whenAssertingSameObject_thenSame` method based on `whenAssertingNotSameObject_thenDifferent` and make test pass

```
@Test
public void whenAssertingSameObject_thenSame() {
	Object oathkeeper = new Object();
	Object widowswall = oathkeeper;
	assertSame(oathkeeper, widowswall);
	}
```

5. Modify the `methodThatShouldThrowException` method to make the `whenCheckingExceptionMessage_thenEqual` fail

```
public void methodThatShouldThrowException() {
	throw new UnsupportedOperationException("Different Message");
	}
```

6. Modify `testAssertThatHasItems` method to make it pass

```
@Test
public void testAssertThatHasItems() {
```

```
            assertThat(Arrays.asList("Harrenhal", "Dragonstone",
"Winterfell", "Riverrun")).contains("Winterfell", "Riverrun");   }
```

7. Modify `testMultiply` method to use the proper assert and to pass the test

```
@Test
public void testMultiply() {
        try {
                OtherClass multiplier = new OtherClass();
            multiplier.multiply(5, 10);
          } catch (IllegalArgumentException e) {
              assertEquals("X should be less than 1000", e.getMessage());
          }
      }
```

8. Modify `testMultiply_ExceptionIsThrown` method to use the proper assert and to pass the test

```
@Test
public void testMultiply_ExceptionIsThrown() {
        try {
                OtherClass multiplier = new OtherClass();
            multiplier.multiply(5000, 10);
          } catch (IllegalArgumentException e) {
              assertEquals("X should be less than 1000", e.getMessage());
          }
      }
```

Git Repo:
https://github.com/GerardoSoftwareQualityAndTesting/Lab-2.git