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

1. 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.

1. 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");

}

1. 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);

}

1. Modify the methodThatShouldThrowException method to make the whenCheckingExceptionMessage\_thenEqual fail

**public** **void** methodThatShouldThrowException() {

**throw** **new** UnsupportedOperationException("Different Message");

}

1. Modify testAssertThatHasItems method to make it pass

@Test

**public** **void** testAssertThatHasItems() {

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

1. 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());

}

}

1. 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>