SOEN 6011 : SOFTWARE ENGINEERING PROCESSES

SUMMER 2022

ETERNITY

PROBLEM - 5

Unit Test cases

Author

Neona Sheetal Pinto

https://www.overleaf.com/project/62ed25babfddf15056d6b5f7

Contents

List of Figures

1 PROBLEM 7 - F6: $B(x, y)$

1.1 Test cases of F6 $B(x, y)$

As per Java coding standards, the JUnit test cases are created and maintained in a separate folder structure to perform the testing process with zero impact on code section as shown in figure 2

1.2 Test Environment

- IntelliJ IDE for Java
- JUnit4 framework in IntelliJ IDE for testing(4.12) shown in figure

1.3 Testing steps

- Build and Compile the F6 function code in IntelliJ.
- On successful compilation run the F6 function.
- Give different inputs according to the requirements(R2)
- Run the JUnit4 for all the test classes to check the output
- Run all the test cases uses the Test suite, making sure all the test cases are successful.
- Verify the output as shown in figure 1

1.4 Test cases

1.4.1 Test case 1:

Function	testBetaFunctionWithPositiveValues()
Input	BetaFunction(3,4) , 0.001)
Expected	0.0166667
Result	Passed
Requirement ID	R1
Comment	Testing the Beta function with 2 positive values using assertEquals and comparing expected and actual value

1.4.2 Test case 2:

Function	testBetaFunctionWithNegativeValues()
Input	BetaFunction(-3.4,-4.5)
Expected	-1
Result	Passed
Requirement ID	R1
Comment	Testing the Beta function with 2 negative values using assertEquals and comparing expected and actual value

1.4.3 Test case 3:

Function	testBetaFunctionWithOneNegativeValues()
Input	BetaFunction(-3.4,4.5)
Expected	-1
Result	Passed
Requirement ID	R1
Comment	Testing the Beta function with one negative values using assertEquals and comparing expected and actual value

1.4.4 Test case 4:

Function	testBetaFunctionWithZeroValues()
Input	BetaFunction(0,0)
Expected	-1
Result	Passed
Requirement ID	R1
Comment	Testing the Beta function with zero values using assertEquals and comparing expected and actual value

1.4.5 Test case 5:

Function	testBetaFunctionWithIntegralFunction()
Input	integral(0, 1, 3.4, 4.5, (x1, p, q) -> (exp.calculateResult(x1, (p - 1))) * (exp.calculateResult((1- x1), (q-1)))), 0.001)
Expected	0.0084110
Result	Passed
Requirement ID	R1
Comment	Testing the Beta function along with Integral function with double values using assertEquals and comparing expected and actual value

1.4.6 Test case 6:

Function	positiveNumberPowerofPositiveNumber()
Input	calculateResult(5, 9)
Expected	Math.pow(5,9)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with positive values using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.7 Test case 7:

Function	zeroPowerofZero()
Input	calculateResult(0, 0)
Expected	Math.pow(0,0)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with zero values using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.8 Test case 8:

Function	zeroPowerofRealNumber()
Input	calculateResult(0, 3)
Expected	Math.pow(0,3)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with zero value as base and real number as power using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.9 Test case 9:

Function	positiveNumberPowerofZero()
Input	calculateResult(7, 0)
Expected	Math.pow(7,0)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with real value as base and zero as the power using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.10 Test case 10:

Function	negativeNumberPowerofZero()
Input	calculateResult(-4, 0)
Expected	Math.pow(-4,0)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with real negative value as base and zero as the power using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.11 Test case 11:

Function	positiveNumberPowerofOne()
Input	calculateResult(7, 1)
Expected	Math.pow(7,1)
Result	Passed
Requirement ID	R3
Comment	Testing the Exponent function used in the Beta function along with real value as base and one as the power using assertEquals and comparing expected and actual value. The expected value is computed using Math function to check the accuracy.

1.4.12 Test case 12:

Function	numericInputCheckTest()
Input	numericInputCheck(12.3, 43) numericInputCheck(0, 0) numericInputCheck(3, 6) numericInputCheck(-43, 89) numericInputCheck(54, -65) numericInputCheck(-54, -56)
Expected	True or false based on the inputs
Result	Passed
Requirement ID	R1 AND R7
Comment	Multiple input checks to ensure that the correct inputs are passed to the Beta function.

1.5 Snapshots of testing for $B(x, y)$

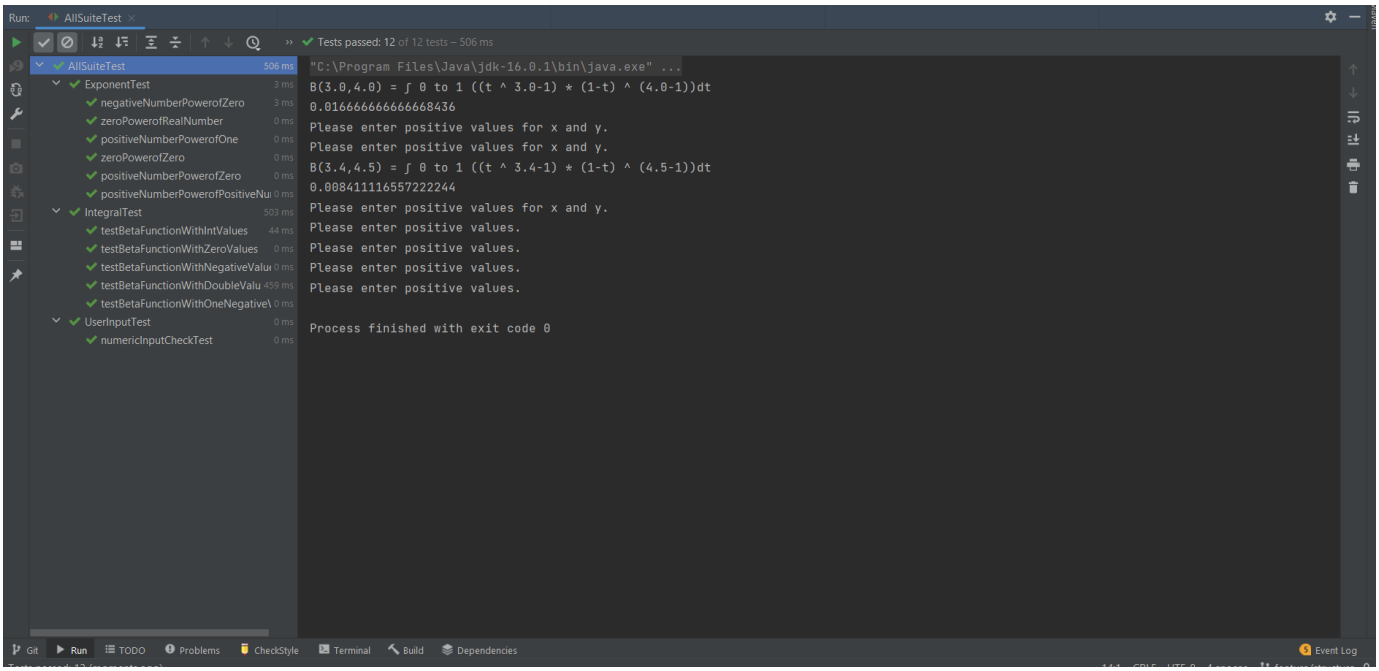

Figure 1: Testing Results Using JUnit-4 All Suite Test for F6 Function

1.6 Annexure:

- **Trello Board :** *https://trello.com/eternity119*
- **Code Version Control :** *https://github.com/neonapinto/Scientific_calculator*
- **Overleaf :** *https://www.overleaf.com/project/62ed25babfddf15056d6b5f7*

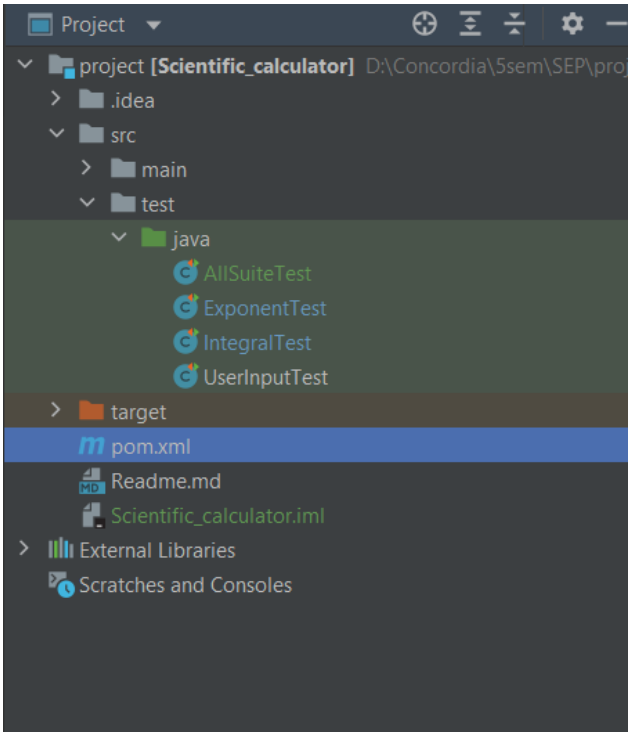

Figure 2: Separation of code from test cases