```
[INFO] -------------------------------------------------------
[INFO]  T E S T S
[INFO] -------------------------------------------------------
[INFO] Running hellocucumber.RunCucumberTest

Scenario Outline: Add two numbers              # hellocucumber/addition.feature:32
  Given A first number 20.5                #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 40                   #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press sum                         # hellocucumber.StepDefinitions.i_press_sum()
  Then the result should be 60.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Add two numbers              # hellocucumber/addition.feature:33
  Given A first number 10.5                #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 0                    #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press sum                         # hellocucumber.StepDefinitions.i_press_sum()
  Then the result should be 10.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Add two numbers              # hellocucumber/addition.feature:34
  Given A first number -2                  #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 4.5                  #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press sum                         # hellocucumber.StepDefinitions.i_press_sum()
  Then the result should be 2.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Add two numbers              # hellocucumber/addition.feature:35
  Given A first number 0                   #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number -4.5                 #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press sum                         # hellocucumber.StepDefinitions.i_press_sum()
  Then the result should be -4.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Add infinite numbers                 # hellocucumber/addition.feature:37
  When I add two infinite numbers              #
hellocucumber.StepDefinitions.i_add_two_infinite_numbers()
  Then test should thrown an ArithmeticException #
hellocucumber.StepDefinitions.test_should_thrown_an_arithmetic_exception()

Scenario Outline: Two numbers divided   # hellocucumber/division.feature:32
```

```
  Given A first number 10                   #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 2                     #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press division                     # hellocucumber.StepDefinitions.i_press_division()
  Then the result should be 5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Two numbers divided        # hellocucumber/division.feature:33
  Given A first number 2000                 #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 100                   #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press division                     # hellocucumber.StepDefinitions.i_press_division()
  Then the result should be 20 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Two numbers divided   # hellocucumber/division.feature:34
  Given A first number 0                    #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 2                     #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press division                     # hellocucumber.StepDefinitions.i_press_division()
  Then the result should be 0 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Number divided by zero             # hellocucumber/division.feature:37
  When I divide a number by zero             #
hellocucumber.StepDefinitions.i_divide_a_number_by_zero()
  Then test should thrown an ArithmeticException #
hellocucumber.StepDefinitions.test_should_thrown_an_arithmetic_exception()

Scenario: Is primo?                          # hellocucumber/esPrimo.feature:31
  Given A integer number 0                          #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press esPrimo                              #
hellocucumber.StepDefinitions.i_press_es_primo()
  Then the result for the int number should be true on screen #
hellocucumber.StepDefinitions.the_result_for_the_int_number_should_be_true_on_screen()

Scenario: Is primo?                          # hellocucumber/esPrimo.feature:32
  Given A integer number 2                          #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press esPrimo                              #
hellocucumber.StepDefinitions.i_press_es_primo()
  Then the result for the int number should be true on screen #
hellocucumber.StepDefinitions.the_result_for_the_int_number_should_be_true_on_screen()
```

```
Scenario: Is primo?                                # hellocucumber/esPrimo.feature:33
  Given A integer number 6                         #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press esPrimo                             #
hellocucumber.StepDefinitions.i_press_es_primo()
  Then the result for the int number should be false on screen #
hellocucumber.StepDefinitions.the_result_for_the_int_number_should_be_false_on_screen(
)

Scenario: Factorial number              # hellocucumber/factorial.feature:32
  Given A integer number 0              #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press factorial                # hellocucumber.StepDefinitions.i_press_factorial()
  Then the result should be 1 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Factorial number              # hellocucumber/factorial.feature:33
  Given A integer number 1              #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press factorial                # hellocucumber.StepDefinitions.i_press_factorial()
  Then the result should be 1 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Factorial number              # hellocucumber/factorial.feature:34
  Given A integer number 5              #
hellocucumber.StepDefinitions.a_integer_number(java.lang.Integer)
  When I press factorial                # hellocucumber.StepDefinitions.i_press_factorial()
  Then the result should be 120 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Factorial of huge numbers                # hellocucumber/factorial.feature:37
  When I try to test the factorial for a number > sixteen #
hellocucumber.StepDefinitions.i_try_to_test_the_factorial_for_a_number_thirteen()
  Then test should thrown an ArithmeticException          #
hellocucumber.StepDefinitions.test_should_thrown_an_arithmetic_exception()

Scenario Outline: Product of two numbers # hellocucumber/product.feature:32
  Given A first number 10               #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 0                 #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press product                  # hellocucumber.StepDefinitions.i_press_product()
  Then the result should be 0 on screen  #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Product of two numbers  # hellocucumber/product.feature:33
  Given A first number 20               #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
```

```
  And A second number 5                  #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press product                   # hellocucumber.StepDefinitions.i_press_product()
  Then the result should be 100 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Product of two numbers # hellocucumber/product.feature:34
  Given A first number 0                 #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 120                #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press product                   # hellocucumber.StepDefinitions.i_press_product()
  Then the result should be 0 on screen  #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Product of two numbers    # hellocucumber/product.feature:35
  Given A first number 100               #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 100                   #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press product                   # hellocucumber.StepDefinitions.i_press_product()
  Then the result should be 10000 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Product of infinite numbers          # hellocucumber/product.feature:37
  When I multiply two infinite numbers          #
hellocucumber.StepDefinitions.i_multiply_two_infinite_numbers()
  Then test should thrown an ArithmeticException #
hellocucumber.StepDefinitions.test_should_thrown_an_arithmetic_exception()

Scenario Outline: Subtract two numbers   # hellocucumber/subtraction.feature:32
  Given A first number 10                #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 0                  #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press minus                     # hellocucumber.StepDefinitions.i_press_minus()
  Then the result should be 10 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Subtract two numbers       # hellocucumber/subtraction.feature:33
  Given A first number 20                #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 4.5                #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press minus                     # hellocucumber.StepDefinitions.i_press_minus()
  Then the result should be 15.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)
```

```
Scenario Outline: Subtract two numbers       # hellocucumber/subtraction.feature:34
  Given A first number 20.5               #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number -4                  #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press minus                      # hellocucumber.StepDefinitions.i_press_minus()
  Then the result should be 24.5 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario Outline: Subtract two numbers   # hellocucumber/subtraction.feature:35
  Given A first number 20.5               #
hellocucumber.StepDefinitions.a_first_number(java.lang.Double)
  And A second number 10.5                #
hellocucumber.StepDefinitions.a_second_number(java.lang.Double)
  When I press minus                      # hellocucumber.StepDefinitions.i_press_minus()
  Then the result should be 10 on screen #
hellocucumber.StepDefinitions.the_result_should_be_on_screen(java.lang.Double)

Scenario: Subtract infinite numbers              # hellocucumber/subtraction.feature:37
  When I subtract two infinite numbers          #
hellocucumber.StepDefinitions.i_subtract_two_infinite_numbers()  Then test should thrown
an ArithmeticException #
hellocucumber.StepDefinitions.test_should_thrown_an_arithmetic_exception()

[INFO] Tests run: 26, Failures: 0, Errors: 0, Skipped: 0, Time elapsed: 0.431 s - in
hellocucumber.RunCucumberTest
[INFO]
[INFO] Results:
[INFO]
[INFO] Tests run: 26, Failures: 0, Errors: 0, Skipped: 0
[INFO]
[INFO] ------------------------------------------------------------------------
[INFO] BUILD SUCCESS
[INFO] ------------------------------------------------------------------------
[INFO] Total time:  3.025 s
[INFO] Finished at: 2022-03-20T13:39:19+01:00
[INFO] ------------------------------------------------------------------------
```