calculate.feature

```
Feature: Calculate
  Scenario: Add two numbers
    Given the input "2+2"
    When the calculator is run
    Then the output should be "4"

  Scenario: Subtract two numbers
    Given the input "9-4"
    When the calculator is run
    Then the output should be "5"

  Scenario: Multiply two numbers
    Given the input "9*4"
    When the calculator is run
    Then the output should be "36"

  Scenario: Evaluate any expression
    Given the input "(89 * 10)/89 + 1"
    When the calculator is run
    Then the output should be "11"

  Scenario: Evaluate any variable expression
    Given the input "(89 * b)/89 + 1"
    When the variable "b" is set to value 10
    And the calculator is run
    Then the output should be "11"

  Scenario: Evaluate a two variable expression
    Given the input "(a + b) * (a - b) + 2"
    And the variable "a" is set to value 10
    And the variable "b" is set to value 5
    When the calculator is run
    Then the output should be "77"

  Scenario: Evaluate an uninitialized expression
    Given the input "a/b"
    And the variable "a" is set to value 10
    But the variable "b" is not set
    When the calculator is run
```


calculateStepdefs.java

```
package cucumber.examles.java.helloworld;

import static org.junit.Assert.assertEquals;


public class calculateStepdefs {
    private Calculator c;
    private String out;

    @Given("^the input \"([^\"]*)\"$")
    public void the_input(String input) throws Throwable {
        // Express the Regexp above with the code you wish you had
        // throw new PendingException();
        c = new Calculator(input);

    }

    @When("^the variable \"([^\"]*)\" is set to value (\\d+)$")
    public void the_variable_is_set_to_value(String variable, int
value) throws Throwable {
        c.SetVariable(variable, value);

    }

    @When("^the variable \"([^\"]*)\" is not set$")
    public void the_variable_is_not_set(String variable) throws
Throwable {
        // Express the Regexp above with the code you wish you had
        // throw new PendingException();
        c.isVariableSet(variable);
    }


    @When("^the calculator is run$")
    public void the_calculator_is_run() throws Throwable {
        out = c.Run();
    }

    @Then("^the output should be \"([^\"]*)\"$")
```


Calculator.java

```
package cucumber.examles.java.helloworld;

import org.apache.commons.jexl2.*;

public class Calculator {
    private String cString;
    private JexlEngine jexl;
    private Expression expression;
    private JexlContext jexlContext;  // required if expression has
variables

    public Calculator(String input) {
        cString = input;
        jexl = new JexlEngine();
        // to enable exception for Division by zero
        jexl.setStrict(true);
        // jexl.setSilent(true);
        // jexl.setLenient(true);
        expression = jexl.createExpression(cString);
        jexlContext = new MapContext();
    }

    public String Run() {
        if (cString == "Error")
            return cString;

        try {
            Object eval = expression.evaluate(jexlContext);
            return eval.toString();
        } catch (JexlException e) {
            cString = "Overflow";
            return cString;
        }
    }

    public void SetVariable(String variable, int value) {
        // no idea why this doesn't work
        // if (jexlContext.has(variable) == true)
            jexlContext.set(variable, value);
```


calculate.feature

```
  Then the output should be "Error"

  Scenario: Evaluate a division by zero
    Given the input "a / b"
    And the variable "a" is set to value 10
    And the variable "b" is set to value 0
    When the calculator is run
    Then the output should be "Overflow"
```


calculateStepdefs.java

```
    public void the_output_should_be(String output) throws Throwable {
        assertEquals (out, output);
    }

}
```


Calculator.java

```
    }

    public void isVariableSet(String variable) {
        Object val = jexlContext.get(variable);
        try {
            val.toString();
        }
        // must rewrite to use JexclException
        catch (NullPointerException e) {
            cString = "Error";
        }
    }
}
```