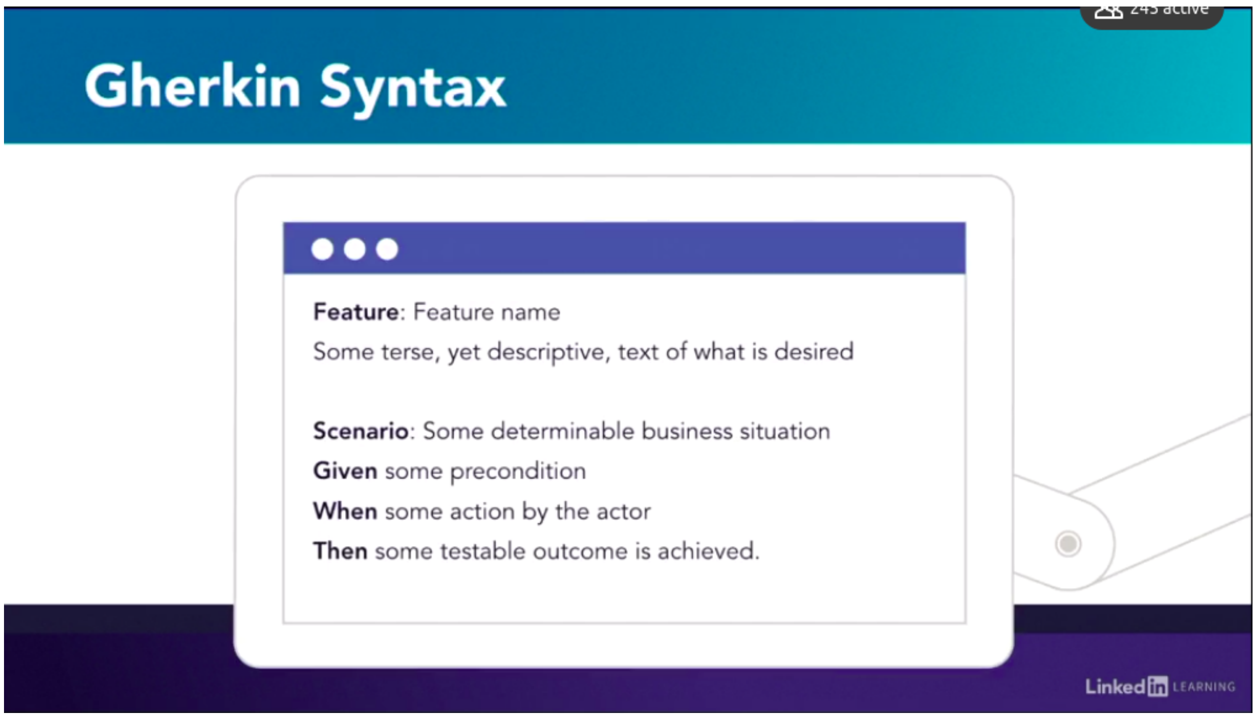


Gherkin Syntax

Feature: Feature name
Some terse, yet descriptive, text of what is desired

Scenario: Some determinable business situation
Given some precondition
When some action by the actor
Then some testable outcome is achieved.

```
sudo mkdir projectbdd
cd projectbdd/
code .
```

Create Gemfile that is the dependences of the project

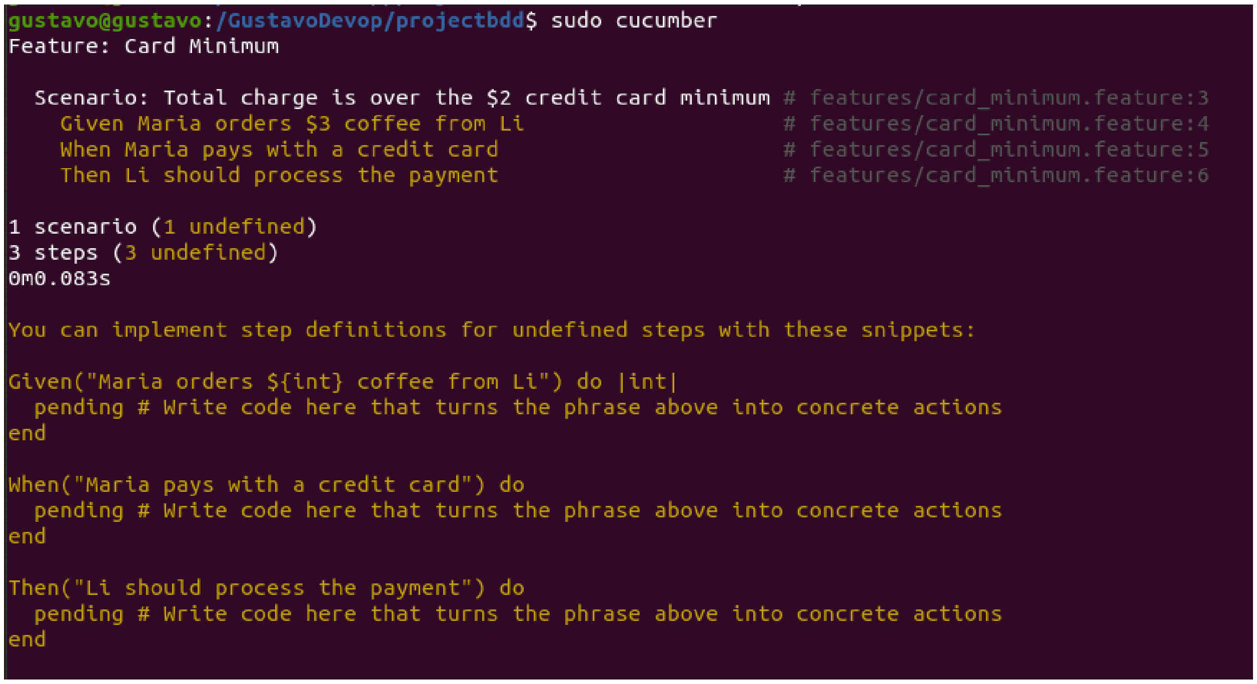
```
#we will define the dependences
source 'https://rubygems.org'

gem 'cucumber', '~> 3.1.0'
gem 'rspec', '~> 3.7.0'
```

```
sudo bundle install
sudo cucumber --init

sudo chmod -R 777 ./
```

Create in the features directory the file card_minimum.feature and write the scenario

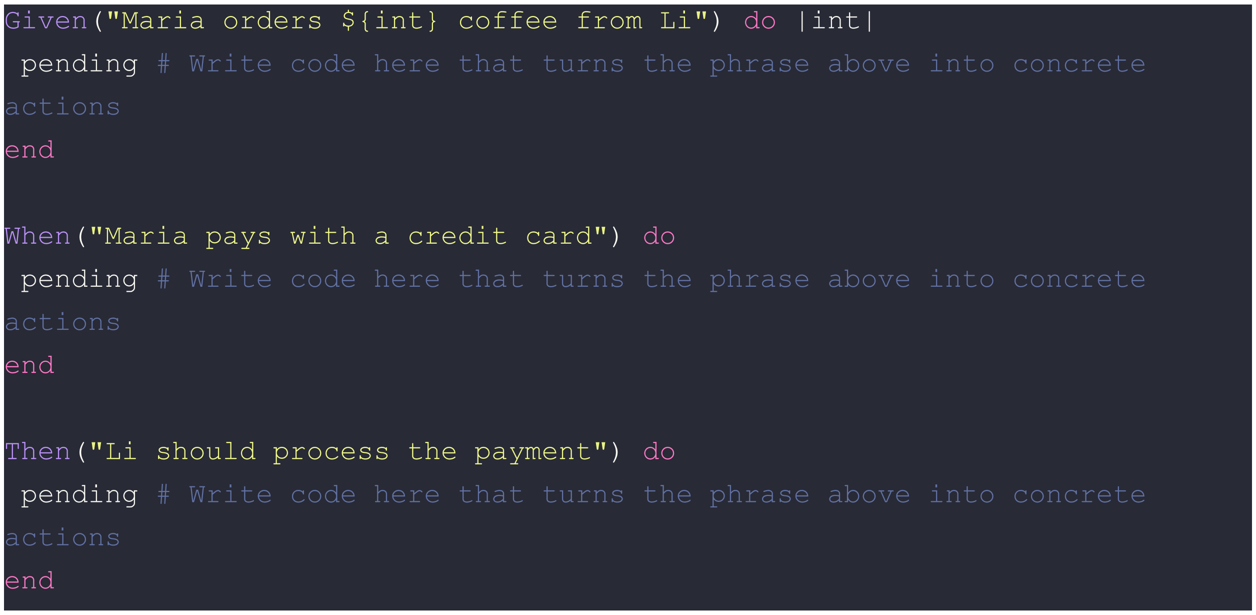
```
Feature: Card Minimum

  Scenario: Total charge is over the $2 credit card minimum
  Given Maria orders $3 coffee from Li
  When Maria pays with a credit card
  Then Li should process the payment
```

sudo cucumber # to show the scenario test as code



```
gustavo@gustavo:/GustavoDevop/projectbdd$ sudo cucumber
Feature: Card Minimum

  Scenario: Total charge is over the $2 credit card minimum # features/card_minimum.feature:3
    Given Maria orders $3 coffee from Li                    # features/card_minimum.feature:4
    When Maria pays with a credit card                      # features/card_minimum.feature:5
    Then Li should process the payment                      # features/card_minimum.feature:6

1 scenario (1 undefined)
3 steps (3 undefined)
0m0.083s

You can implement step definitions for undefined steps with these snippets:

Given("Maria orders ${int} coffee from Li") do |int|
  pending # Write code here that turns the phrase above into concrete actions
end

When("Maria pays with a credit card") do
  pending # Write code here that turns the phrase above into concrete actions
end

Then("Li should process the payment") do
  pending # Write code here that turns the phrase above into concrete actions
end
```

Create the steps.rb file in the steps directory to paste the code of the test



```
Given("Maria orders ${int} coffee from Li") do |int|
 pending # Write code here that turns the phrase above into concrete
actions
end

When("Maria pays with a credit card") do
 pending # Write code here that turns the phrase above into concrete
actions
end

Then("Li should process the payment") do
 pending # Write code here that turns the phrase above into concrete
actions
end
```

sudo cucumber # will match the steps with the scenario to run the bdd test

# We will code the test first

```
require 'ourcode'
Given("Maria orders ${int} coffee from Li") do |int|
@maria = Customer.new
@li = Associate.new
price = int
@maria.orders
end
When("Maria pays with a credit card") do
@maria.pays_with_card
end
Then("Li should process the payment") do
expect(@li.process_payment).to include(true)
end
```

# Then we code to the test to work

```
class Customer
   def orders
   end

   def pays_with_card
   end
end

class Associate
   def process_payment
   end
end
```

EMBRACE FAILURE!

# Require our code in the test script and then run the test again to pass
# Add the @ sign before the variables in ruby makes it visible in all the scopes