

Não Programe ao Acaso

Você está entregando o que foi solicitado?

- Um programa é feito para alguém, um cliente.
- O cliente é quem define se aceita a entrega.

Dificuldades na entrega

- Contradições
- Informações omitidas
- Mudança de idéia
- Divergência do solicitado
- Alternativas não percebidas

Especificação Funcional

- Determina o que o sistema faz
- Pode ser discutida com o cliente
- Ajuda a perceber dúvidas sobre o que deve ser feito
- Pode apresentar exemplos de como o sistema deve funcionar

Escrevendo a especificação

- Dado ...
 - Um estado
 - Uma entrada
- Quando ...
 - Uma ação ou evento ocorrem
- Então ...
 - Uma saída esperada

Repositório de Código

<http://github.com/gherkin-by-example>

- Repositórios com exemplos em diferentes linguagens de programação.
- Esta apresentação.



GitHub

Enunciado do Problema

Leia 2 valores inteiros e armazene-os nas variáveis A e B. Efetue a **soma de A e B** atribuindo o seu resultado à variável X. Imprima X conforme exemplo. Caso a saída seja diferente do que está especificado você receberá "Presentation Error". Não esqueça de imprimir o fim de linha.

Exemplos:

Entrada: 3 4 Saída: X = 7

Entrada: -2 6 Saída: X = 4

Funcionalidade e Narrativa

21 **Feature:** Calculator

22

23 **Narrative:**

24

25 In order to avoid silly mistakes

26 As a math novice

27 I want to be told the sum of two numbers

Cenário

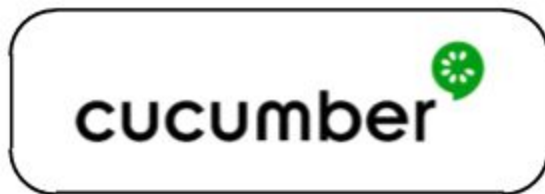
```
29 Scenario: Run program with 10 and 9 (pos,pos)
30
31 Given the input is
32     """
33     10
34     9
35     """
36 When the program runs
37 Then the output should be
38     """
39     X = 19
40
41     """
```

Java

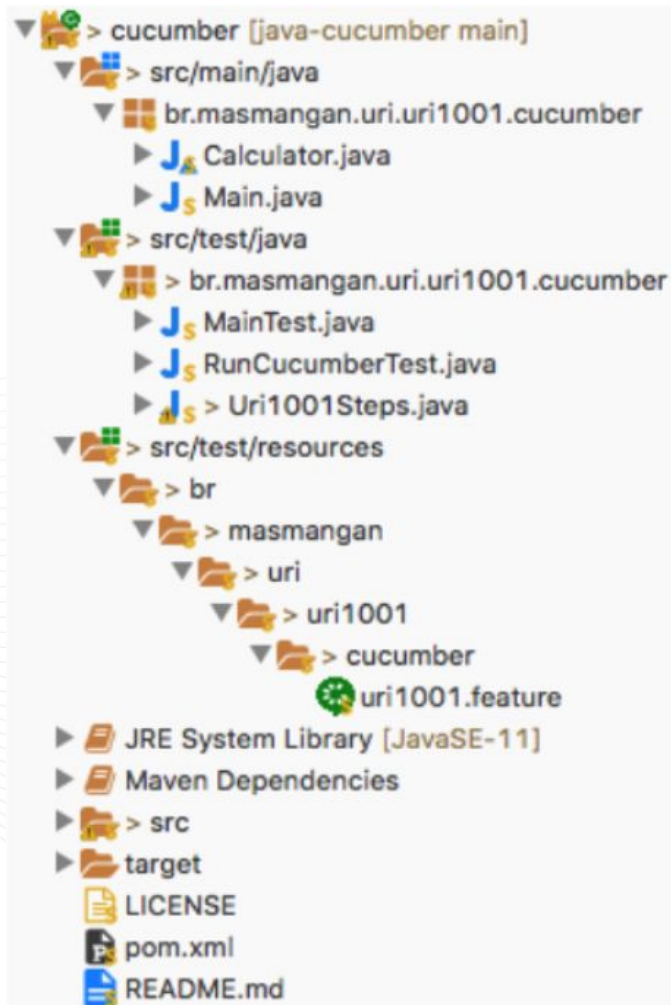
Exemplo de Solução em Java

```
1  import java.io.BufferedReader;
2  import java.io.IOException;
3  import java.io.InputStreamReader;
4
5  public class Main {
6      public static void main(String[] args) throws IOException {
7          InputStreamReader ir = new InputStreamReader(System.in);
8          BufferedReader in = new BufferedReader(ir);
9          int A, B, X;
10         A = Integer.parseInt(in.readLine());
11         B = Integer.parseInt(in.readLine());
12         X = A + B;
13         System.out.printf("X = %d\n", X);
14     }
15 }
```

Dependências



Estrutura de diretórios com Maven e Cucumber



pom.xml (fragmento)

```
<dependencies>
  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-java</artifactId>
    <version>${cucumber.version}</version>
    <scope>test</scope>
  </dependency>

  <dependency>
    <groupId>io.cucumber</groupId>
    <artifactId>cucumber-junit</artifactId>
    <version>${cucumber.version}</version>
    <scope>test</scope>
  </dependency>
```

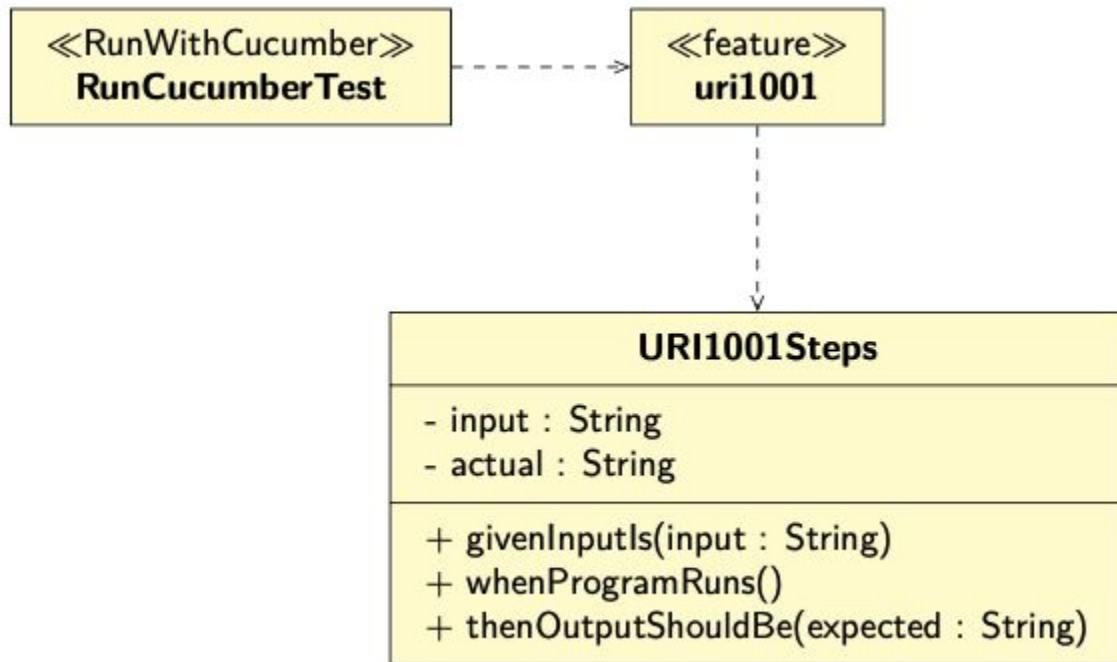

Código Inicial

You can implement missing steps with the snippets below:

```
@Given("the input is")
public void the_input_is(String docString) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@When("the program runs")
public void the_program_runs() {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}

@Then("the output should be")
public void the_output_should_be(String docString) {
    // Write code here that turns the phrase above into concrete actions
    throw new io.cucumber.java.PendingException();
}
```



RunCucumberTest.java

```
27  @RunWith(Cucumber.class)
28  @CucumberOptions(plugin = {"pretty"})
29  public class RunCucumberTest {
30
31  }
```

Uri1001Steps.java (1/3)

```
36  public class Uri1001Steps {  
37  
38      private String input;  
39      private String actual;  
40  
41      @Given("the input is")  
42      public void the_input_is(String input) {  
43          this.input = input;  
44      }
```

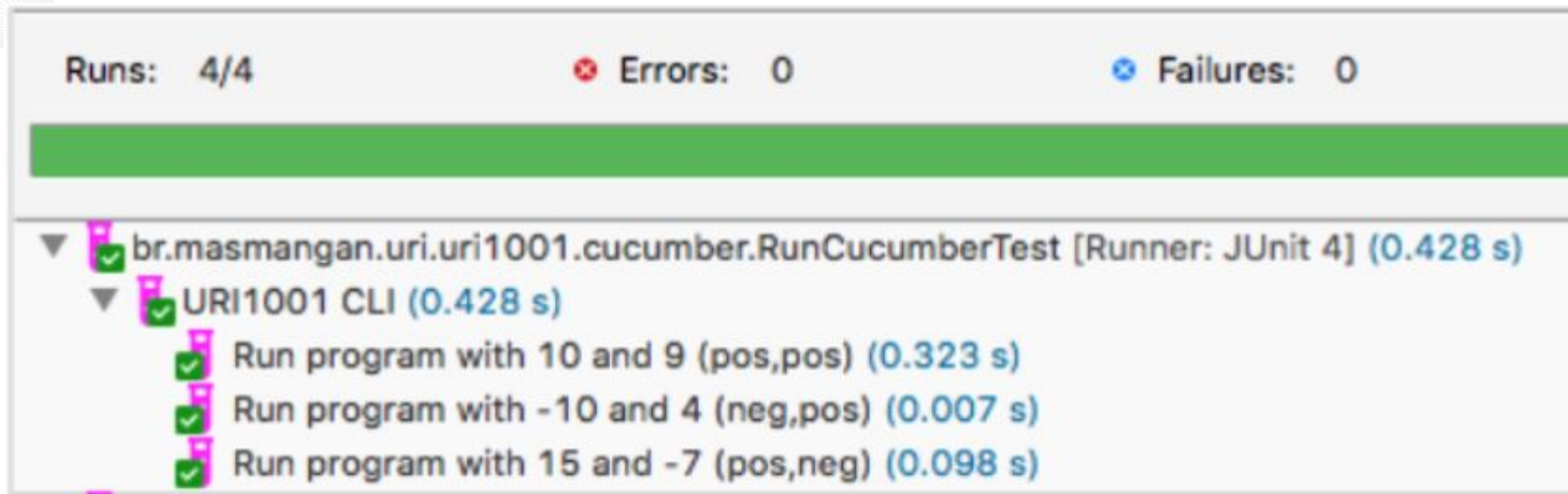
Uri1001Steps.java (2/3)

```
46      @When("the program runs")
47      public void the_program_runs() throws IOException {
48          var inputStream = new ByteArrayInputStream(
49              input.getBytes(Charset.forName("UTF-8")));
50          var baos = new ByteArrayOutputStream();
51          var outputStream = new PrintStream(baos);
52          System.setIn(inputStream);
53          System.setOut(outputStream);
54          Main.main(null);
55          actual = baos.toString();
56          inputStream.close();
57          outputStream.close();
58      }
```

Uri001Steps.java (3/3)

```
58         @Then("the output should be")
59         public void the_output_should_be(String expected) {
60             assertEquals(expected, actual);
61         }
62     }
```

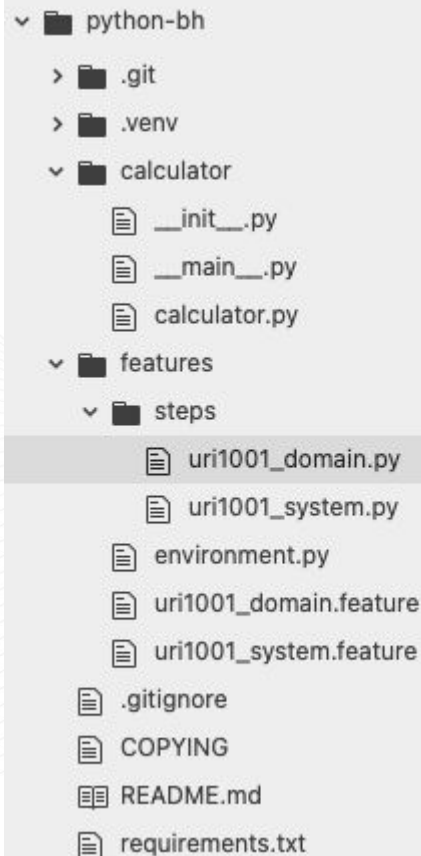
Resultado da Execução



Python

A configuração Python

- Configuração do repositório
- Configuração do projeto
- Estrutura de diretórios para o **behave**
 - features
 - Especificações
 - environment.py
 - steps
 - Implementação dos testes



```
python-bh
├── .git
├── .venv
├── calculator
│   ├── __init__.py
│   ├── __main__.py
│   └── calculator.py
├── features
│   └── steps
│       ├── uri1001_domain.py
│       ├── uri1001_system.py
│       ├── environment.py
│       ├── uri1001_domain.feature
│       └── uri1001_system.feature
├── .gitignore
├── COPYING
├── README.md
└── requirements.txt
```

Configuração dos Testes

```
def before_scenario(context, scenario):  
    tags = set(scenario.tags + scenario.feature.tags)  
    if "domain" in tags:  
        context.calculator = Calculator()
```


Steps para validação do Domínio

```
@given("the first number is {number:d}")
def _given_first_number(context, number):
    context.calculator.add_input(number)

@given("the second number is {number:d}")
def _given_second_number(context, number):
    context.calculator.add_input(number)

@when("the two numbers are added")
def _when_two_numbers_are_added(context):
    context.result = context.calculator.sum()

@then("the result should be {result:d}")
def _then_result_should_be(context, result):
    assert context.result == result
```

Implementação do Domínio

```
class Calculator:  
    """A helper for math novices."""  
  
    def __init__(self):  
        """Initialize calculator object."""  
        self.numbers = []  
  
    def add_input(self, value):  
        """Add a new input to the calculator."""  
        self.numbers.append(value)  
  
    def sum(self):  
        """Sum two numbers."""  
        return self.numbers.pop() + self.numbers.pop()
```

Steps para validação do Sistema (*Dado que...*)

```
@given("the input is")  
def _given_input(context):  
    context.stdin_data = context.text
```

Steps para validação do Sistema (*Quando...*)

```
@when("the program runs")
def _when_program_runs(context):
    context.result = subprocess.run(
        ["python", "-m", "calculator"],
        capture_output=True,
        check=True,
        input=context.stdin_data.encode("utf-8"),
    )
```

Steps para validação do Sistema (*Então...*)

```
@then("the output should be")
def _then_output_should_be(context):
    assert context.text == context.result.stdout.decode("utf-8")
```

Implementação do Sistema

```
def main():  
    """Program entry point."""  
    calculator = Calculator()  
    calculator.add_input(int(input()))  
    calculator.add_input(int(input()))  
    print(f"X = {calculator.sum()}")  
  
if __name__ == "__main__":  
    main()
```


Execução dos Testes

Scenario: Run program with 15 and -7 (pos,neg)

Given the input is

"""

15

-7

"""

When the program runs

Then the output should be

Then the output should be

""" 8

X = 8

"""

"""

2 features passed, 0 failed, 0 skipped

6 scenarios passed, 0 failed, 0 skipped

21 steps passed, 0 failed, 0 skipped, 0 undefined

Took 0m0.093s

(.venv) rafaell@kessel python-bh %



Muito obrigado!

Referências

Repositórios de Exemplos

<http://github.com/gherkin-by-example>

URI Online Judge

<https://www.urionlinejudge.com.br>

SpecFlow

<http://specflow.org>