**Trabalho Prático 2 - esii - Grupo1213**

Test design Specification

Version 7.0

21/01/2022

**Histórico de Versões**

| **Version #** | **Implemented**  **By** | **Revision**  **Date** | **Approved**  **By** | **Approval**  **Date** | **Reason** |
| --- | --- | --- | --- | --- | --- |
| 1.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 12/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 12/01/2022 | Análise testes caixa preta para o módulo transações |
| 2.0 | Bruno Ferreira  Gonçalo Oliveira | 13/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 13/01/2022 | Correção de pormenores nas tabelas de casos de teste |
| 3.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 14/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 14/01/2022 | Correção de alguns erros identificados |
| 4.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 15/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 15/01/2022 | Adicionar casos de testes identificados na análise testes de caixa branca do módulo de transações |
| 5.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 17/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 17/01/2022 | Análise testes caixa preta para o módulo de custos de envio |
| 6.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 18/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 18/01/2022 | Adicionar casos de testes identificados na análise testes de caixa branca do módulo de cálculo de custos |
| 7.0 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 21/01/2022 | Bruno Ferreira  Jorge Correia  Gonçalo Oliveira  Nuno Castro | 21/01/2022 | Análise testes caixa preta para o módulo de expedição |

**Tabela de Conteúdos**

[**1. Introdução**](#_toqkrpxxq053) **4**

[1.1. Identificador do documento](#_30j0zll) 4

[1.2. Âmbito](#_17dp8vu) 4

[1.3. Repositório](#_3rdcrjn) 4

[1.4. Referências](#_26in1rg) 4

[1.5. Glossário](#_lnxbz9) 5

[**2. Features/Itens a testar**](#_hmex55ht29be) **5**

[**3. Detalhes da abordagem aos testes**](#_u58uqu53waiz) **7**

[**4. Identificação dos Testes**](#_emuwx11191ck) **7**

[1.GestãoEncomendas](#_oyt1j4uf2j0r) 8

[1.1 addEncomenda](#_nn7f1s7kercb) 8

[1.2 cancelarEncomenda](#_ilyi6wwfj69d) 9

[1.3 findEncomenda](#_49c0d2eax18p) 10

[2.BasicLinhaTransação](#_q85ipb6bfsh) 11

[2.1 addTransactionLine](#_5ru7r4sb1qnq) 11

[3.Estatísticas](#_wtiobotraay) 12

[3.1 getValorMedioTransacoes](#_1m7b9fle1znb) 12

[3.2 getNumMedioProdutosPorTransacao](#_d9hqz2c3kv8f) 13

[3.3 getNumMedioEncomendasPorDia](#_eyttob3p9nl) 14

[3.4 getTotalCustosEnvio](#_nfixybzgiicw) 15

[3.5 getValorMedioVendasComprasPorDistrito](#_ft3f7nn7ryon) 16

[4.Custos](#_e3cic7lhwi5i) 17

[4.1 calculaCustoEnvio](#_cj44ec22hwdb) 17

[5.Expedição](#_6dnoh8ixmsl1) 18

[5.1 assignEncomendas](#_qbz4kt6oul38) 18

[**5. Critérios de passagem ou falha das features**](#_fcrko0th2bzr) **19**

# **1. Introdução**

## 1.1. Identificador do documento

TestCaseSpecification\_Grupo1213

## 1.2. Âmbito

Este relatório de Test Case Specification está a ser realizado no âmbito do Trabalho Prático 2 da disciplina de Engenharia de Software II em que, e tal como o nome indica, o objetivo deste relatório é especificar os testes BlackBoxTests bem como, toda a documentação referente a abordagens efetuadas e resultados esperados.

Ao longo do desenvolvimento do módulo foram feitos testes unitários aos métodos que, na nossa opinião, se identificavam como cruciais para o bom funcionamento da aplicação.

Estão explicitados neste relatório todas as informações necessárias para a realização dos testes, bem como as ferramentas e técnicas usadas para tal.

## 1.3. Repositório

A hiperligação para o repositório onde foi realizado o trabalho é a seguinte:

<https://gitlab.estg.ipp.pt/esii.grupo1213/esii-grupo1213-tp2>

## 1.4. Referências

Como referência para a realização deste documento foram usados documentos como:

* Slides disponibilizados no Moodle da Unidade Curricular
* IEEE Standard for Software Unit Testing

## 1.5. Glossário

**Testes de software** - Um teste de software é um software que executa outro software, validando se os resultados são os esperados (teste de estado) ou se executa a sequência de eventos esperado (teste de comportamento).

Resumidamente, os testes de software permitem aos programadores verificar se a lógica do programa desenvolvido está de acordo com os requisitos.

A execução automática de testes permite identificar “bugs” resultantes de mudanças no código fonte.

**ECP** - Técnica destinada a reduzir o número de testes necessários dividindo o domínio de entrada (ou saída) em classes de dados em que os casos de teste podem ser derivados para cada operação, o “tester” deve identificar as classes de equivalência dos argumentos e os estados dos objetos

**BVA** - Técnica baseada na observação de bugs que ocorrem frequentemente em valores fronteira. É focada em testar valores especiais (null, 0, etc) e limites do domínio de entrada (ou saída) imediatamente acima e abaixo (além de ou em vez de valores intermédios)

**JUnit** - Framework open-source para realizar testes unitários para código em Java.

Permite a execução de testes de forma:

* Fácil
* Regular
* Fiável

Contém várias funcionalidades para testing entre elas a capacidade de testar cada componente de um programa de forma independente do resto do programa

**Gradle** - É uma ferramenta que permite integrar e automatizar várias tarefas relacionadas com o processo de desenvolvimento de software em várias linguagens de programação.

O Gradle determina quais os componentes do projeto que estão atualizados, evitando a recompilação de todo o projeto.

# **2. Features/Itens a testar**

| **Item a testar** | **Descrição** | **Requisitos** | **Responsabilidade** |
| --- | --- | --- | --- |
| addEncomenda | Adicionar uma encomenda ao sistema que, ainda não se encontre no mesmo, para que esta possa ser gerida devidamente. | --------------------- | Bruno Ferreira |
| cancelarEncomenda | Cancelar uma encomenda que esteja adicionada e ainda não tenha sido feito o pagamento. | --------------------- | Jorge Correia |
| findEncomenda | Verifica se uma determinada encomenda se encontra registada. | --------------------- | Gonçalo Oliveira |
| addTransactionLine | Adicionar linhas de transação a uma transação. Uma linha de transação contém informação sobre o produto e sobre a própria linha de transação. | --------------------- | Nuno Castro |
| getValorMedioTransacoes | Métrica para calcular o número médio de transações. | --------------------- | Bruno Ferreira |
| getNumMedioProdutosPorTransacao | Métrica para obter o número médio de produtos que uma transação possui. | --------------------- | Bruno Ferreira |
| getNumMedioEncomendasPorDia | Métrica para obter o número médio de encomendas por dia | --------------------- | Gonçalo Oliveira |
| getTotalCustosEnvio | Métrica para obter o total custo de envio gasto num determinado tempo. | --------------------- | Gonçalo Oliveira |
| calculaCustoEnvio | Calcular o custo de envio de uma determinada encomenda para que posteriormente seja calculado o custo total da encomenda | --------------------- | Jorge Correia |
| assignEncomendas | Atribuir encomendas a contentores e posteriormente exportar documento JSON que contém a informação dos contentores e das suas encomendas | --------------------- | Bruno Ferreira |
| getValorMedioVendasComprasPorDistrito | Métrica para calcular o valor médio de vendas e compras por distrito.. | --------------------- | Bruno Ferreira |

## 

# **3. Detalhes da abordagem aos testes**

Para assegurar a qualidade do software foram realizados testes de forma a tornar o software robusto e resistente a falhas. Tendo isto em conta foram utilizadas técnicas como ECP (equivalency class partitioning) e BVA (boundary value analysis).

Inicialmente foram identificados os métodos mais importantes de todo o projeto e foram realizadas tabelas tendo em conta as técnicas identificadas acima, em seguida passou-se para a codificação dos testes utilizando a framework JUnit.

No caso da técnica ECP foram realizadas tabelas identificando classes de equivalência válidas e inválidas para cada método e foram realizados casos de teste baseados nessas classes, no caso do BVA foram identificadas entradas em que as pré-condições são asseguradas bem como o contrário.

Para finalizar será criada uma tabela com casos de teste para cada método, identificando o ECP em que se insere, os casos limite (BVA), o resultado esperado, o resultado obtido e o resultado do teste.

Pretende-se com esta abordagem detetar o maior número de erros executando o menor número de casos de teste.

# 

# 

# 

# 

# 

# 

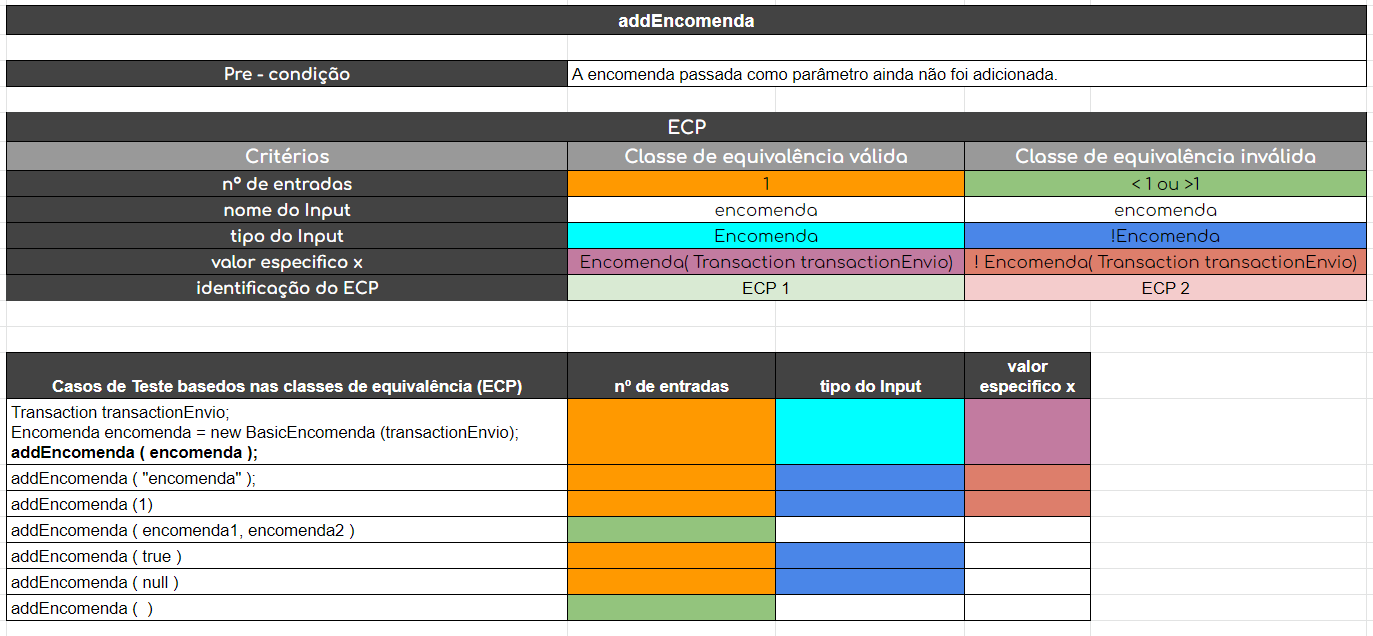
# 

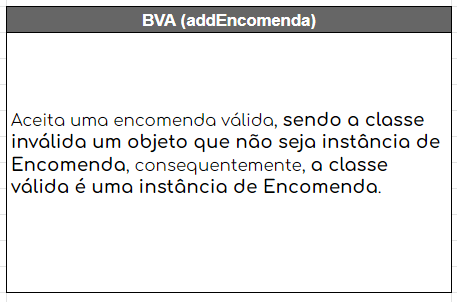
# **4. Identificação dos Testes**

Neste tópico estão identificadas todas as tabelas ECP e BVA para cada método, realizadas pelo grupo.

## 1.GestãoEncomendas

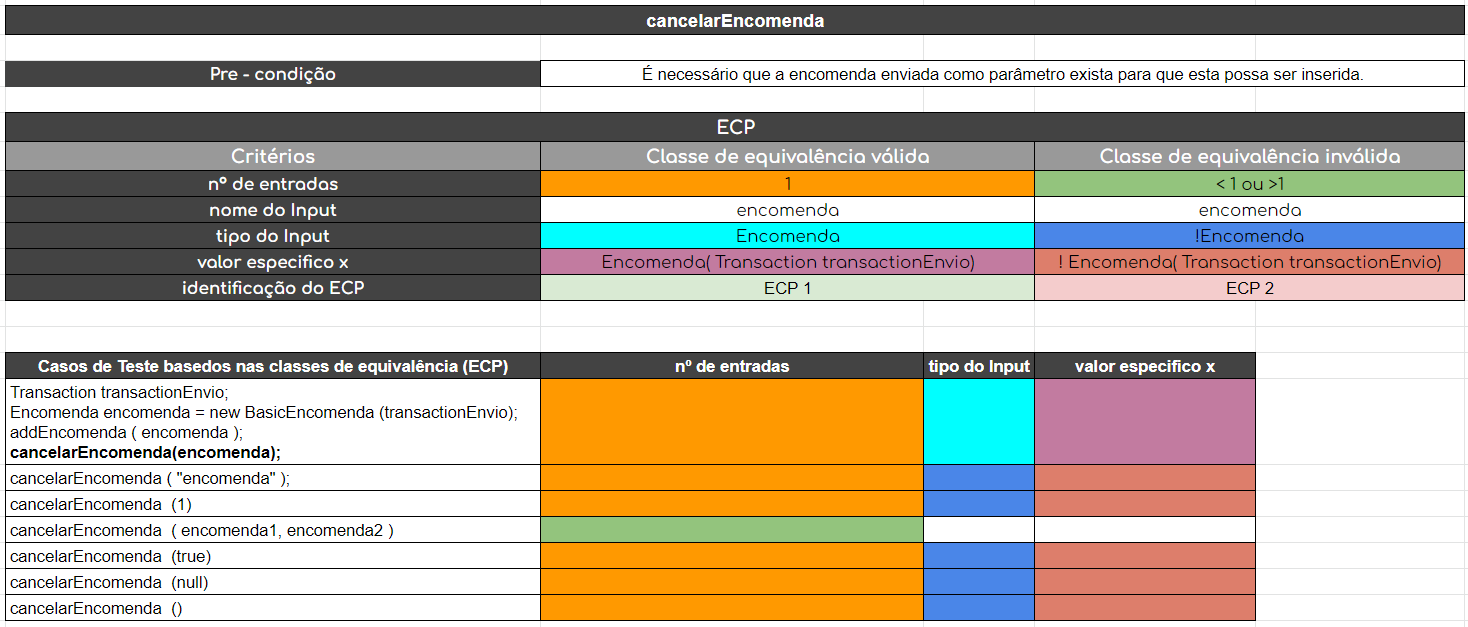
### 1.1 addEncomenda

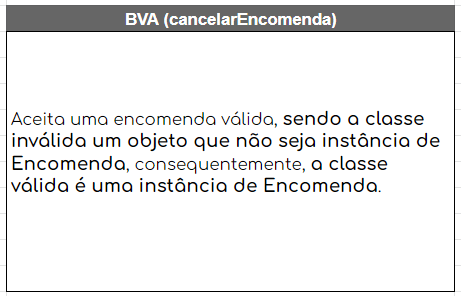


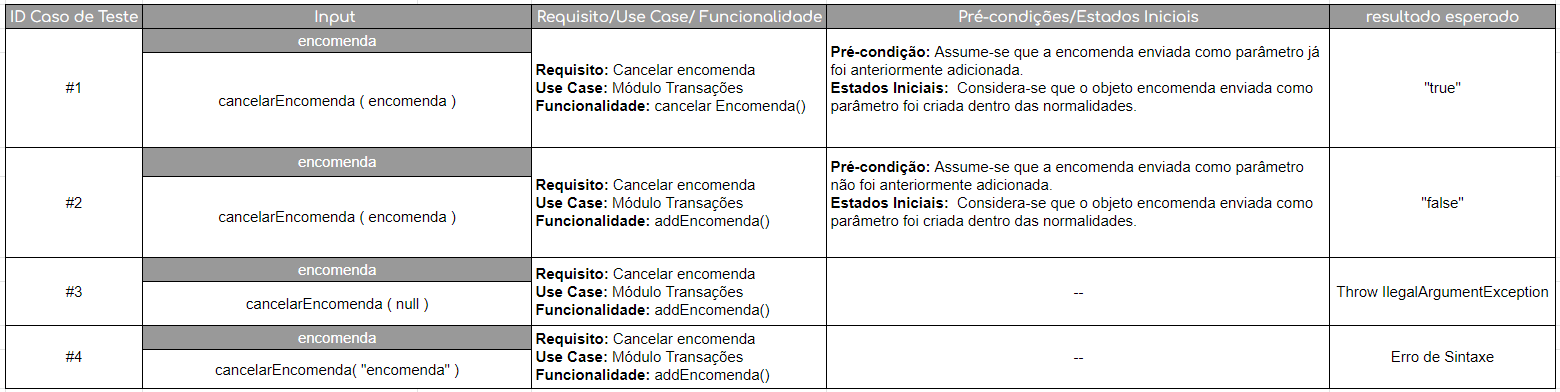


### 

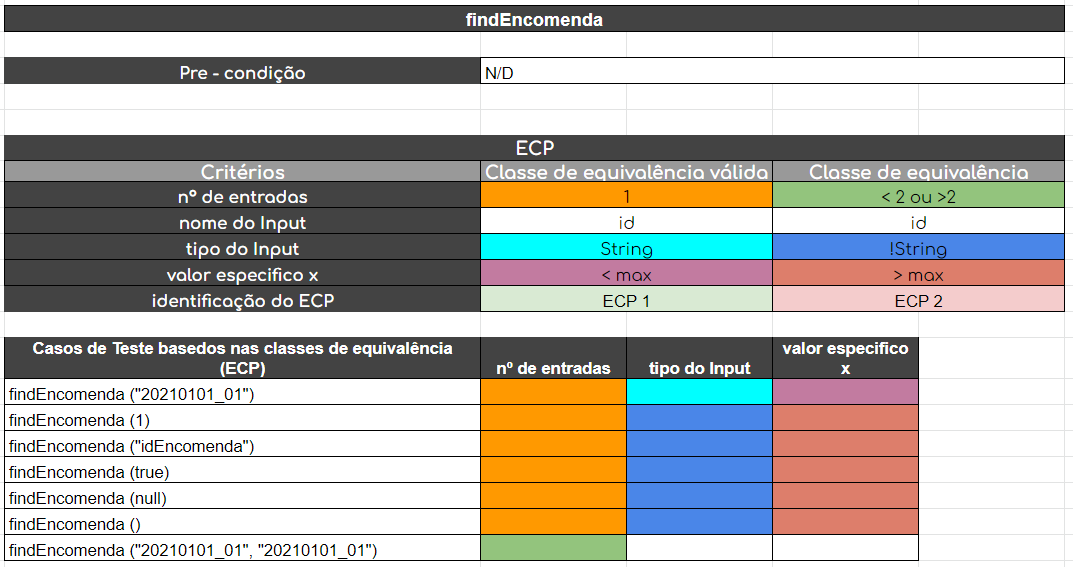
### 1.2 cancelarEncomenda

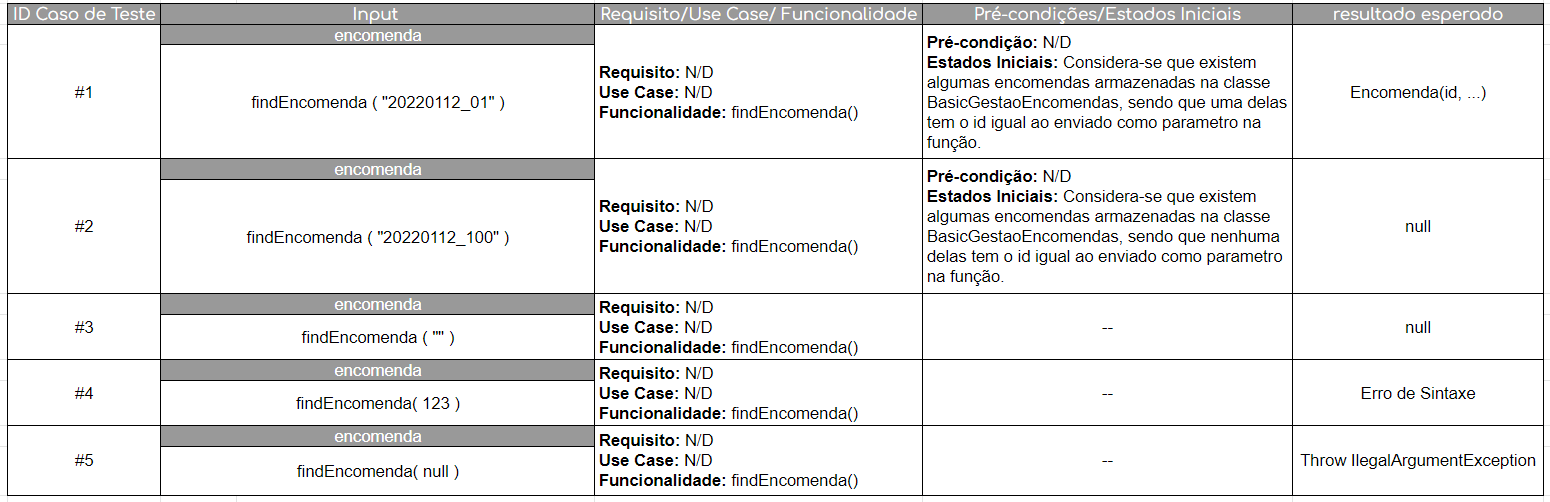
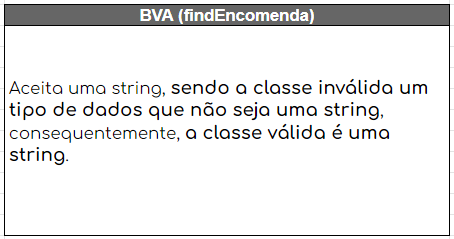






### 1.3 findEncomenda

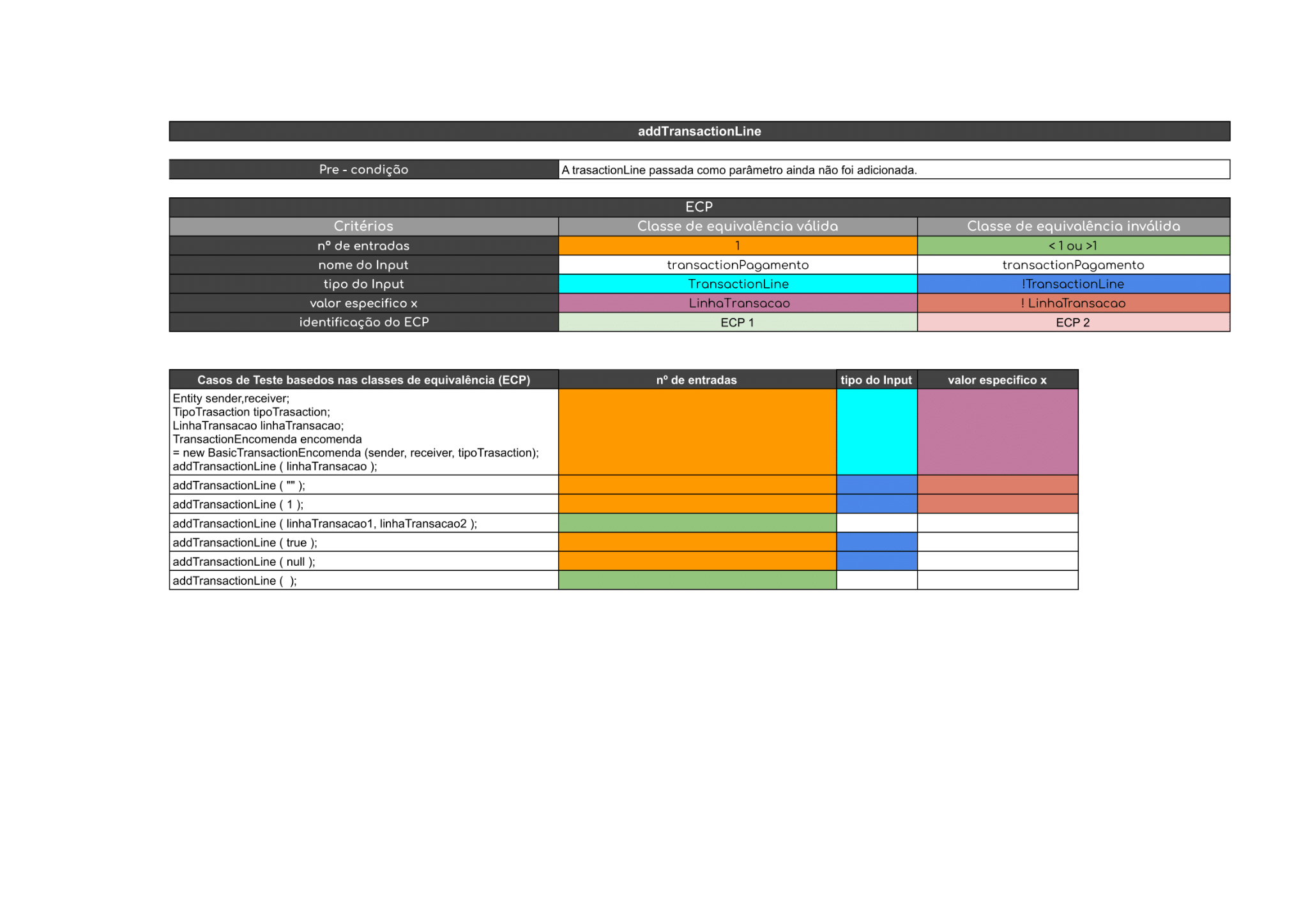


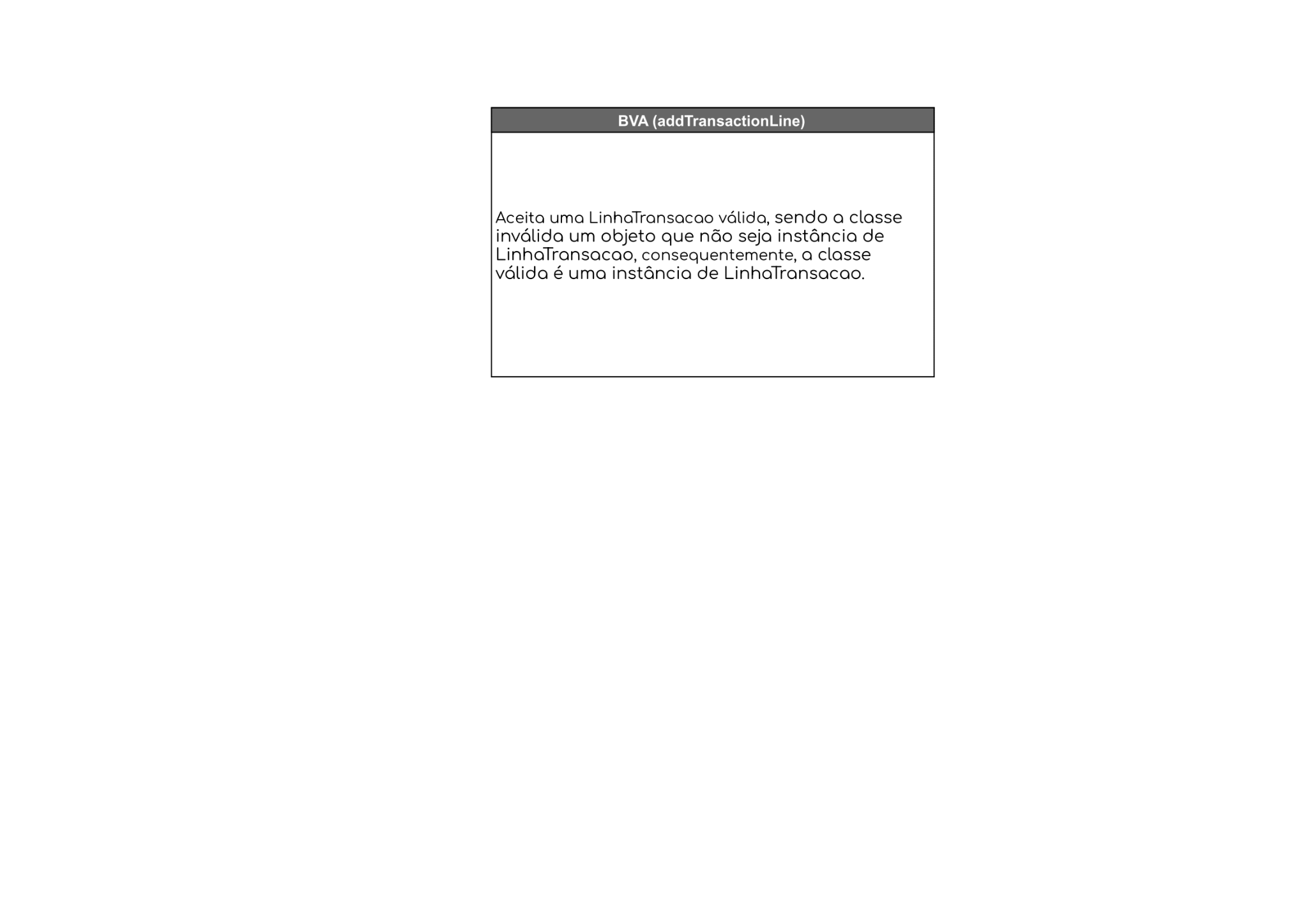


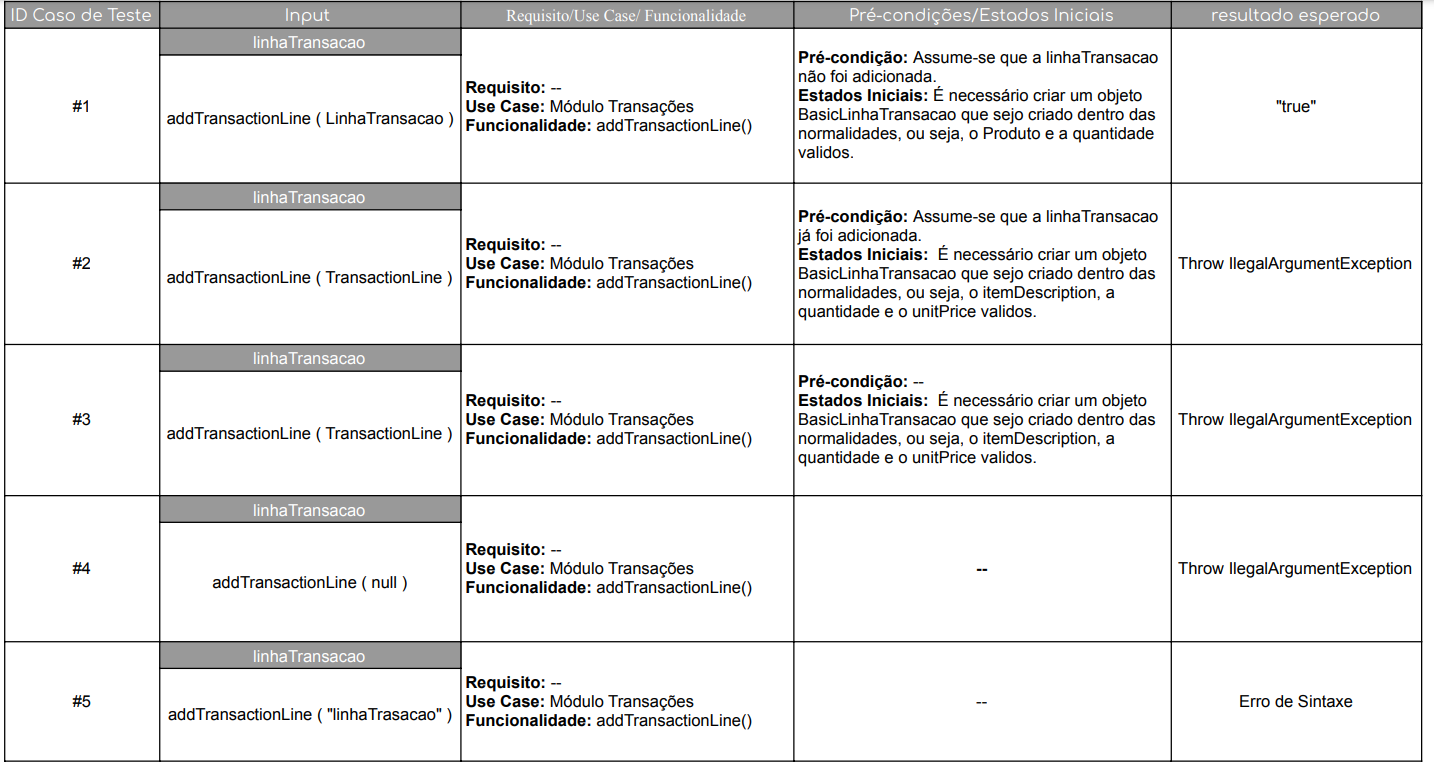
## 

## 2.BasicLinhaTransação

### 2.1 addTransactionLine

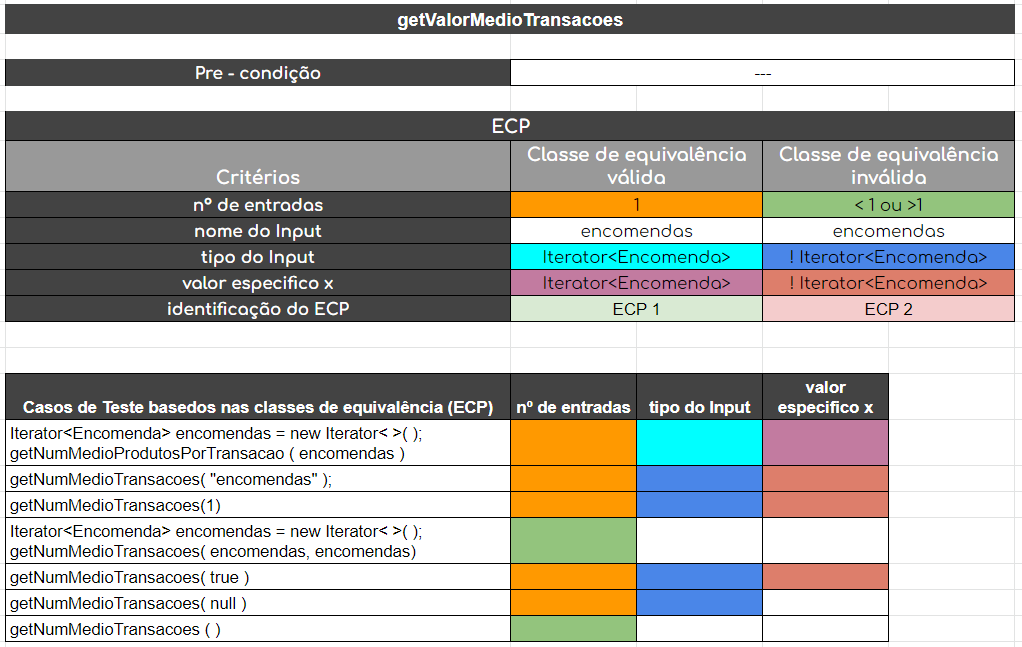


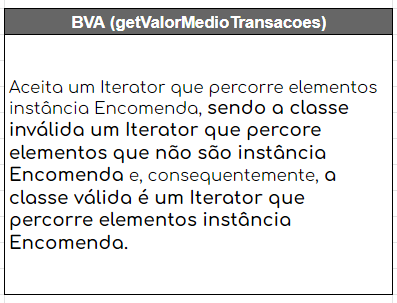


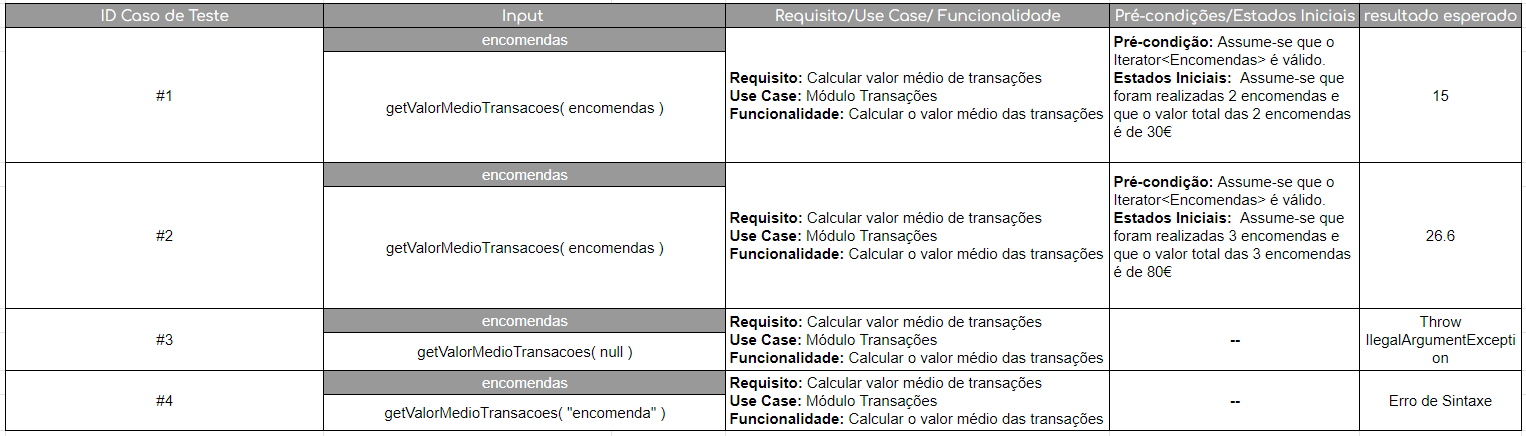


## 3.Estatísticas

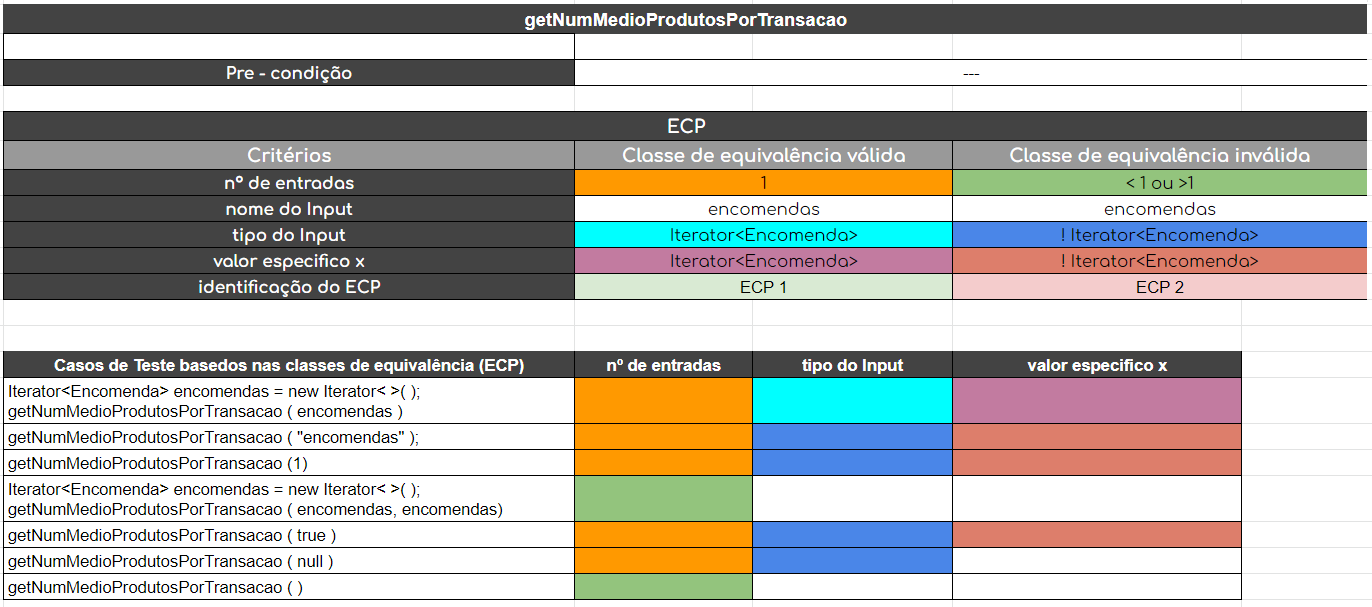
### 3.1 getValorMedioTransacoes



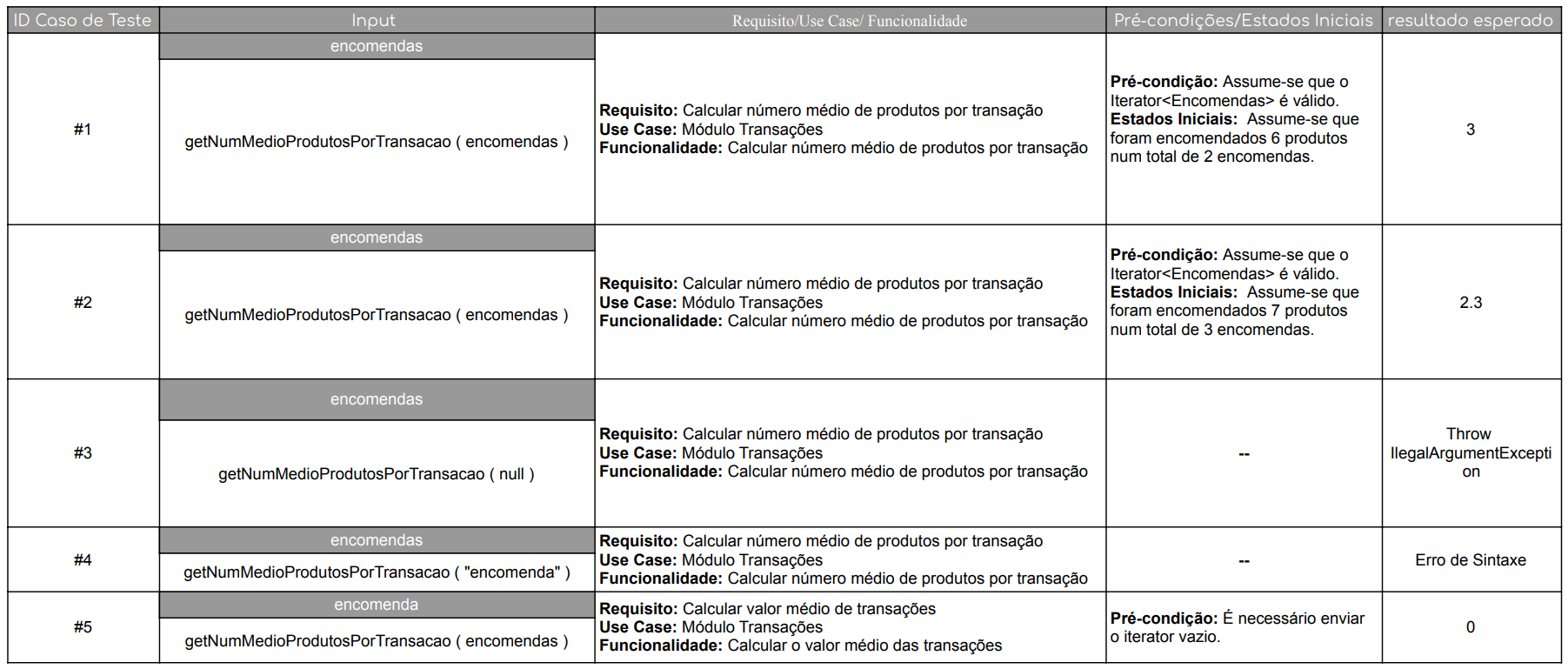




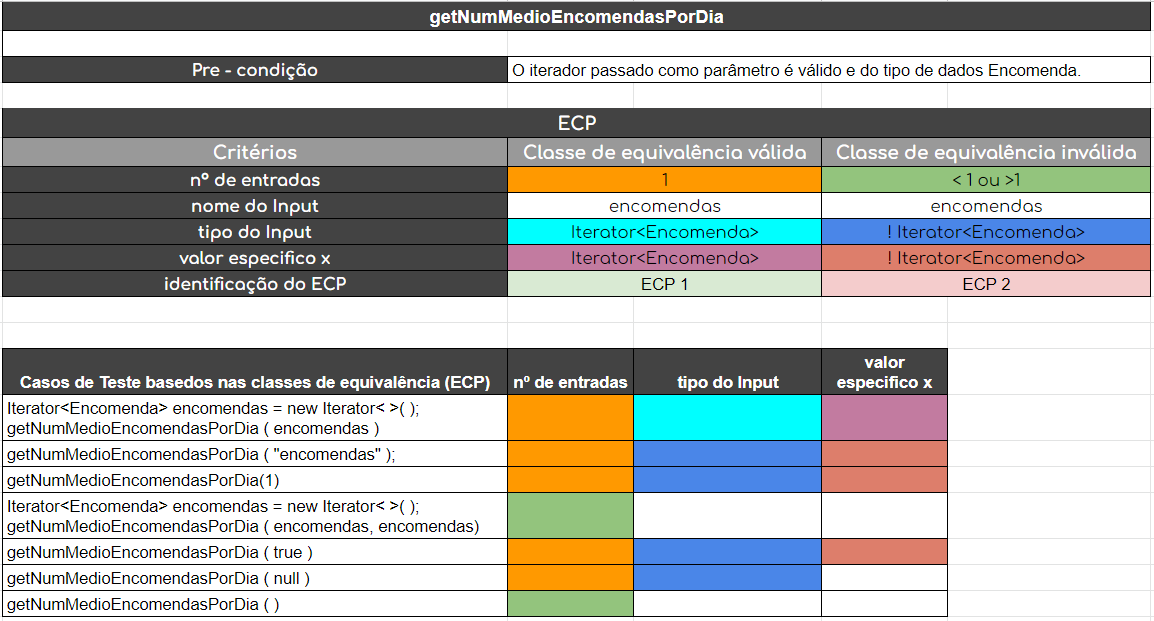
### 3.2 getNumMedioProdutosPorTransacao

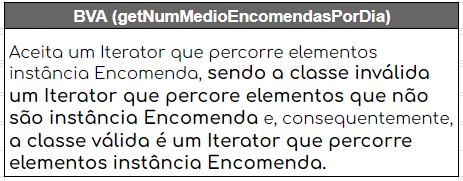


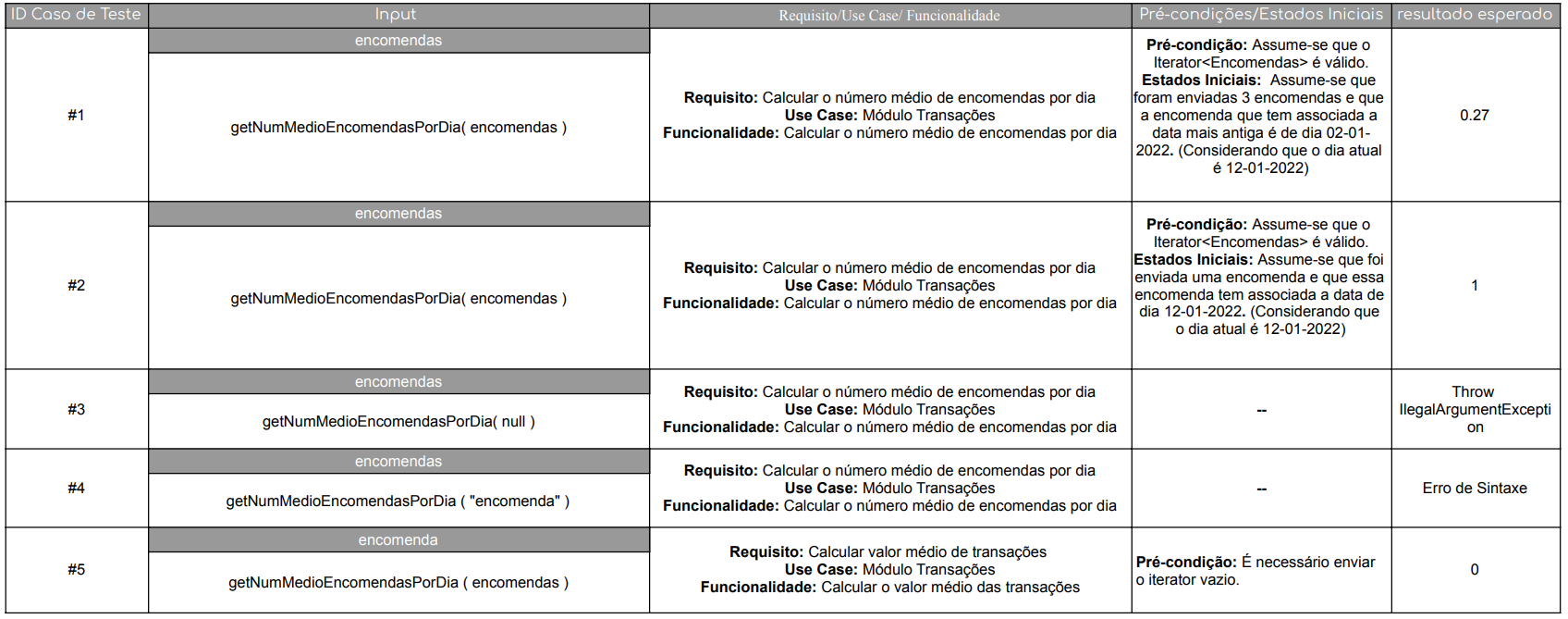
### 



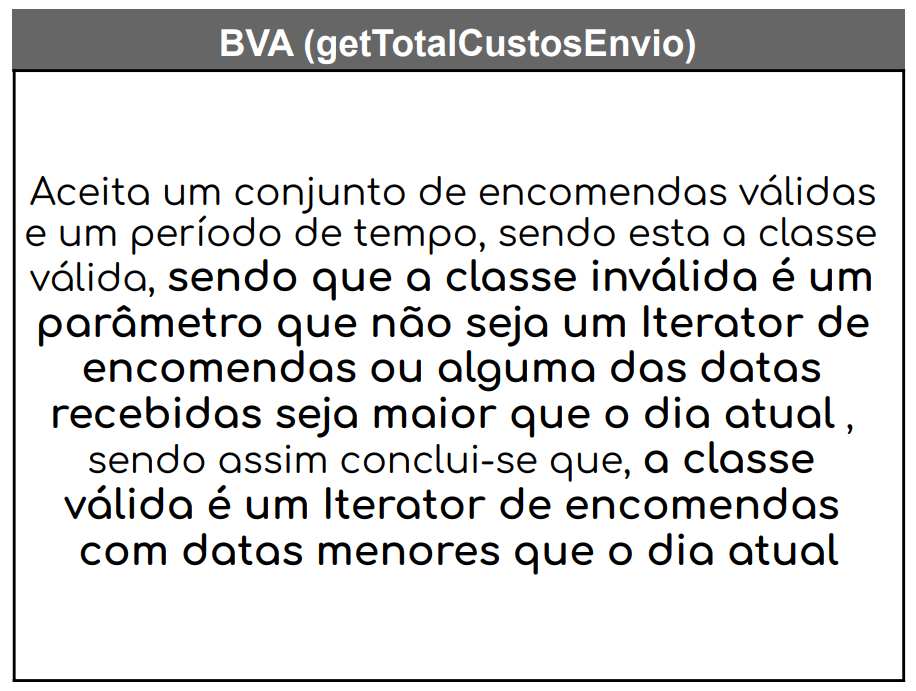
### 3.3 getNumMedioEncomendasPorDia

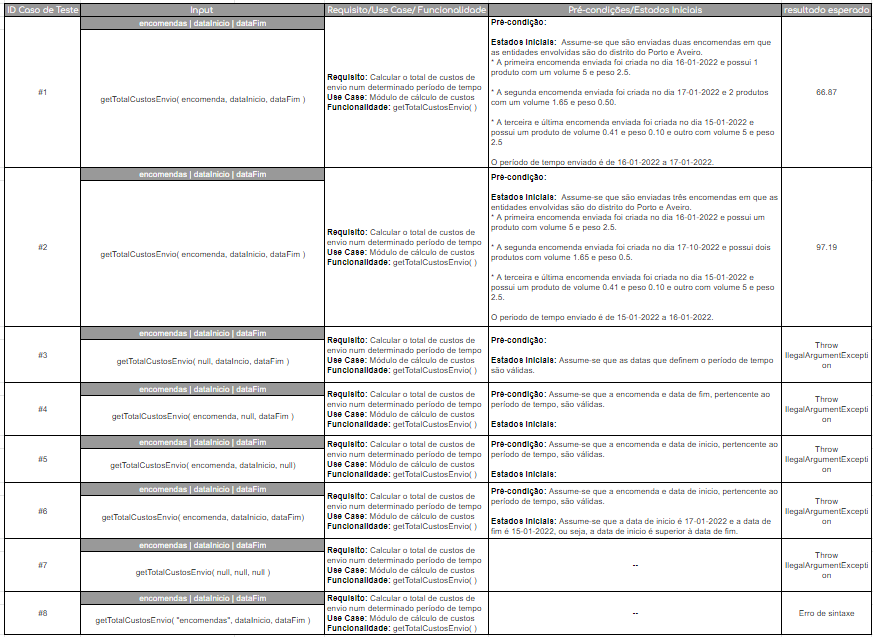




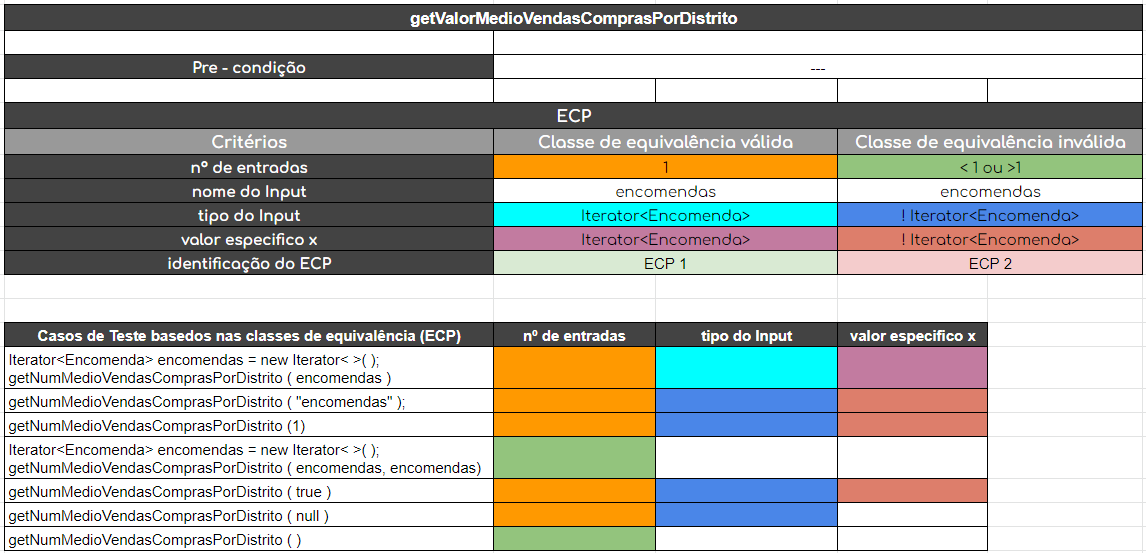


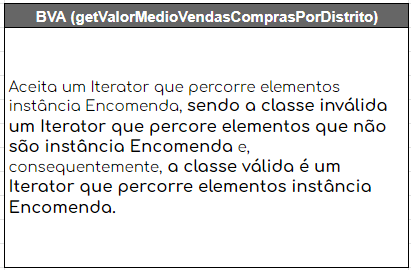
### 3.4 getTotalCustosEnvio

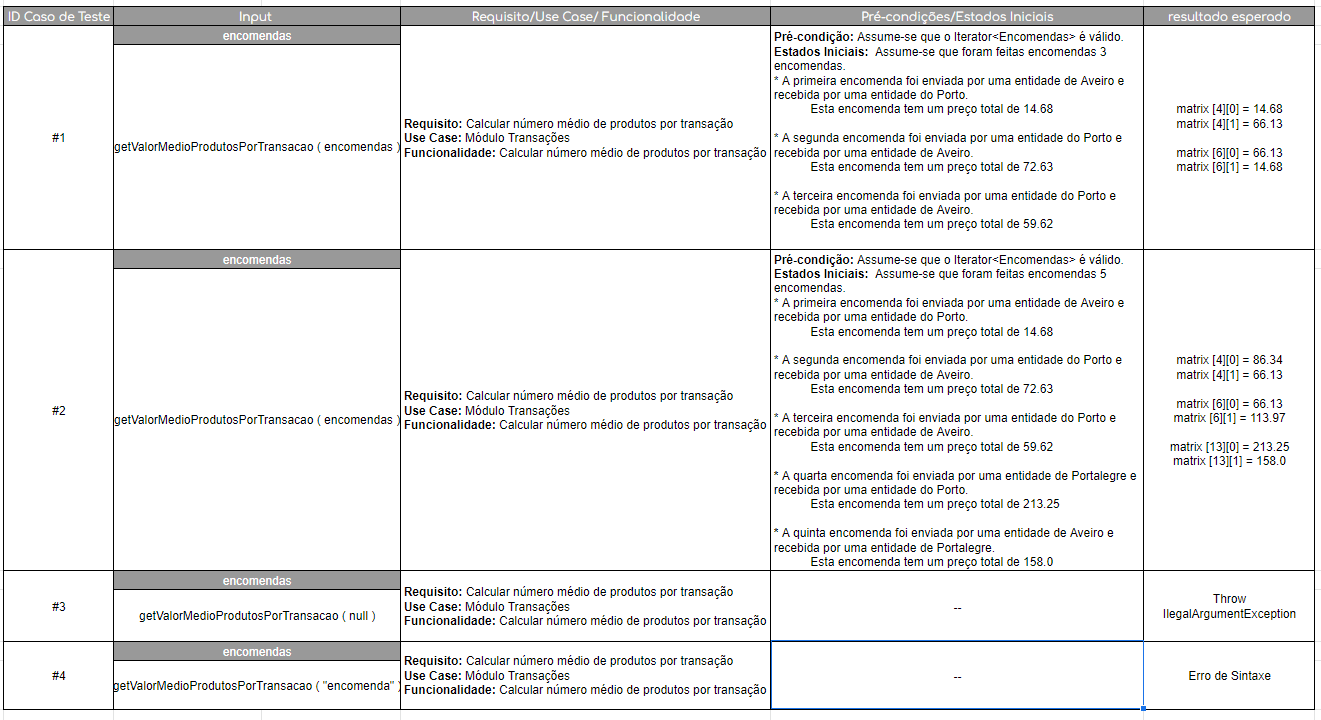




### 3.5 getValorMedioVendasComprasPorDistrito

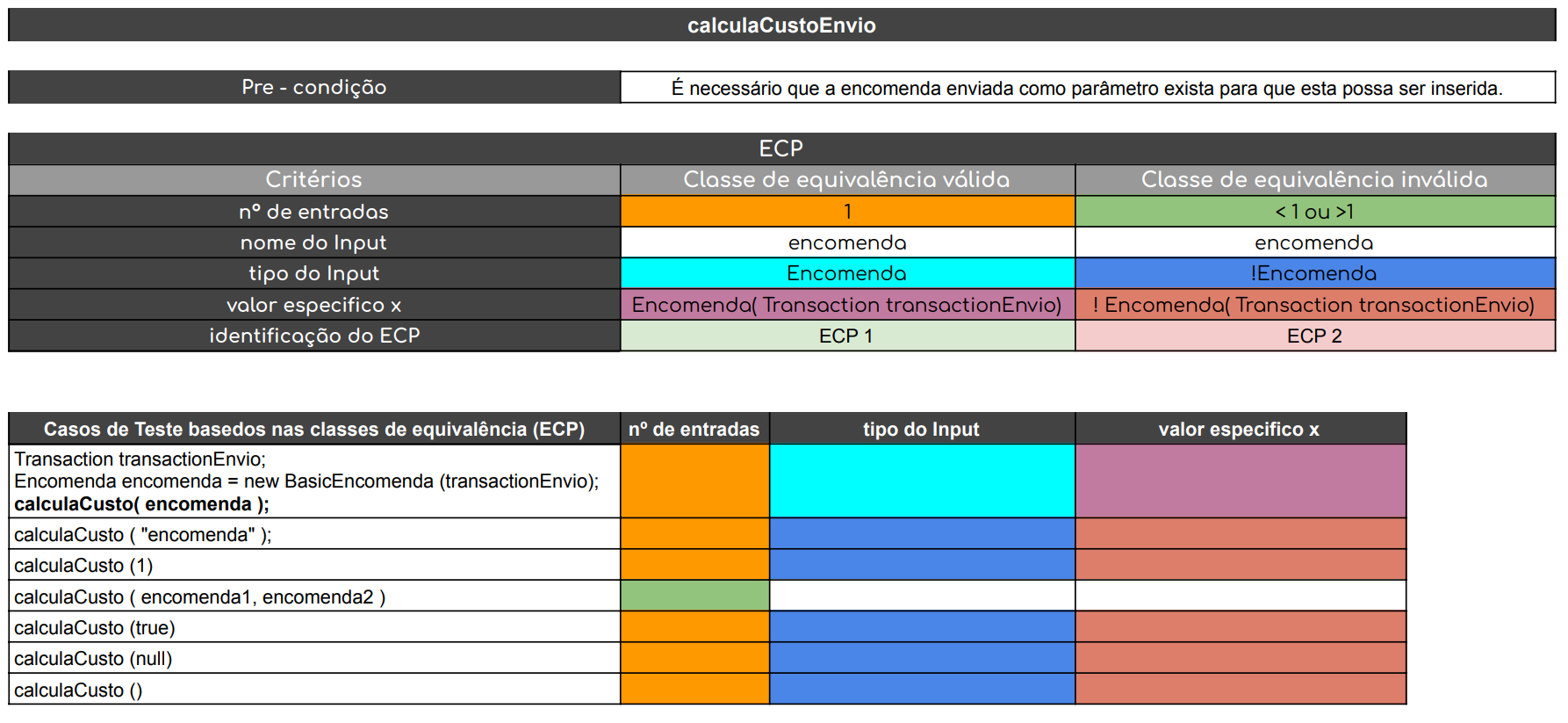


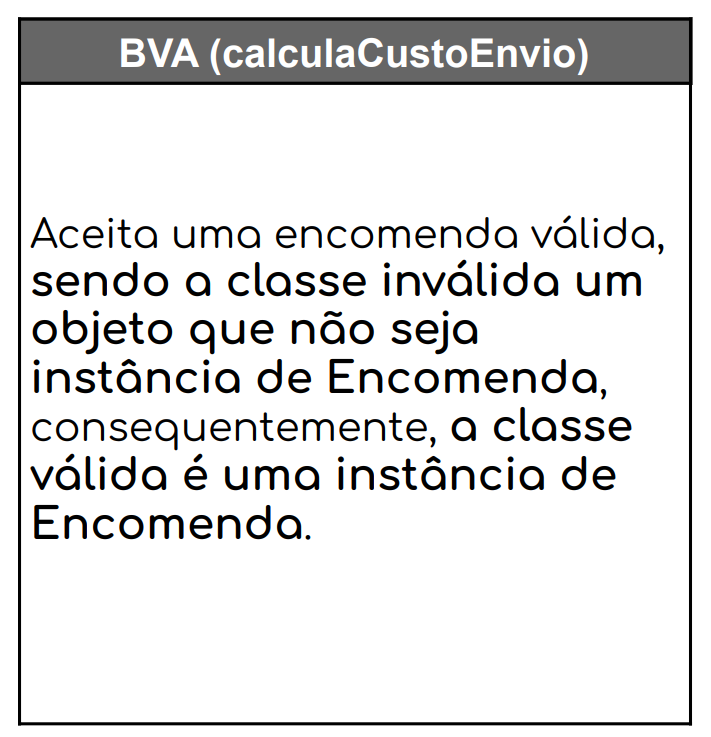


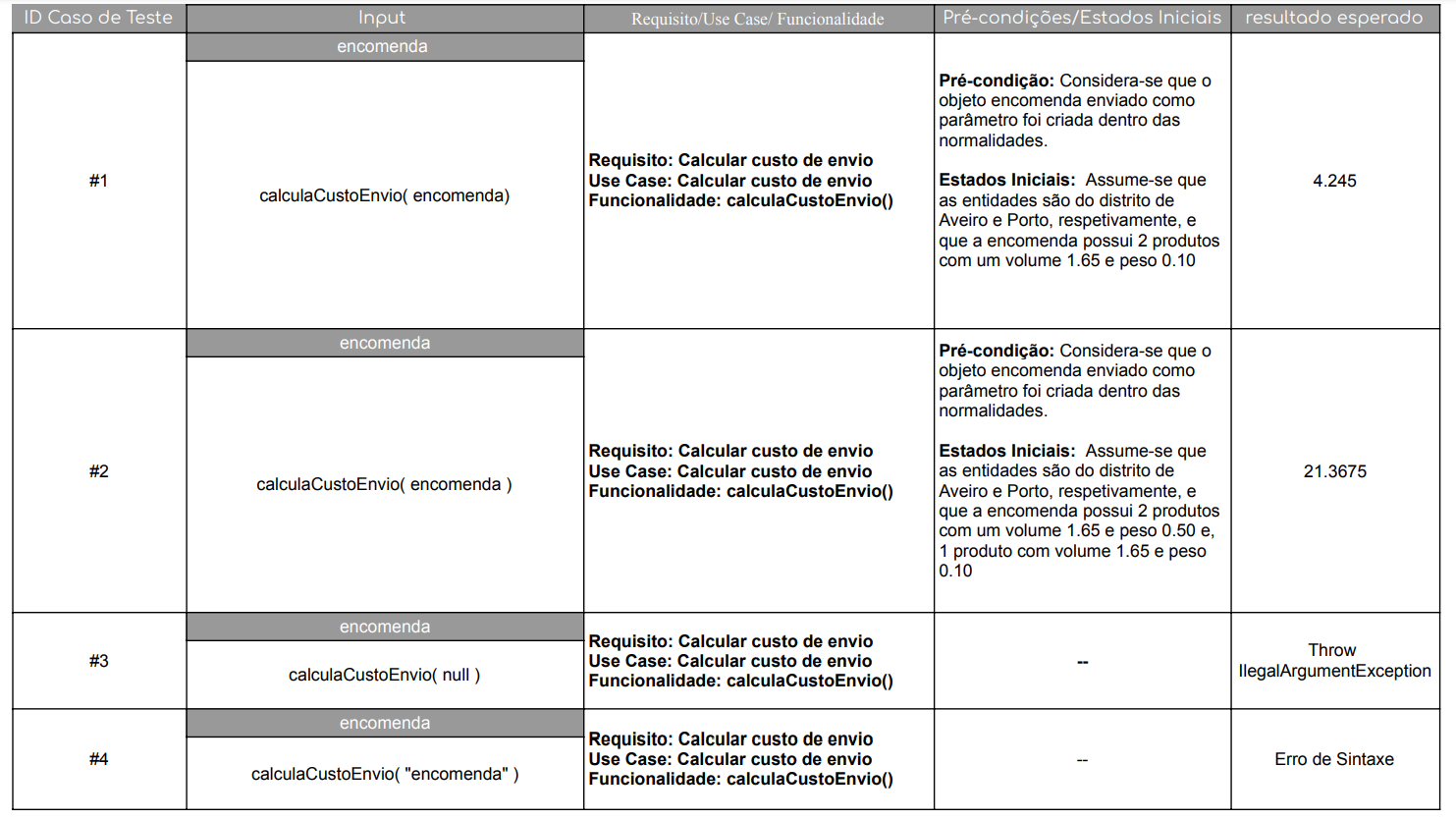


## 4.Custos

### 4.1 calculaCustoEnvio

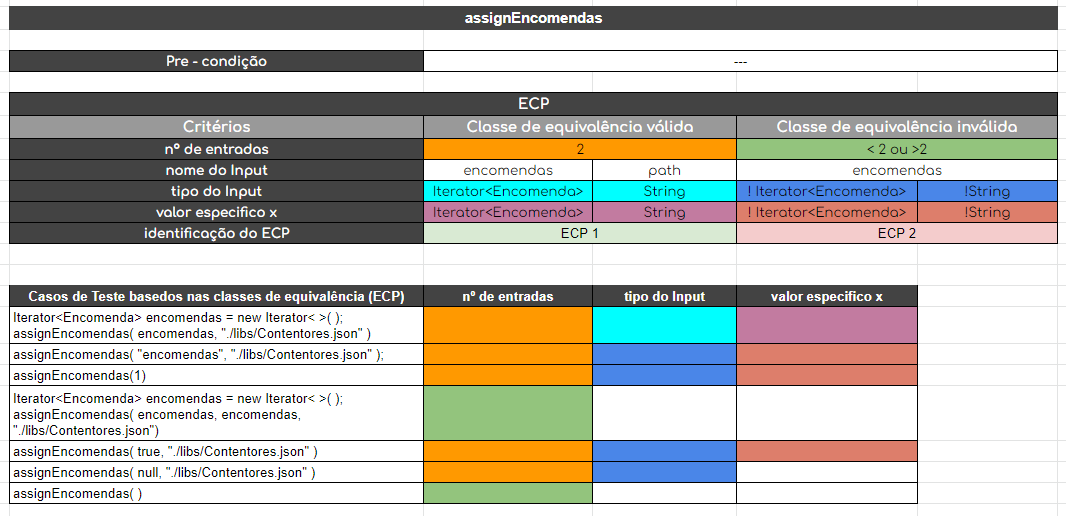


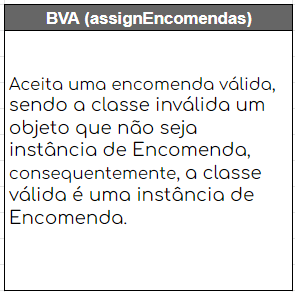


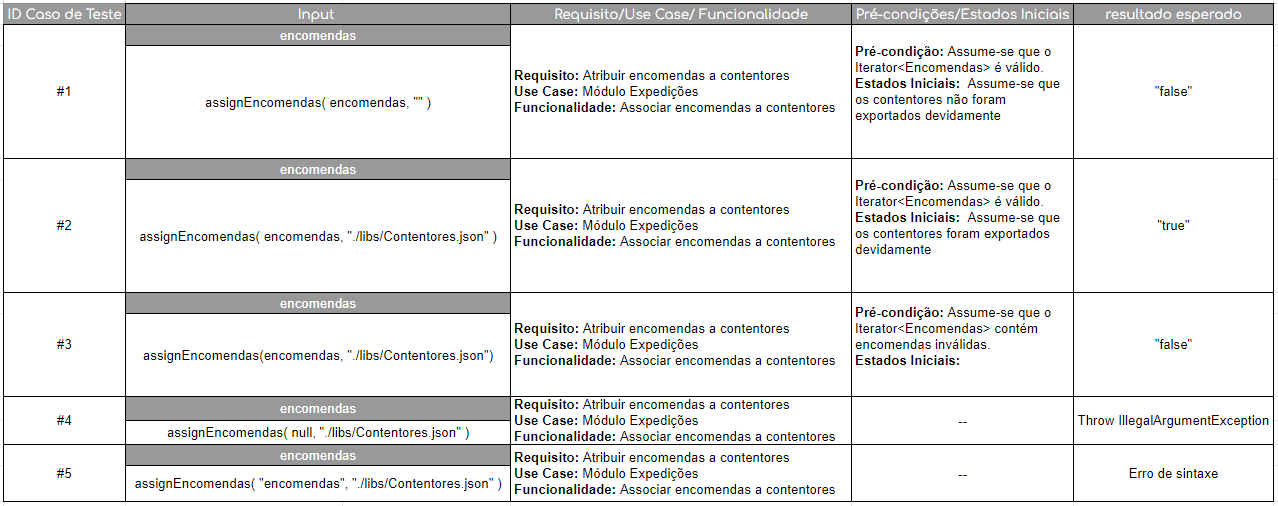


## 5.Expedição

### 5.1 assignEncomendas







# **5. Critérios de passagem ou falha das features**

Através do IDE InteliJ e do plugin JUnit podemos definir um resultado esperado e, ao correr todos os testes, ser feita a comparação entre o resultado obtido e o resultado esperado. Caso sejam iguais, o teste é identificado como “Passado”, caso contrário será identificado como “Falhado”.

Os critérios que serão usados para a determinação dos testes que passaram e falharam será através da informação fornecida pelo Plugin JUnit que, automaticamente compara o resultado esperado com o obtido e caso estes sejam diferentes é notificada a falha nos testes. Tal como se pode observar na imagem seguinte:

