P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Engenharia de Software 2

Trabalho pratico 2

Testes caixa branca

<ES2_2021_GRUPO_11>

Version <1.1>

<20/01/2022>

Realizado por:

8180378-Hugo Silva

8200306-Fábio Costa

Histórico de Versões

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	Hugo	18-01-2022	Todos	18-01-2022	Fluxo grama de teste de caixa branca
1.1	Hugo	19-01-2022	Todos	19-01-2022	Fluxo grama de teste de caixa branca

Conteúdo

1.Introdução

Identificador do documento

WhiteBoxTest.

Âmbito

Este documento refere-se a um relatório de testes de caixa branca desenvolvido para a disciplina de Engenharia de Software II do curso de Engenharia Informática.

Testes Caixa branca

Neste tipo de teste o *tester* tem acesso ao código. Deve ser abordado o fluxo de dados, o fluxo de controlo e ramificação. Os testes de "caixa branca" são mais apropriados para testar componentes pequenos (pelo facto do detalhe requerido para o desenho do teste ser muito elevado)[1]. Assim sendo neste trabalho será selecionado um método mais pequeno de modo que o desenho do fluxograma seja facilitado e outro mais complexo para ser demonstrado que esta informação é verdadeira.

Plugin utilizado

Para obter a cobertura dos testes foi utilizado o *plugin* Jacoco.

[1] Informação retirada do slide 56 "tests de software"

Método getDistanceValueFromShipping

Fluxograma

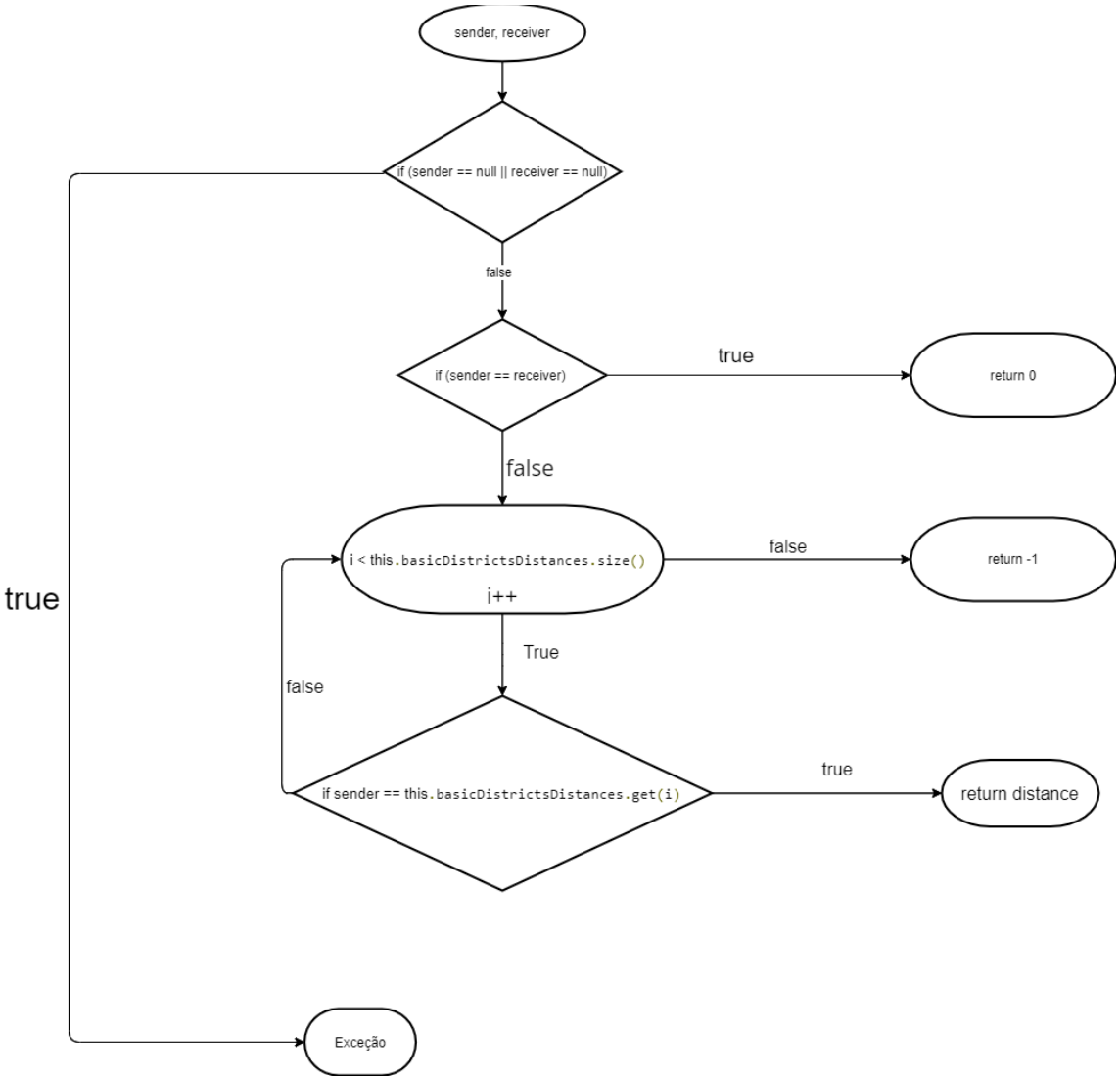

Figura 1 Fluxograma método getDistanceValueFromShipping

O que existe dos testes de caixa preta:

Nos testes de caixa preta foram testadas uma boa parte das condições do fluxograma, no entanto existe instrução que não foi testada.

Primeira condição: um dos inputs é null, **testado**;

Segunda condição: *receiver* e *sender* iguais, **condição não testada**;

Terceira condição: não existir distancia entre o *sender* e o *receiver*, **testado**;

Procedimento:

Invocar método getDistanceValueFromShipping , passando o mesmo receiver e sender como inputs de forma a cobrir a instrução que ainda não foi testada.

```
    /**
     * {@inheritDoc}
     */
    @Override
    public int getDistanceValueFromShipping(Entity sender, Entity receiver) {
        if (sender == null || receiver == null) {
            throw new IllegalArgumentException("The entities can't be null.");
        }

        if (sender.getDistrict().equals(receiver.getDistrict())) {
            return 0;
        }

        for (int i = 0; i < this.basicDistrictsDistances.size(); i++) {
            if (sender.getDistrict().equals(this.basicDistrictsDistances.get(i).getName())) {

                for (int j = 0; j < this.basicDistrictsDistances.get(i).getDistances().size(); j++) {

                    if (receiver.getDistrict().equals(this.basicDistrictsDistances.get(i).getDistances().get(j).getId())) {
                        return this.basicDistrictsDistances.get(i).getDistances().get(j).getDistanceValue();
                    }
                }
            }
        }

        return -1;
    }
```

Figura 2 branch não testado

Identificação do Teste

testGetDistanceValueFromShippingWhiteBox01

Resultado esperado

É esperado que o método retorne 0.

Cobertura do Método após teste de caixa branca

```
    /**
     * {@inheritDoc}
     */
    @Override
    public int getDistanceValueFromShipping(Entity sender, Entity receiver) {
        if (sender == null || receiver == null) {
            throw new IllegalArgumentException("The entities can't be null.");
        }

        if (sender.getDistrict().equals(receiver.getDistrict())) {
            return 0;
        }

        for (int i = 0; i < this.basicDistrictsDistances.size(); i++) {
            if (sender.getDistrict().equals(this.basicDistrictsDistances.get(i).getName())) {

                for (int j = 0; j < this.basicDistrictsDistances.get(i).getDistances().size(); j++) {

                    if (receiver.getDistrict().equals(this.basicDistrictsDistances.get(i).getDistances().get(j).getId())) {
                        return this.basicDistrictsDistances.get(i).getDistances().get(j).getDistanceValue();
                    }
                }
            }
        }

        return -1;
    }
```

Figura 3 Cobertura após teste de caixa branca

Método groupOrdersByTrucks

Fluxograma

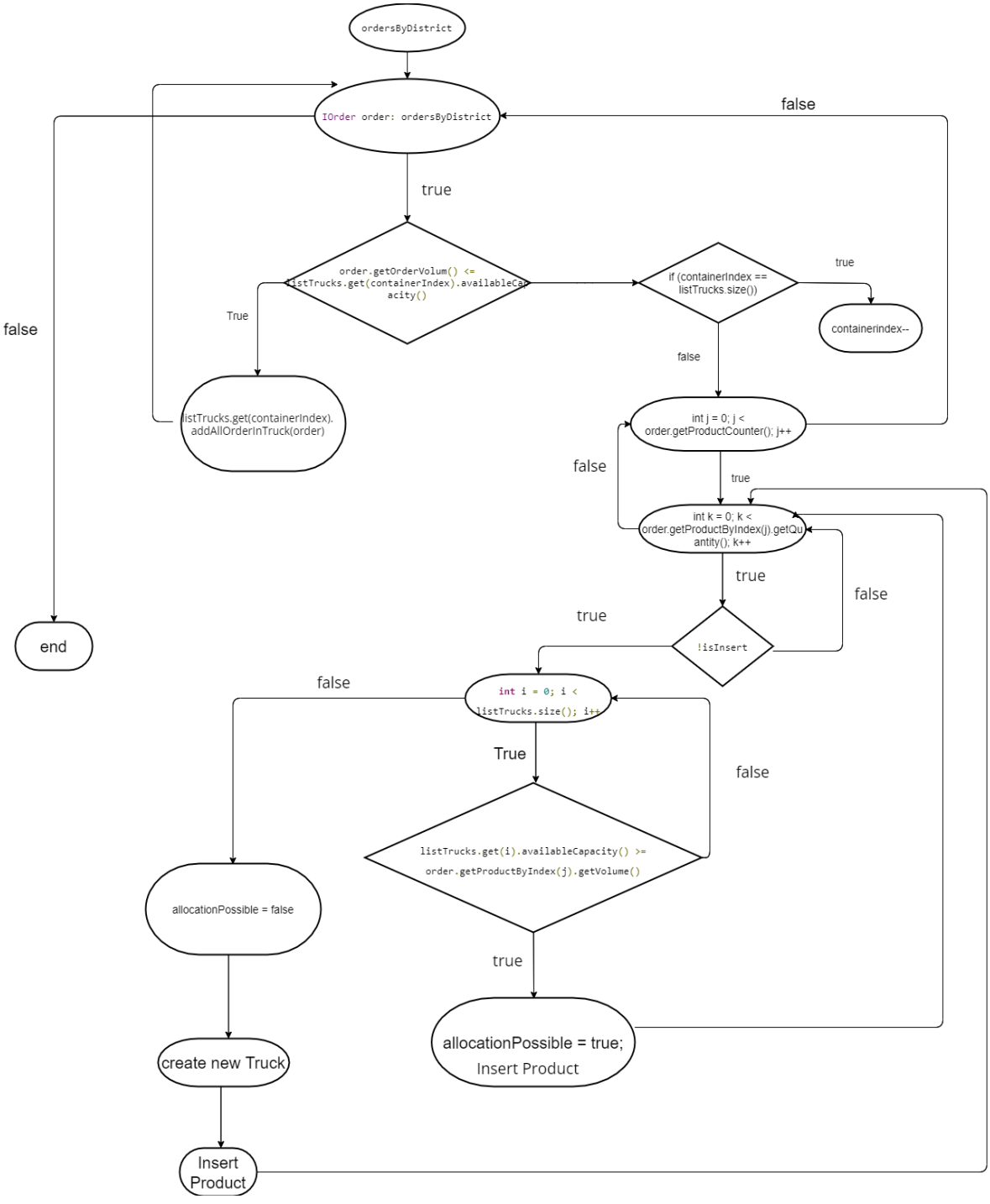

Figura 4 Fluxograma método groupOrdersByTrucks

O que existe dos testes de caixa preta:

Nos testes de caixa preta foram testadas uma boa parte das condições do fluxograma, no entanto existe dois *branchs* não testados.

Primeira condição: order cabe toda dentro de um camião, **testado**;

Segunda condição: índex do camião atual ultrapassa tamanho da lista, **condição não testada**;

Terceira condição: camião onde produto era previsto inserir sem capacidade, é procurado espaço noutro camião, **testado**;

Quarta condição: produto não cabe em nenhum dos camiões atuais, ou seja, é preciso mais um camião, **não testado esta branch**;

Procedimento:

Invocar método groupOrdersByTrucks:

Input

Camiões

Camião 1 = 63 m^3

Camiao 2 = 63 m^3

Total: 126m^3

Produtos:

Produto 1 : 62 m^3

Produto 2 : 62 m^3

Produto 3 : 2 m^3

Total: 126m^3

Segunda condição: segunda iteração do for irá cobrir este método;

Quarta condição: Embora dois camiões levem 126 m^3 serão necessários 3 camiões para a encomenda, dado que em nenhum existe espaço para o produto 3. Assim será coberta esta *branch*.

Identificação do Teste

testGroupOrdersByTrucksWhiteBox02.

Resultado esperado

É esperado que o método retorne 3 camiões alocados.

Cobertura do Método antes do teste de caixa branca

```
@Override
public void groupOrders() {
    int containerIndex = 0;

    for (IOrder order : ordersByDistrict) {

        if (order.getOrderVolum() <= listTrucks.get(containerIndex).availableCapacity()) {
            listTrucks.get(containerIndex).addAllOrderInTruck(order);
        } else {

            containerIndex++;
            if (containerIndex == listTrucks.size()) {
                containerIndex--;
            }
            for (int j = 0; j < order.getProductCounter(); j++) {
                for (int k = 0; k < order.getProductByIndex(j).getQuantity(); k++) {
                    Boolean isInsert = listTrucks.get(containerIndex).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());

                    //NAO INSERIDO
                    if (!isInsert) {

                        boolean allocationPossible = false;
                        for (int i = 0; i < listTrucks.size(); i++) {
                            if (listTrucks.get(i).availableCapacity() >= order.getProductByIndex(j).getVolume()) {

                                allocationPossible = true;
                                listTrucks.get(i).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());
                                break;
                            }
                        }

                        if (!allocationPossible) { //Peciso novo camiao

                            Truck container = new Truck("Truck" + String.valueOf(listTrucks.size() - 1));
                            listTrucks.add(container);

                            isInsert = listTrucks.get(listTrucks.size() - 1).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());
                        }
                    }
                }
            }
        }

    }
}
```

Figura 5 branchs não testado

Cobertura do Método após o teste de caixa branca

```
@Override
public void groupOrders() {
    int containerIndex = 0;

    for (IOrder order : ordersByDistrict) {

        if (order.getOrderVolum() <= listTrucks.get(containerIndex).availableCapacity()) {
            listTrucks.get(containerIndex).addAllOrderInTruck(order);
        } else {

            containerIndex++;
            if (containerIndex == listTrucks.size()) {
                containerIndex--;
            }
            for (int j = 0; j < order.getProductCounter(); j++) {
                for (int k = 0; k < order.getProductByIndex(j).getQuantity(); k++) {
                    Boolean isInsert = listTrucks.get(containerIndex).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());

                    //NAO INSERIDO
                    if (!isInsert) {

                        boolean allocationPossible = false;
                        for (int i = 0; i < listTrucks.size(); i++) {
                            if (listTrucks.get(i).availableCapacity() >= order.getProductByIndex(j).getVolume()) {

                                allocationPossible = true;
                                listTrucks.get(i).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());
                                break;
                            }
                        }

                        if (!allocationPossible) { //Peciso novo camiao

                            Truck container = new Truck("Truck" + String.valueOf(listTrucks.size() - 1));
                            listTrucks.add(container);

                            isInsert = listTrucks.get(listTrucks.size() - 1).addProductInTruck(order.getId(), String.valueOf(order.getProductByIndex(j).getId()), order.getProductByIndex(j).getVolume());
                        }
                    }
                }
            }
        }

    }
}
```

Figura 6 Cobertura após caixa branca

testGroupOrdersByTrucksWhiteBox02

Test Case ID	TC# testGroupOrdersByTrucksWhiteBox02.	**Dependências**	

Data de Execução	**Hora de Execução**	***Tester***	**Pass/Fail**	**Resumo da Falha**
16 janeiro	23:57h	Hugo	PASS	

Pré-condições / estado inicial
-Objeto do tipo ExpeditionMethods instanciado; - 1 Objetos do tipo Order instanciados (Deve ter mesmo emisor e o mesmo receptor); - 1 Objeto do tipo Orders instanciado; - 2 objetos do tipo Product instanciado;
Procedimentos
- Adicionar produtos às order de como a que o volume total seja 126m^3; - Invocar método groupOrdersByTrucks, devem ser retornados 3 camiões porque existe um produto que não cabe nos dois camiões inicialmente previstos ; - Verificar número de camiões criados;

Requisito/Use Case/ Funcionalidade	**Inputs**	**Valores do Inputs**	**Resultados Esperados**	**Pass/Fail/Untested**
Agrupar encomendas por distrito	Ver tabela XYW	Ver tabela XYW	3	PASS

Tabela auxiliar XYM

```
this.product3 = new Product(3, "nome", "produto reacondiciona", 5.0, 6, 1, 4);

this.basicOrder1 = new Order("20210101_01", "2021-2-2", basicEntity1, basicEntity2);
this.basicOrder2 = new Order("20210101_02", "2021-2-2", basicEntity1, basicEntity4);
this.basicOrder3 = new Order("3", "2021-2-2", basicEntity3, basicEntity3);

basicOrder3.addProduct(product);
basicOrder3.addProduct(product);
basicOrder3.addProduct(product3);
basicOrders.addOrder(basicOrder3);
```

Resultado Esperado:
3