

PONTIFÍCIA UNIVERSIDADE CATÓLICA DO RIO GRANDE DO SUL

Verificação e Validação de Software

T2 – Tarefa sobre “Teste de Unidade”

Acadêmicos:

Gustavo Santos Silva

Henrique Baptista de Oliveira

Pedro Carlo Brun Iglesias

PORTO ALEGRE, RS

2022

1. Projetos dos casos de teste

Para o desenvolvimento dos casos de teste foi utilizado a técnica de valores limites. Portanto a partir do enunciado entregue, foram desenvolvidos casos para os valores de entrada da interface descrita. Sendo assim foram projetados casos de valores limite para a inicialização do centro de distribuição, para o recebimento de combustíveis e para a entrega a postos de gasolina, dessa forma gerando os seguintes casos:

- Inicialização do centro de distribuição:

Nos casos de inicialização foram consideradas as porcentagens de virada de estado NORMAL, SOBREAviso e EMERGÊNCIA para a geração dos casos.

Inicialização Normal	Tanques $\geq 50\%$	on	{500, 10000, 1250, 1250}	
		off		{250, 5000, 625, 625}
Inicialização Sobreaviso	Tanques $\leq 49\%$	on	{249, 5000, 625, 625}	
		off		{250, 5000, 624, 625}
	Tanques $\geq 25\%$	on	{125, 2500, 313, 313}	
		off		{250, 4999, 625, 625}
Inicialização Emergência	Tanques $< 25\%$	on	{124, 2500, 313, 313}	
		off		{125, 2500, 311, 313}
	Tanques $\geq 0\%$	on	{0, 0, 0, 0}	
		off		{125, 2500, 313, 311}

- Recebimento de Combustíveis:

Já no caso dos combustíveis foi utilizado os limites do retorno do método de recebimento de combustíveis para os três tipos (álcool, gasolina, aditivo).

Variável	Condição	Tipo	T1	T2	T3	T4	T5
Aditivo com Excesso	$x \leq 250$	on	250				
Tanque Aditivo = 250		off		125			
	$x \geq 0$	on			0		
		off				250	
	typical	in					125
Aditivo sem Excesso	$x > 250$	on	251				
Tanque Aditivo = 250		off		500			
	typical	in			500		
Aditivo Inválido	$x < 0$	on	-1				
Tanque Aditivo = 0		off		-500			

Gasolina sem Excesso	$x \leq 5000$	on	5000				
Tanque Gasolina = 5000		off		2500			
	$x \geq 0$	on			0		
		off				5000	2500
	typical	in					
Gasolina com Excesso	$x > 5000$	on	5001				
Tanque Gasolina = 5000		off		10000			
	typical	in			10000		
Gasolina Inválida	$x < 0$	on	-1				
Tanque Gasolina = 0		off		-5000			

Álcool sem Excesso	$x \leq 1250$	on	1250				
Tanque Álcool = 1250 x2		off		625			
	$x \geq 0$	on			0		
		off				1250	
	typical	in					625
Álcool com Excesso	$x > 1250$	on	1251				
Tanque Álcool = 1250 x2		off		2500			
Álcool Inválido	$x < 0$	on	-1				
Tanque Álcool = 0		off		-2500			

- Encomendas Postos:

Por último nas encomendas de postos de combustíveis, foram feitos casos para as situações de encomendas aceitas, recusadas pela situação de entrega, pela falta de combustível para a mistura e por valores inválidos.

Encomenda Aceita	$x \geq 0$	on	0				
Tanques Cheios		off		500			
	$x \leq 10000$	on			10000		
		off				500	
	typical	in					500
Encomenda Inválida	$x < 0$	on	-1				
		off	-1000				
Encomenda Recusada Situação							
Tanques < 25%	$x < 125$	on	124				
		off		10			
Encomenda Recusada Combustível	$x \geq 600$	on	600				
Tanques < 600		off		1000			

Após a definição dos valores de entrada possíveis e seus resultados esperados o grupo desenvolveu as soluções através de testes parametrizados a fim de deixar o código de teste mais compacto e efetivo, abrangendo todos esses valores, para assim iniciar um desenvolvimento com uma fundamentação sólida (em questões de lógica e prevenção de erros).

2. Código da Classe Driver

```
3. package com.holiveira;
4.
5. import static org.junit.jupiter.api.Assertions.assertArrayEquals;
6. import static org.junit.jupiter.api.Assertions.assertEquals;
7. import static org.junit.jupiter.api.Assertions.assertThrows;
8.
9. import java.security.InvalidParameterException;
10.
11. import com.holiveira.CentroDistribuicao.TIPOPOSTO;
12.
13. import org.junit.jupiter.api.Test;
14. import org.junit.jupiter.params.ParameterizedTest;
15. import org.junit.jupiter.params.provider.MethodSource;
16. import org.junit.jupiter.params.provider.ValueSource;
17.
18. public class CentroDistribuicaoTest {
19.
20.     static CentroDistribuicao cd;
21.
22.     //Testes de Inicialização de Centros de Distribuição
23.
24.     @ParameterizedTest
25.     @MethodSource("casosNormalICD")
26.     public void inicializaCDNormal(int[] casosNormal)
27.     {
28.         cd = new
29.         CentroDistribuicao(casosNormal[0],casosNormal[1],casosNormal[2],cas
30.         osNormal[3]);
31.         assertEquals("NORMAL",cd.gettSituacao().toString());
32.     }
33.
34.     @ParameterizedTest
35.     @MethodSource("casosSobreavisoICD")
36.     public void inicializaCDSobreaviso(int[] casosSobreaviso)
37.     {
38.         cd = new
39.         CentroDistribuicao(casosSobreaviso[0],casosSobreaviso[1],casosSobre
40.         aviso[2],casosSobreaviso[3]);
41.         assertEquals("SOBREAVISO",cd.gettSituacao().toString());
42.     }
43.
44.     @ParameterizedTest
45.     @MethodSource("casosEmergenciaICD")
46.     public void inicializaCDEmergencia(int[] casosEmergencia)
47.     {
48.         cd = new CentroDistribuicao(casosEmergencia[0],
49.         casosEmergencia[1] , casosEmergencia[2], casosEmergencia[3]);
```

```

45.         assertEquals("EMERGENCIA",cd.gettSituacao().toString());
46.     }
47.
48.     @ParameterizedTest
49.     @MethodSource("casosInvalidosICD")
50.     public void inicializaCDInvalida(int[] casosInvalidos)
51.     {
52.         assertThrows(InvalidParameterException.class, () -> new
CentroDistribuicao(casosInvalidos[0], casosInvalidos[1] ,
casosInvalidos[2], casosInvalidos[3]) );
53.     }
54.
55.     public static int[][] casosNormalICD() {
56.         return new int[][] { { 500 , 10000, 1250, 1250 }, { 250,
5000, 625, 625 } };
57.     }
58.
59.     public static int[][] casosSobreavisoICD() {
60.         return new int[][] { { 249, 5000, 625, 625 }, { 250, 4999,
625, 625 }, { 250, 5000, 624, 625 }, { 250, 5000, 625, 624 }, {125,
2500, 313, 313} };
61.     }
62.
63.     public static int[][] casosEmergenciaICD() {
64.         return new int[][] { {124, 2500, 313, 313}, {125, 2499,
313, 313}, {125, 2500, 311, 313}, {125, 2500, 313, 311}, {0, 0, 0,
0} };
65.     }
66.
67.     public static int[][] casosInvalidosICD() {
68.         return new int[][]
69.         {
70.             {-1, 2500, 313, 313},
71.             {-1, -1, 313, 313},
72.             {-1, -1, -1, 313},
73.             {-1, -1, -1, 313},
74.             {-1, -1, -1, -1},
75.
76.             {125, -1, 313, 313},
77.             {125, -1, -1, 313},
78.             {125, -1, -1, -1},
79.
80.             {125, 2500, -1, 313},
81.             {125, 2500, -1, -1},
82.
83.             {125, 2500, 313, -1},
84.
85.             {50000, 2500, 313, 313},
86.             {50000, 50000, 313, 313},

```

```

87.         {50000, 50000, 50000, 313},
88.         {50000, 50000, 50000, 50000},
89.
90.         {125, 50000, 313, 313},
91.         {125, 50000, 50000, 313},
92.         {125, 50000, 50000, 50000},
93.
94.         {125, 2500, 50000, 313},
95.         {125, 2500, 50000, 50000},
96.
97.         {125, 2500, 313, 50000}
98.
99.     };
100. }
101.
102.     //Teste de Abastecimento do CD
103.
104.     @ParameterizedTest
105.     @ValueSource( ints = {0, 125, 250} )
106.     public void recebeAditivoSemExcesso(int aditivo){
107.
108.         cd = new CentroDistribuicao(250, 0, 0, 0);
109.         assertEquals( aditivo, cd.recebeAditivo(aditivo) );
110.
111.     }
112.
113.     @ParameterizedTest
114.     @ValueSource( ints = {251, 500} )
115.     public void recebeAditivoComExcesso(int aditivo){
116.
117.         cd = new CentroDistribuicao(250, 0, 0, 0);
118.         int aditivoArmazenado = cd.getAditivo();
119.         assertEquals( CentroDistribuicao.MAX_ADITIVO -
aditivoArmazenado , cd.recebeAditivo(aditivo) );
120.
121.     }
122.
123.     @ParameterizedTest
124.     @ValueSource( ints = {-1, -500} )
125.     public void recebeAditivoInvalido(int aditivo){
126.         cd = new CentroDistribuicao(0, 0, 0, 0);
127.         assertEquals( -1 , cd.recebeAditivo(aditivo) );
128.     }
129.
130.     @ParameterizedTest
131.     @ValueSource( ints = {0, 2500, 5000} )
132.     public void recebeGasolinaSemExcesso(int gasolina){
133.
134.         cd = new CentroDistribuicao(0, 5000, 0, 0);

```

```
135.         assertEquals( gasolina, cd.recebeGasolina(gasolina)
136.     );
137.     }
138.
139.     @ParameterizedTest
140.     @ValueSource( ints = {5001, 10000})
141.     public void recebeGasolinaComExcesso(int gasolina){
142.
143.         cd = new CentroDistribuicao(0, 5000, 0, 0);
144.         int gasolinaArmazenada = cd.gettGasolina();
145.         assertEquals( CentroDistribuicao.MAX_GASOLINA -
gasolinaArmazenada , cd.recebeGasolina(gasolina) );
146.
147.     }
148.
149.     @ParameterizedTest
150.     @ValueSource( ints = {-1, -5000})
151.     public void recebeGasolinaInvalido(int gasolina){
152.         cd = new CentroDistribuicao(0, 0, 0, 0);
153.         assertEquals( -1 , cd.recebeGasolina(gasolina) );
154.     }
155.
156.     @ParameterizedTest
157.     @ValueSource( ints = {0, 625, 1250} )
158.     public void recebeAlcoolSemExcesso(int alcool){
159.
160.         cd = new CentroDistribuicao(0, 0, 625, 625);
161.         assertEquals( alcool, cd.recebeAlcool(alcool) );
162.
163.     }
164.
165.     @ParameterizedTest
166.     @ValueSource( ints = {1251, 2500})
167.     public void recebeAlcoolComExcesso(int alcool){
168.
169.         cd = new CentroDistribuicao(0, 0, 625, 625);
170.         int alcoolArmazenado = cd.gettAlcool1()*2;
171.         assertEquals( CentroDistribuicao.MAX_ALCOOL -
alcoolArmazenado , cd.recebeAlcool(alcool) );
172.
173.     }
174.
175.     @ParameterizedTest
176.     @ValueSource( ints = {-1, -2500})
177.     public void recebeAlcoolInvalido(int alcool){
178.         cd = new CentroDistribuicao(0, 0, 0, 0);
179.         assertEquals( -1 , cd.recebeAlcool(alcool) );
180.     }
```

```

181.
182.         //Testes de Encomendas de Postos
183.
184.         @ParameterizedTest
185.         @ValueSource( ints = {500, 10000})
186.         public void encomendaAceitado(int combustivel){
187.             cd = new CentroDistribuicao(500, 10000, 1250, 1250);
188.             int[] tanquesAtual =
189.                 cd.encomendaCombustivel(combustivel, TIPOPOSTO.COMUM);
190.             assertEquals(new int[]{cd.gettAditivo(),
191.                 cd.gettGasolina(), cd.gettAlcool1(), cd.gettAlcool2()},
192.                 tanquesAtual);
193.         }
194.
195.         @ParameterizedTest
196.         @ValueSource(ints = {-1, -1000})
197.         public void encomendaRecusadoParametro(int combustivel){
198.             cd = new CentroDistribuicao(0, 0, 0, 0);
199.             assertEquals(new int[]{-7},
200.                 cd.encomendaCombustivel(combustivel, TIPOPOSTO.COMUM));
201.             assertEquals(new int[]{-7},
202.                 cd.encomendaCombustivel(combustivel, TIPOPOSTO.ESTRATEGICO));
203.         }
204.
205.         @ParameterizedTest
206.         @ValueSource(ints = {1, 10, 100, 1000})
207.         public void encomendaRecusadoSituacao(int combustivel){
208.             cd = new CentroDistribuicao(120, 2400, 300, 300);
209.             assertEquals(new int[]{-14},
210.                 cd.encomendaCombustivel(combustivel, TIPOPOSTO.COMUM));
211.         }
212.
213.         @ParameterizedTest
214.         @ValueSource(ints = {600, 1000})
215.         public void encomendaRecusadoCombustivel(int
216.             combustivel){
217.             cd = new CentroDistribuicao(500, 200, 1250, 1250);
218.             assertEquals(new int[]{-21},
219.                 cd.encomendaCombustivel(combustivel, TIPOPOSTO.ESTRATEGICO));
220.         }
221.
222.         //Code-Coverage
223.
224.         @Test
225.         public void
226.             encomendaRecusadoSituacaoSobreavisoQuantidadeMenor(){
227.             cd = new CentroDistribuicao(249, 10000, 1250, 1250);
228.             assertEquals(new int[]{-14},
229.                 cd.encomendaCombustivel(1, TIPOPOSTO.COMUM));

```



```

220.         }
221.
222.         @Test
223.         public void
224.             encomendaRecusadoSituacaoEmergenciaQuantidadeMenor(){
225.                 cd = new CentroDistribuicao(10, 10000, 1250, 1250);
226.                 assertEquals(new int[]{-14},
227.                     cd.encomendaCombustivel(1, TIPOPOSTO.ESTRATEGICO));
228.             }
229.
230.         @Test
231.         public void
232.             encomendaAceitadoSituacaoSobreavisoQuantidadeMaiorPostoComum(){
233.                 cd = new CentroDistribuicao(249, 10000, 1250, 1250);
234.                 assertEquals(new int[]{246,9965,1243,1243},
235.                     cd.encomendaCombustivel(100, TIPOPOSTO.COMUM));
236.             }
237.
238.         @Test
239.         public void
240.             encomendaAceitadoSituacaoEmergenciaQuantidadeMaiorPostoComum(){
241.                 cd = new CentroDistribuicao(249, 10000, 1250, 1250);
242.                 assertEquals(new int[]{244,9930,1237,1237},
243.                     cd.encomendaCombustivel(100, TIPOPOSTO.ESTRATEGICO));
244.             }
245.
246.         @ParameterizedTest
247.         @MethodSource("casosTanquesVazios")
248.         public void entregaRecusadaTanquesVazios(int[] dados){
249.             cd = new CentroDistribuicao(dados[0], dados[1],
250.                 dados[2], dados[3]);
251.             assertEquals(new int[]{-21},
252.                 cd.encomendaCombustivel(1000, TIPOPOSTO.ESTRATEGICO));

```

3. Código da Classe Alvo

```
4. package com.holiveira;
5.
6. public class CentroDistribuicao {
7.     public enum SITUACAO {
8.         NORMAL, SOBRAVISO, EMERGENCIA
9.     }
10.
11.     public enum TIPOPOSTO {
12.         COMUM, ESTRATEGICO
13.     }
14.
15.     private static final int MAX_ADITIVO = 500;
16.     private static final int MAX_ALCOOL = 2500;
17.     private static final int MAX_GASOLINA = 10000;
18.
19.     private int ad;
20.     private int gas;
21.     private int al1;
22.     private int al2;
23.     private CentroDistribuicao.SITUACAO situacao;
24.
25.     public CentroDistribuicao(int tAditivo, int tGasolina, int
tAlcool1, int tAlcool2) {
26.         ad = 0;
27.         gas = 0;
28.         al1 = 0;
29.         al2 = 0;
30.
31.         recebeAditivo(tAditivo);
32.         recebeGasolina(tGasolina);
33.         recebeAlcool((tAlcool1 + tAlcool2) / 2);
34.         defineSituacao();
35.     }
36.
37.     public void defineSituacao() {
38.         if (ad < MAX_ADITIVO * 0.25 || gas < MAX_GASOLINA * 0.25 ||
al1 < MAX_ALCOOL / 2 * 0.25
39.             || al2 < MAX_ALCOOL / 2 * 0.25) {
40.             situacao = SITUACAO.EMERGENCIA;
41.         } else if (ad < MAX_ADITIVO * 0.5 || gas < MAX_GASOLINA *
0.5 || al1 < MAX_ALCOOL / 2 * 0.5
42.             || al2 < MAX_ALCOOL / 2 * 0.5) {
43.             situacao = SITUACAO.SOBRAVISO;
44.         } else {
45.             situacao = SITUACAO.NORMAL;
46.         }
47.     }
```

```
48.
49.     public SITUACAO getSituacao() {
50.         return situacao;
51.     }
52.
53.     public int gettGasolina() {
54.         return gas;
55.     }
56.
57.     public int gettAditivo() {
58.         return ad;
59.     }
60.
61.     public int gettAlcool1() {
62.         return al1;
63.     }
64.
65.     public int gettAlcool2() {
66.         return al2;
67.     }
68.
69.     public int recebeAditivo(int qtdade) {
70.         if (qtdade < 0) {
71.             return -1;
72.         } else if (qtdade + ad >= MAX_ADITIVO) {
73.             ad = MAX_ADITIVO;
74.             return MAX_ADITIVO - ad;
75.         } else {
76.             ad = qtdade + ad;
77.             return qtdade;
78.         }
79.     }
80.
81.     public int recebeGasolina(int qtdade) {
82.         if (qtdade < 0) {
83.             return -1;
84.         } else if (qtdade + gas >= MAX_GASOLINA) {
85.             gas = MAX_GASOLINA;
86.             return MAX_GASOLINA - gas;
87.         } else {
88.             gas = qtdade + gas;
89.             return qtdade;
90.         }
91.     }
92.
93.     public int recebeAlcool(int qtdade) {
94.         if (qtdade < 0) {
95.             return -1;
96.         } else if (qtdade + al1 >= MAX_ALCOOL / 2) {
```

```

97.         al1 = MAX_ALCOOL / 2;
98.         al2 = MAX_ALCOOL / 2;
99.         return MAX_ALCOOL / 2 - al1;
100.        } else {
101.            al1 = qtdade + al1;
102.            al2 = qtdade + al2;
103.            return qtdade;
104.        }
105.    }
106.
107.    public int[] encomendaCombustivel(int qtdade, TIPOPOSTO
    tipoPosto) {
108.        if (qtdade <= 0)
109.            return new int[] { -7 };
110.
111.        switch (situacao) {
112.            case NORMAL: {
113.                if (!validaQtd(qtdade))
114.                    return new int[] { -21 };
115.                break;
116.            }
117.            case SOBRAVISO: {
118.                if (tipoPosto == TIPOPOSTO.COMUM) {
119.                    qtdade = qtdade / 2;
120.                    if (!validaQtd(qtdade))
121.                        return new int[] { -21 };
122.                } else {
123.                    if (!validaQtd(qtdade))
124.                        return new int[] { -21 };
125.                }
126.                break;
127.            }
128.            case EMERGENCIA: {
129.                if (tipoPosto == TIPOPOSTO.COMUM) {
130.                    return new int[] { -14 };
131.                } else {
132.                    if (!validaQtd(qtdade))
133.                        return new int[] { -21 };
134.                }
135.                break;
136.            }
137.            default:
138.                break;
139.        }
140.        atualizaTanque(qtdade);
141.        return new int[] { ad, gas, al1, al2 };
142.    }
143.
144.    private boolean validaQtd(int qtdade) {

```

```

145.         if ((qtdade * 0.7) > gas || ((qtdade * 0.25) / 2) >
    a11 || ((qtdade * 0.25) / 2) > a12 || (qtdade * 0.05) > a12)
146.             return false;
147.         return true;
148.     }
149.
150.     private void atualizaTanque(int qtdade) {
151.         ad -= qtdade * 0.05;
152.         a11 -= (qtdade * 0.25) / 2;
153.         a12 -= (qtdade * 0.25) / 2;
154.         gas -= qtdade * 0.7;
155.         defineSituacao();
156.     }
157. }
158.

```

4. Defeitos Encontrados/Corrigidos

Durante a realização dos testes, apenas um pequeno defeito foi encontrado e corrigido, sendo ele a impossibilidade de um recebimento de combustível (aditivo, gasolina ou álcool) ser feito com o valor 0.

```


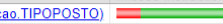


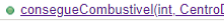



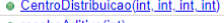

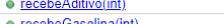
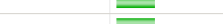


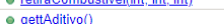





public int recebeAlcool(int qtdade) {
    if(qtdade < 1){ //Defeito aqui <0
        return -1;
    }
}

```

5. Análise de cobertura de código usando o “Code Coverage”.






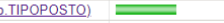
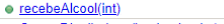

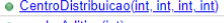



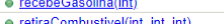

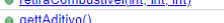
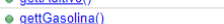




Após a implementação dos testes no projeto trocado com o outro grupo, os casos de teste obtiveram uma taxa de cobertura de código de 80%, tendo todos os métodos (e a classe) cobertos, porém com alguns blocos de decisões com casos não testados.

CentroDistribuicao

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
● encomendaCombustivel(int_CentroDistribuicao.TIPOPOSTO)		80%		72%	4	10	0
● defineSituacao()		100%		100%	0	7	0
● consegueCombustivel(int_CentroDistribuicao.TIPOPOSTO)		100%		66%	2	4	0
● recebeAlcool(int)		100%		100%	0	3	0
● CentroDistribuicao(int_int_int)		100%		100%	0	9	0
● recebeAditivo(int)		100%		100%	0	3	0
● recebeGasolina(int)		100%		100%	0	3	0
● retiraCombustivel(int_int_int)		100%	n/a	n/a	0	1	0
● gettAditivo()		100%	n/a	n/a	0	1	0
● gettGasolina()		100%	n/a	n/a	0	1	0
● gettAlcool1()		100%	n/a	n/a	0	1	0
● gettAlcool2()		100%	n/a	n/a	0	1	0
● gettSituacao()		100%	n/a	n/a	0	1	0
Total	20 of 387	94%	7 of 64	89%	6	45	4

Após essa análise inicial, os blocos não testados tiveram casos de teste devidamente desenvolvidos, assim como alguns casos não cobertos pelas técnicas pré-implementação, em seguida alcançando a taxa de cobertura de 100%.

CentroDistribuicao

Element	Missed Instructions	Cov.	Missed Branches	Cov.	Missed Cxty	Missed Lines	Missed Methods
● encomendaCombustivel(int_CentroDistribuicao.TIPOPOSTO)		100%		100%	0	10	0
● defineSituacao()		100%		100%	0	7	0
● consegueCombustivel(int_CentroDistribuicao.TIPOPOSTO)		100%		100%	0	4	0
● recebeAlcool(int)		100%		100%	0	3	0
● CentroDistribuicao(int_int_int)		100%		100%	0	9	0
● recebeAditivo(int)		100%		100%	0	3	0
● recebeGasolina(int)		100%		100%	0	3	0
● retiraCombustivel(int_int_int)		100%	n/a	n/a	0	1	0
● gettAditivo()		100%	n/a	n/a	0	1	0
● gettGasolina()		100%	n/a	n/a	0	1	0
● gettAlcool1()		100%	n/a	n/a	0	1	0
● gettAlcool2()		100%	n/a	n/a	0	1	0
● gettSituacao()		100%	n/a	n/a	0	1	0
Total	0 of 387	100%	0 of 64	100%	0	45	0

6. Casos de teste adicionais visando cobertura de blocos e de decisão

Portanto, após a primeira análise de code-coverage, testes adicionais foram implementados, com o objetivo de testar condicionais dentro de métodos que não foram exercitadas, assim como todas as possibilidades em condicionais múltiplas como && ou ||. Dessa forma todos os caminhos e blocos de decisão foram exercitados, mesmo que indiretamente nos casos iniciais, como propositalmente nos testes extras.

7. Código dos métodos de teste adicionais

```
@Test
public void encomendaRecusadoSituacaoSobreavisoQuantidadeMenor(){
    cd = new CentroDistribuicao(249, 10000, 1250, 1250);
    assertEquals(new int[]{-14}, cd.encomendaCombustivel(1,
TIPOPOSTO.COMUM));
}

@Test
public void encomendaRecusadoSituacaoEmergenciaQuantidadeMenor(){
    cd = new CentroDistribuicao(10, 10000, 1250, 1250);
    assertEquals(new int[]{-14}, cd.encomendaCombustivel(1,
TIPOPOSTO.ESTRATEGICO));
}

@Test
public void
encomendaAceitadoSituacaoSobreavisoQuantidadeMaiorPostoComum(){
    cd = new CentroDistribuicao(249, 10000, 1250, 1250);
    assertEquals(new int[]{246,9965,1243,1243},
cd.encomendaCombustivel(100, TIPOPOSTO.COMUM));
}

@Test
public void
encomendaAceitadoSituacaoEmergenciaQuantidadeMaiorPostoComum(){
    cd = new CentroDistribuicao(249, 10000, 1250, 1250);
    assertEquals(new int[]{244,9930,1237,1237},
cd.encomendaCombustivel(100, TIPOPOSTO.ESTRATEGICO));
}
```

```

    }

    @ParameterizedTest
    @MethodSource("casosTanquesVazios")
    public void entregaRecusadaTanquesVazios(int[] dados){
        cd = new CentroDistribuicao(dados[0], dados[1], dados[2],
        dados[3]);
        assertEquals(new int[]{-21}, cd.encomendaCombustivel(1000,
        TIPOPOSTO.ESTRATEGICO));
    }

    public static int[][] casosTanquesVazios() {
        return new int[][] { { 0, 10000, 1250, 1250 }, { 500, 0, 1250,
        1250 }, { 500, 10000, 0, 100 }, { 500, 10000, 100, 0 } };
    }

```

8. Eventuais defeitos adicionais encontrados

Nenhum defeito adicional foi encontrando durante a expansão dos casos de teste iniciais para os casos extras de 100% code coverage.

9. Análise das técnicas e ferramentas utilizadas para a melhoria do código gerado

O uso de técnicas de definição de testes anterior ao desenvolvimento do código é de extrema valia, pois essas são a “espinha dorsal” dos ditos TDD, ou seja, desenvolvimento orientado a testes. Por isso desenvolver essas técnicas trazem muitos benefícios não tão óbvios como o de “eliminar bugs”, como também garantem que o código está coeso, confiável e que se houver futuras alterações, a sua lógica se manterá como a dos requisitos iniciais. Outro fato importante de se mencionar é o uso de técnicas diversas para aumentar o nível de confiabilidade do código, como no nosso caso, os testes de valor limite com estrutural. No total foram realizados 72 testes, sendo que muitos foram através dos

testes parametrizados, que facilitaram e muito a criação dos testes. Além disso o uso do Jacoco para o code-coverage é muito interessante, pois garante que todo código implementado foi efetivamente testado, dessa forma deixando o código muito mais confiável, assim como demonstra indícios de código que nunca é executado.