TVS - Project Report

Group 19
Rodrigo Dias (99552)
Ana Margarida Almeida (102618)
Tiago Farinha (103327)

May 2025

Contents

1 Test cases for the `Client` class

We chose to employ the **Non-Modal Class Test Pattern**.

1.1 Class Invariant

$$\begin{array}{ll} & name.length() \leq 40 \\ \&\& & 0 \leq p \leq 200 \\ \&\& & 1 \leq terminals.size() \leq 9 \\ \&\& & friends.size() \leq 5 \times terminals.size() - 3 \\ \&\& & !friends.contains(this) \text{ (Condition 5)} \end{array}$$

1.2 Domain Matrix

Boundary			Input Test Values													
Variable	Condition	Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14
name.length()	≤ 40	ON	40													
		OFF		41												
	Typical	IN			1	2	3	4	5	6	7	8	9	10	11	12
points	≥ 0	ON			0											
		OFF				-1										
	≤ 200	ON					200									
		OFF						201								
	Typical	IN	20	25					30	35	40	45	55	60	65	70
#terminals	≥ 1	ON							1							
		OFF								0						
	≤ 9	ON									9					
		OFF										10				
	Typical	IN	2	3	4	5	6	7					8	2	3	4
#friends	$\leq 5 * \#\text{terminals} - 3$	ON											37			
		OFF												8		
	Typical	IN	0	1	2	3	4	5	1	0	6	7			11	16
friends	!friends.contains(this)	ON													T	
		OFF														F
	Typical	IN	T	T	T	T	T	T	T	T	T	T	T	T		
Expected Result			✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗	✓	✗

1.3 Message Selection

To allow code coverage, all methods of class `Client` can and should be used for a specified test battery, with either a define-use or random strategy, or a mix of both, since the available methods are simply a constructor and getters/setters, hence comparison of an OUT vector from accessors and the test inputs is trivial.

2 Test case battery for class `Client`

```java
import static org.testng.Assert.assertEquals;
import static org.testng.Assert.assertThrows;
import static org.testng.Assert.assertTrue;

import java.util.ArrayList;
import java.util.List;

import org.testng.annotations.Test;

@Test
public class TestClient {

    private List<Terminal> generateTerminals(int amount) {
        List<Terminal> terminals = new ArrayList<>();
        for (int i = 1; i <= amount; i++) {
            terminals.add(new Terminal(Integer.toString(i)))
                ;
        }

        return terminals;
    }

    private List<Client> generateFriends(int amount) {
        List<Client> friends = new ArrayList<>();
        for (int i = 1; i <= amount; i++) {
            friends.add(new Client("0" + Integer.toString(i)
                , 1, new Terminal(Integer.toString(i + 10))))
                ;
        }

        return friends;
    }

    // Test 1
    public void givenValidNameWhenCreatingThenCreate() {
        String name = "A".repeat(40);
        Terminal terminal1 = new Terminal("1");
        Terminal terminal2 = new Terminal("2");

        Client client = new Client(name, 0, terminal1);
        client.addTerminal(terminal2);

        assertEquals(client.getName().length(), 40);
        assertEquals(client.getPoints(), 20);
        assertEquals(client.numberOfTerminals(), 2);
        assertTrue(!client.hasFriend(client));
    }

    // Test 2
    public void
        givenInvalidNameWhenUpdatingNameThenThrowException()
        {
         String name = "A".repeat(41);
         int points = 25;
         Terminal terminal1 = new Terminal("1");
         Terminal terminal2 = new Terminal("2");
         Client client = new Client("Valid Name", 1,
             terminal1);
         client.updatePoints(points);
         client.addTerminal(terminal2);

         assertThrows(InvalidOperationException.class, () ->
             client.updateName(name));
    }

    // Test 3
    public void
        givenLowestValidPointsWhenUpdatingThenUpdatePoints()
        {
         String name = "A";
         int points = 0;
         List<Terminal> terminals = generateTerminals(4);
         Client client = new Client(name, 2, terminals.get(0)
             );
         for (int i = 1; i < terminals.size(); i++) {
             client.addTerminal(terminals.get(i));
         }
         for (Client friend : generateFriends(2)) {
             client.addFriend(friend);
         }

         client.updatePoints(points);

         assertEquals(client.getPoints(), 0);
    }

    // Test 4
    public void
        givenInvalidPointsBelowLimitWhenUpdatingThenThrowException
        () {
         String name = "AB";
         int points = -1;
         List<Terminal> terminals = generateTerminals(5);
         Client client = new Client(name, 3, terminals.get(0)
             );
         for (int i = 1; i < terminals.size(); i++) {
             client.addTerminal(terminals.get(i));
         }
         for (Client friend : generateFriends(3)) {
             client.addFriend(friend);
         }

         assertThrows(InvalidOperationException.class, () ->
             client.updatePoints(points));
    }

    // Test 5
    public void
        givenGreatestValidPointsWhenUpdatingThenUpdatePoints
        () {
         String name = "ABC";
         int points = 200;
         List<Terminal> terminals = generateTerminals(4);
         Client client = new Client(name, 4, terminals.get(0)
             );
         for (int i = 1; i < terminals.size(); i++) {
             client.addTerminal(terminals.get(i));
         }
         for (Client friend : generateFriends(2)) {
             client.addFriend(friend);
         }

         client.updatePoints(points);

         assertEquals(client.getPoints(), 200);
    }

    // Test 6
    public void
        givenInvalidPointsAboveLimitWhenUpdatingThenThrowException
        () {
         String name = "ABCD";
         int points = 201;
         List<Terminal> terminals = generateTerminals(5);
         Client client = new Client(name, 5, terminals.get(0)
             );
         for (int i = 1; i < terminals.size(); i++) {
             client.addTerminal(terminals.get(i));
         }
         for (Client friend : generateFriends(3)) {
             client.addFriend(friend);
         }

         assertThrows(InvalidOperationException.class, () ->
             client.updatePoints(points));
    }

    // Test 7
    public void
        givenValidAmountOfTerminalsWhenAddingTerminalThenAddTerminals
        () {
         String name = "ABCDE";
         int points = 30;
         Terminal terminal = new Terminal("1");
         Client friend = new Client("Friend", 100, new
             Terminal("100"));

         Client client = new Client(name, 6, terminal);
         client.updatePoints(points);
         client.addFriend(friend);

         assertEquals(client.numberOfTerminals(), 1);
         assertEquals(client.getPoints(), points);
         assertTrue(client.hasFriend(friend));
    }

    // Test 8
    public void
        givenInvalidAmountOfTerminalsWhenAddingTerminalThenThrowException
        () {
         String name = "ABCDEF";
         Terminal terminal = null;

         assertThrows(InvalidOperationException.class, () ->
             new Client(name, 7, terminal));
    }

}
```

3 Test cases for the `Terminal` class

In order to test the Terminal class, we will use a Modal Class Test Pattern.

3.1 Generate a state model for CUT

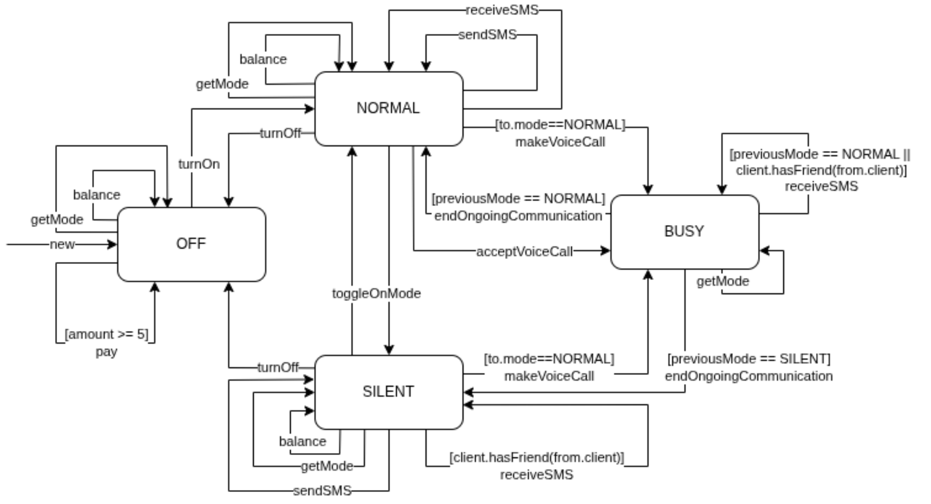

Figure 1: Initial State Model for CUT

3.2 Full expansion of conditional transition variants

3.2.1 Develop a Truth Table for each conditional transition

State	Message	Condition	Next State
OFF	pay	Pre: amount ≥ 5	OFF
NORMAL	makeVoiceCall	Pre: to.MODE == NORMAL	BUSY
SILENT	makeVoiceCall	Pre: to.MODE == NORMAL	BUSY
SILENT	receiveSMS	Pre: client.hasFriend(from.client)	SILENT
BUSY	receiveSMS	Pre: previousMode == NORMAL || client.hasFriend(from.client)	BUSY
BUSY	endOngoingCommunication	Pre: previousMode == NORMAL	NORMAL
BUSY	endOngoingCommunication	Pre: previousMode == SILENT	SILENT

Table 1: Truth Table

- *pay* in OFF generates an additional transition when amount < 5

- $makeVoiceCall$ in NORMAL generates an additional transition when to.MODE != NORMAL
- $makeVoiceCall$ in SILENT generates an additional transition when to.MODE != NORMAL
- $receiveSMS$ in SILENT generates an additional transition when !client.hasFriend(from.client)
- $receiveSMS$ in BUSY generates an additional transition when previousMode != NORMAL and !client.hasFriend(from.client)

3.2.2 Update state diagram with the additional transitions

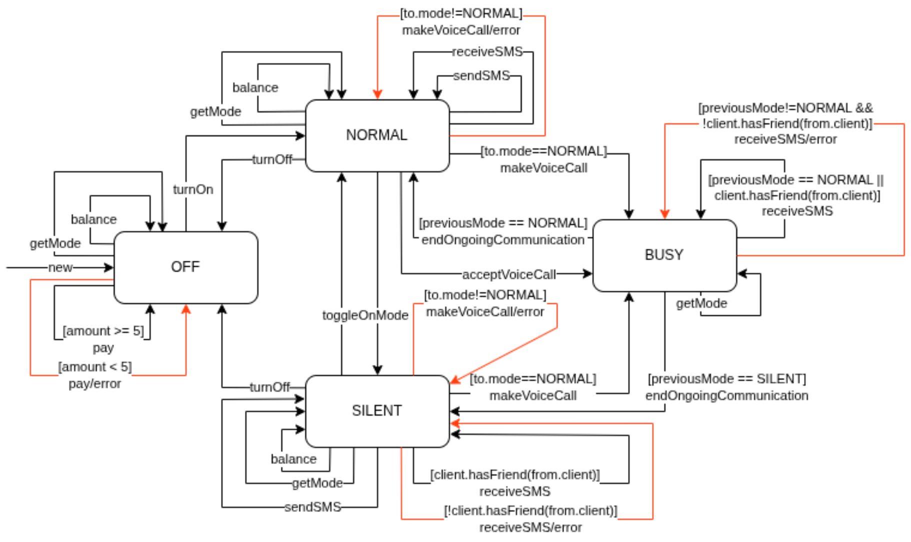

Figure 2: Updated State Model for CUT

3.3 Generate transition tree

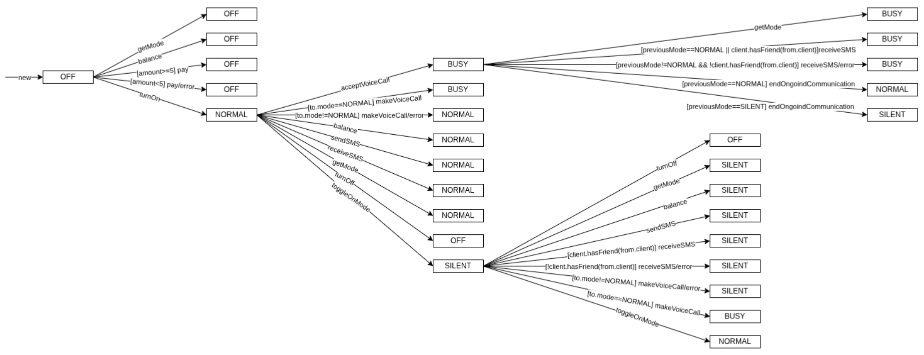
new
OFF
getMode
OFF
balance
OFF
[amount>=5] pay
OFF
[amount<5] pay/error
OFF
turnOn
NORMAL
acceptVoiceCall
BUSY
[to.mode==NORMAL] makeVoiceCall
BUSY
[to.mode!=NORMAL] makeVoiceCall/error
NORMAL
balance
NORMAL
sendSMS
NORMAL
receiveSMS
NORMAL
getMode
NORMAL
turnOff
OFF
toggleOnMode
SILENT
getMode
BUSY
[previousMode==NORMAL || client.hasFriend(from.client)]receiveSMS
BUSY
[previousMode!=NORMAL && !client.hasFriend(from.client)] receiveSMS/error
BUSY
[previousMode==NORMAL] endOngoindCommunication
NORMAL
[previousMode==SILENT] endOngoindCommunication
SILENT
turnOff
OFF
getMode
SILENT
balance
SILENT
sendSMS
SILENT
[client.hasFriend(from.client)] receiveSMS
SILENT
[!client.hasFriend(from.client)] receiveSMS/error
SILENT
[to.mode!=NORMAL] makeVoiceCall/error
SILENT
[to.mode==NORMAL] makeVoiceCall
BUSY
toggleOnMode
NORMAL

Figure 3: Transition Tree

3.4 Generate Conformance Test Suite

Run	Level 1	Level 2	Level 3	Level 4	Expected Terminal State	Error
1	new	-	-	-	OFF	No
2	new	getMode	-	-	OFF	No
3	new	balance	-	-	OFF	No
4	new	[amount≥5] pay	-	-	OFF	No
5	new	[amount<5] pay	-	-	OFF	Yes
6	new	turnOn	-	-	NORMAL	No
7	new	turnOn	acceptVoiceCall	-	BUSY	No
8	new	turnOn	[to.mode==NORMAL] makeVoiceCall	-	BUSY	No
9	new	turnOn	[to.mode!=NORMAL] makeVoiceCall	-	NORMAL	Yes
10	new	turnOn	balance	-	NORMAL	No
11	new	turnOn	sendSMS	-	NORMAL	No
12	new	turnOn	receiveSMS	-	NORMAL	No
13	new	turnOn	getMode	-	NORMAL	No
14	new	turnOn	turnOff	-	OFF	No
15	new	turnOn	toggleOnMode	-	SILENT	No
16	new	turnOn	acceptVoiceCall	getMode	BUSY	No
17	new	turnOn	acceptVoiceCall	[previousMode==NORMAL || client.hasFriend(from.client)] receiveSMS	BUSY	No
18	new	turnOn	acceptVoiceCall	[previousMode!=NORMAL && !client.hasFriend(from.client)] receiveSMS	BUSY	Yes
19	new	turnOn	acceptVoiceCall	[previousMode==NORMAL] endOngoingCommunication	NORMAL	No
20	new	turnOn	acceptVoiceCall	[previousMode==SILENT] endOngoingCommunication	SILENT	No
21	new	turnOn	toggleOnMode	turnOff	OFF	No
22	new	turnOn	toggleOnMode	getMode	SILENT	No
23	new	turnOn	toggleOnMode	balance	SILENT	No
24	new	turnOn	toggleOnMode	sendSMS	SILENT	No
25	new	turnOn	toggleOnMode	[client.hasFriend(from.client)] receiveSMS	SILENT	No
26	new	turnOn	toggleOnMode	[!client.hasFriend(from.client)] receiveSMS	SILENT	Yes
27	new	turnOn	toggleOnMode	[to.mode!=NORMAL] makeVoiceCall	SILENT	Yes
28	new	turnOn	toggleOnMode	[to.mode==NORMAL] makeVoiceCall	BUSY	No
29	new	turnOn	toggleOnMode	toggleOnMode	NORMAL	No

Table 2: Initial Conformance Test Suite

3.5 Develop test data for each path using Invariant Boundaries

Condition	**On Point**	**Off Point**
amount ≥ 5	5	4
amount < 5	5	4

Table 3: *pay* in OFF

Condition	**On Point**	**Off Point**
to.mode==NORMAL	NORMAL	SILENT
to.mode!=NORMAL	NORMAL	OFF

Table 4: $makeVoiceCall$ in NORMAL

Condition	**On Point**	**Off Point**
previousMode==NORMAL	NORMAL	SILENT
client.hasFriend(from.client)	true	false
previousMode!=NORMAL	NORMAL	SILENT
!client.hasFriend(from.client)	false	true

Table 5: $receiveSMS$ in BUSY

Condition	**On Point**	**Off Point**
previousMode==NORMAL	NORMAL	SILENT
previousMode==SILENT	SILENT	NORMAL

Table 6: $endOngoingCommunication$ in BUSY

Condition	**On Point**	**Off Point**
client.hasFriend(from.client)	true	false
!client.hasFriend(from.client)	false	true

Table 7: *receiveSMS* in SILENT

Condition	**On Point**	**Off Point**
to.mode==NORMAL	NORMAL	OFF
to.mode!=NORMAL	NORMAL	BUSY

Table 8: *makeVoiceCall* in SILENT

3.6 Build Transition Table

VT = Valid Transition, **PSP** = Possible sneak path, **?** = Conditional Transition

Event \ State	**OFF**	**NORMAL**	**BUSY**	**SILENT**
getMode	VT	VT	VT	VT
balace	VT	VT	PSP	VT
pay	?	PSP	PSP	PSP
turnOn	VT	PSP	PSP	PSP
turnOff	PSP	VT	PSP	VT
toggleOnMode	PSP	VT	PSP	VT
makeVoiceCall	PSP	?	PSP	?
acceptVoiceCall	PSP	VT	PSP	PSP
sendSMS	PSP	VT	PSP	VT
receiveSMS	PSP	VT	?	?
endOngoingCommunication	PSP	PSP	?	PSP

Table 9: Transition Table

3.7 Develop Sneak Path Test Suite

Run	Level 1	Level 2	Level 3	Level 4	Expected Terminal State	Error
30	new	turnOff	-	-	OFF	Yes
31	new	toggleOnMode	-	-	OFF	Yes
32	new	makeVoiceCall	-	-	OFF	Yes
33	new	acceptVoiceCall	-	-	OFF	Yes
34	new	sendSMS	-	-	OFF	Yes
35	new	receiveSMS	-	-	OFF	Yes
36	new	endOngoingCommunication	-	-	OFF	Yes
37	new	turnOn	pay	-	NORMAL	Yes
38	new	turnOn	turnOn	-	NORMAL	Yes
39	new	turnOn	endOngoingCommunication	-	NORMAL	Yes
40	new	turnOn	acceptVoiceCall	balance	BUSY	Yes
41	new	turnOn	acceptVoiceCall	pay	BUSY	Yes
42	new	turnOn	acceptVoiceCall	turnOn	BUSY	Yes
43	new	turnOn	acceptVoiceCall	turnOff	BUSY	Yes
44	new	turnOn	acceptVoiceCall	toggleOnMode	BUSY	Yes
45	new	turnOn	acceptVoiceCall	makeVoiceCall	BUSY	Yes
46	new	turnOn	acceptVoiceCall	acceptVoiceCall	BUSY	Yes
47	new	turnOn	acceptVoiceCall	sendSMS	BUSY	Yes
48	new	turnOn	toggleOnMode	pay	SILENT	Yes
49	new	turnOn	toggleOnMode	turnOn	SILENT	Yes
50	new	turnOn	toggleOnMode	acceptVoiceCall	SILENT	Yes
51	new	turnOn	toggleOnMode	endOngoingCommunication	SILENT	Yes

Table 10: To be added to Conformance Test Suite

3.8 Final Conformance Test Suites

Run	Level 1	Level 2	Level 3	Level 4	Expected Terminal State	Error
1	new	-	-	-	OFF	No
2	new	getMode	-	-	OFF	No
3	new	balance	-	-	OFF	No
4	new	[amount≥5] pay	-	-	OFF	No
5	new	[amount<5] pay	-	-	OFF	Yes
6	new	turnOn	-	-	NORMAL	No
7	new	turnOn	acceptVoiceCall	-	BUSY	No
8	new	turnOn	[to.mode==NORMAL] makeVoiceCall	-	BUSY	No
9	new	turnOn	[to.mode!=NORMAL] makeVoiceCall	-	NORMAL	Yes
10	new	turnOn	balance	-	NORMAL	No
11	new	turnOn	sendSMS	-	NORMAL	No
12	new	turnOn	receiveSMS	-	NORMAL	No
13	new	turnOn	getMode	-	NORMAL	No
14	new	turnOn	turnOff	-	OFF	No

15	new	turnOn	toggleOnMode	-	SILENT	No
16	new	turnOn	acceptVoiceCall	getMode	BUSY	No
17	new	turnOn	acceptVoiceCall	[previousMode==NORMAL || client.hasFriend(from.client)] receiveSMS	BUSY	No
18	new	turnOn	acceptVoiceCall	[previousMode!=NORMAL && !client.hasFriend(from.client)] receiveSMS	BUSY	Yes
19	new	turnOn	acceptVoiceCall	[previousMode==NORMAL] endOngoingCommunication	NORMAL	No
20	new	turnOn	acceptVoiceCall	[previousMode==SILENT] endOngoingCommunication	SILENT	No
21	new	turnOn	toggleOnMode	turnOff	OFF	No
22	new	turnOn	toggleOnMode	getMode	SILENT	No
23	new	turnOn	toggleOnMode	balance	SILENT	No
24	new	turnOn	toggleOnMode	sendSMS	SILENT	No
25	new	turnOn	toggleOnMode	[client.hasFriend(from.client)] receiveSMS	SILENT	No
26	new	turnOn	toggleOnMode	[!client.hasFriend(from.client)] receiveSMS	SILENT	Yes
27	new	turnOn	toggleOnMode	[to.mode!=NORMAL] makeVoiceCall	SILENT	Yes
28	new	turnOn	toggleOnMode	[to.mode==NORMAL] makeVoiceCall	BUSY	No
29	new	turnOn	toggleOnMode	toggleOnMode	NORMAL	No
30	new	turnOff	-	-	OFF	Yes
31	new	toggleOnMode	-	-	OFF	Yes
32	new	makeVoiceCall	-	-	OFF	Yes
33	new	acceptVoiceCall	-	-	OFF	Yes
34	new	sendSMS	-	-	OFF	Yes
35	new	receiveSMS	-	-	OFF	Yes
36	new	endOngoingCommunication	-	-	OFF	Yes
37	new	turnOn	pay	-	NORMAL	Yes
38	new	turnOn	turnOn	-	NORMAL	Yes
39	new	turnOn	endOngoingCommunication	-	NORMAL	Yes
40	new	turnOn	acceptVoiceCall	balance	BUSY	Yes
41	new	turnOn	acceptVoiceCall	pay	BUSY	Yes
42	new	turnOn	acceptVoiceCall	turnOn	BUSY	Yes
43	new	turnOn	acceptVoiceCall	turnOff	BUSY	Yes
44	new	turnOn	acceptVoiceCall	toggleOnMode	BUSY	Yes
45	new	turnOn	acceptVoiceCall	makeVoiceCall	BUSY	Yes
46	new	turnOn	acceptVoiceCall	acceptVoiceCall	BUSY	Yes
47	new	turnOn	acceptVoiceCall	sendSMS	BUSY	Yes
48	new	turnOn	toggleOnMode	pay	SILENT	Yes
49	new	turnOn	toggleOnMode	turnOn	SILENT	Yes
50	new	turnOn	toggleOnMode	acceptVoiceCall	SILENT	Yes
51	new	turnOn	toggleOnMode	endOngoingCommunication	SILENT	Yes

Table 11: Final Conformance Test Suites

4 Test cases for the `computeCost()` method

In order to create the test cases for the `computeCost()` method we are going to use the combinational functional test design pattern.

First, we need to model the method using a decision tree.

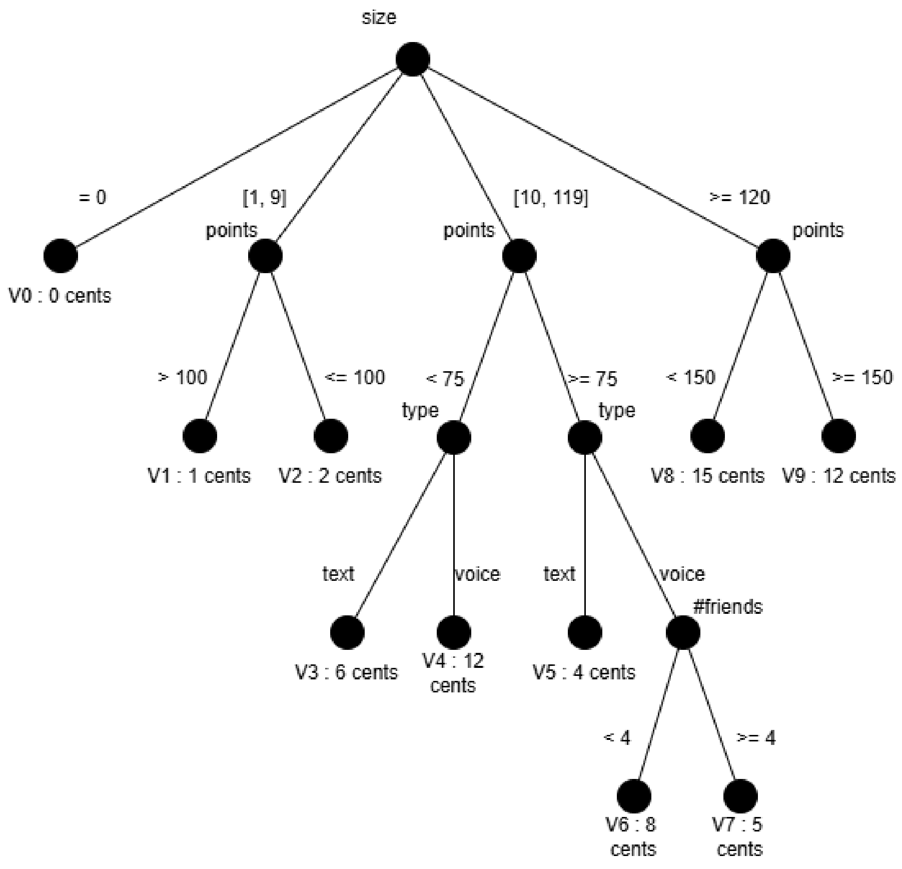

Figure 4: `computeCost()` decision tree.

This gives us the following variants:

- V_0 : size $= 0$: 0 cents
- V_1 : size $> 0 \wedge$ size $< 10 \wedge$ points > 100 : 1 cents
- V_2 : size $> 0 \wedge$ size $< 10 \wedge$ points ≤ 100 : 2 cents
- V_3 : size $\geq 10 \wedge$ size $< 120 \wedge$ points $< 75 \wedge$ type $=$ text : 6 cents
- V_4 : size $\geq 10 \wedge$ size $< 120 \wedge$ points $< 75 \wedge$ type $=$ voice : 12 cents

- V_5 : size $\geq 10 \land$ size $< 120 \land$ points $\geq 75 \land$ type = text : 4 cents
- V_6 : size $\geq 10 \land$ size $< 120 \land$ points $\geq 75 \land$ type = voice $\land$ #friends < 4 : 8 cents
- V_7 : size $\geq 10 \land$ size $< 120 \land$ points $\geq 75 \land$ type = voice $\land$ #friends ≥ 4 : 5 cents
- V_8 : size $\geq 120 \land$ points < 150 : 15 cents
- V_9 : size $\geq 120 \land$ points ≥ 150 : 12 cents

The next step is to build a domain matrix for each variant.

V_0			1	-	-
size	$= 0$	On	0		
		Off		1	
		Off			-1
points	In		1	2	3
type	In		text	voice	text
#friends	In		0	1	2
Expected result			0	V_2	Impossible

Table 12: V_0 domain matrix.

V_1			-	2	-	3	-	4
size	> 0	On	0					
		Off		1				
	< 10	On			10			
		Off				9		
	In						5	6
points	> 100	On					100	
		Off						101
	In		110	120	130	140		
type	In		text	voice	text	voice	text	voice
#friends	In		0	1	2	3	4	5
Expected result			V_0	1	V_5	1	V_2	1

Table 13: V_1 domain matrix.

V_2			-	5	-	6	7	-
size	> 0	On	0					
		Off		1				
	< 10	On			10			
		Off				9		
	In						5	6
points	≤ 100	On					100	
		Off						101
	In		50	60	70	80		
type	In		text	voice	text	voice	text	voice
#friends	In		0	1	2	3	4	5
Expected result			V_0	2	V_3	2	2	V_1

Table 14: V_2 domain matrix.

V_3			8	-	-	9	-	10	11	-
size	≥ 10	On	10							
		Off		9						
	< 120	On			120					
		Off				119				
	In						65	66	67	68
points	< 75	On					75			
		Off						74		
	In		5	15	25	35			45	55
type	= text	On							text	
		Off								voice
	In		text	text	text	text	text	text		
#friends	In		0	1	2	3	4	5	6	7
Expected result			6	V_2	V_8	6	V_5	6	6	V_4

Table 15: V_3 domain matrix.

V_4			12	-	-	13	-	14	15	-
size	≥ 10	On	10							
		Off		9						
	< 120	On			120					
		Off				119				
	In						65	66	67	68
points	< 75	On					75			
		Off						74		
	In		5	15	25	35			45	55
type	= voice	On							voice	
		Off								text
	In		voice	voice	voice	voice	voice	voice		
#friends	In		0	1	2	3	4	5	6	7
Expected result			12	V_2	V_8	12	V_7	12	12	V_3

Table 16: V_4 domain matrix.

V_5			16	-	-	17	18	-	19	-
size	≥ 10	On	10							
		Off		9						
	< 120	On			120					
		Off				119				
	In						65	66	67	68
points	≥ 75	On					75			
		Off						74		
	In		80	90	100	110			120	130
type	= text	On							text	
		Off								voice
	In		text	text	text	text	text	text		
#friends	In		0	1	2	3	4	5	6	7
Expected result			4	V_2	V_8	4	4	V_3	4	V_7

Table 17: V_5 domain matrix.

V_6			20	-	-	21	22	-	23	-	-	24
size	≥ 10	On	10									
		Off		9								
	< 120	On			120							
		Off				119						
	In						65	66	67	68	69	70
points	≥ 75	On					75					
		Off						74				
	In		80	90	100	110			120	130	140	150
type	= voice	On							voice			
		Off								text		
	In		voice	voice	voice	voice	voice	voice			voice	voice
#friends	< 4	On									4	
		Off										3
	In		0	1	2	0	1	2	0	1		
Expected result			8	V_2	V_8	8	8	V_4	8	V_5	V_7	8

Table 18: V_6 domain matrix.

V_7			25	-	-	26	27	-	28	-	29	-
size	≥ 10	On	10									
		Off		9								
	< 120	On			120							
		Off				119						
	In						65	66	67	68	69	70
points	≥ 75	On					75					
		Off						74				
	In		80	90	100	110			120	130	140	150
type	= voice	On							voice			
		Off								text		
	In		voice	voice	voice	voice	voice	voice			voice	voice
#friends	≥ 4	On									4	
		Off										3
	In		6	7	8	9	10	11	12	13		
Expected result			5	V_2	V_8	5	5	V_4	5	V_5	5	V_6

Table 19: V_7 domain matrix.

V_8			30	-	-	31
size	≥ 120	On	120			
		Off		119		
	In				150	160
points	< 150	On			150	
		Off				149
	In		75	76		
type	In		text	voice	text	voice
#friends	In		0	1	2	3
Expected result			15	V_6	V_9	15

Table 20: V_8 domain matrix.

V_9			32	-	33	-
size	≥ 120	On	120			
		Off		119		
	In				150	160
points	< 150	On			150	
		Off				149
	In		170	180		
type	In		text	voice	text	voice
#friends	In		0	1	2	3
Expected result			12	V_6	12	V_8

Table 21: V_9 domain matrix.

5 Test cases for the `removeTerminal()` method

In order to create the test cases for the `removeTerminal()` method we are going to use the category partition test design pattern.

5.1 Identify all functions

The `removeTerminal()` method has the following functions:

1. removes *terminal* from the client's list of terminals if possible and return true. If it is not possible return false.
2. if *terminal* is removed, updates size of the client's list of terminals.
3. if removing *terminal* would put the invoked client in an invalid state, throw *InvalidOperationException* exception.

5.2 Identify input and output parameters

Function	Input	Output
1. removes *terminal*	*terminal*, list of terminals, *terminal*.balance()	list of terminals, return value
2. update size	*terminal*, list of terminals	size
3. throw exception	#friends, *terminal*, list of terminals	exception

Table 22: input and output for each function.

5.3 Identify categories for each input parameter

Parameter	Category
terminal	invalid
	terminal $\in$ list of terminals
	terminal $\notin$ list of terminals
list of terminals	singleton
	full
	holding
terminal.balance()	valid
	invalid
#friends	valid

Table 23: categories for each parameter.

5.4 Partition each category into choices

Parameter	**Category**	**Choices**
terminal	invalid	null
	terminal $\in$ list of terminals	t_1
	terminal $\notin$ list of terminals	$t_\times$
list of terminals	singleton	$\{t_1\}$
	full	$\{t_1, \ldots, t_9\}$
	holding	$\{t_1, \ldots, t_n\}, n \in [2, 8]$
terminal.balance()	valid	0, 5
	invalid	-1, -10
#friends	valid	2, 3, 22, 42

Table 24: choices for each category.

5.5 Identify constraints on choices

- If *terminal* is null, the behavior of `removeTerminal()` is always the same.
- If *terminal* $\notin$ list of terminals, the behavior of `removeTerminal()` is always the same.
- If *terminal*.balance() is negative, the behavior of `removeTerminal()` is always the same.

5.6 Generate test cases by enumerating all choices

Test case	**terminal**	**list of terminals**	**terminal.balance()**	**#friends**
1	null	$n = \text{random}(2, 8)$	0	2
2	$t_\times$	$n = \text{random}(2, 8)$	0	2
3	t_1	$n = \text{random}(2, 8)$	-1	2
4	t_1	$n = \text{random}(2, 8)$	-10	2
5	t_1	$\{t_1, t_2\}$	0	2
6	t_1	$\{t_1, t_2\}$	0	3
7	t_1	$\{t_1, t_2\}$	5	2
8	t_1	$\{t_1, t_2\}$	5	3
9	t_1	$\{t_1\}$	0	2
10	t_1	$\{t_1\}$	0	3
11	t_1	$\{t_1\}$	5	2
12	t_1	$\{t_1\}$	5	3
13	t_1	$\{t_1, \ldots, t_9\}$	0	2
14	t_1	$\{t_1, \ldots, t_9\}$	0	42
15	t_1	$\{t_1, \ldots, t_9\}$	5	2
16	t_1	$\{t_1, \ldots, t_9\}$	5	42
17	t_1	$\{t_1, \ldots, t_5\}$	0	2
18	t_1	$\{t_1, \ldots, t_5\}$	0	22
19	t_1	$\{t_1, \ldots, t_5\}$	5	2
20	t_1	$\{t_1, \ldots, t_5\}$	5	22

Table 25: generated test cases.

5.7 Develop expected values for each test case

	Inputs				**Expected output**			
Test case	**terminal**	**list of terminals**	**terminal.balance()**	**#friends**	**return value**	**list**	**size**	**exception**
1	null	$n = \text{random}(2,8)$	0	2	false	-	n	-
2	$t_\times$	$n = \text{random}(2,8)$	0	2	false	-	n	-
3	t_1	$n = \text{random}(2,8)$	-1	2	false	-	n	-
4	t_1	$n = \text{random}(2,8)$	-10	2	false	-	n	-
5	t_1	$\{t_1, t_2\}$	0	2	true	$\{t_2\}$	1	-
6	t_1	$\{t_1, t_2\}$	0	3	-	-	2	throw
7	t_1	$\{t_1, t_2\}$	5	2	true	$\{t_2\}$	1	-
8	t_1	$\{t_1, t_2\}$	5	3	-	-	2	throw
9	t_1	$\{t_1\}$	0	2	-	-	1	throw
10	t_1	$\{t_1\}$	0	3	-	-	1	throw
11	t_1	$\{t_1\}$	5	2	-	-	1	throw
12	t_1	$\{t_1\}$	5	3	-	-	1	throw
13	t_1	$\{t_1, \ldots, t_9\}$	0	2	true	$\{t_2, \ldots, t_9\}$	8	-
14	t_1	$\{t_1, \ldots, t_9\}$	0	42	-	-	9	throw
15	t_1	$\{t_1, \ldots, t_9\}$	5	2	true	$\{t_2, \ldots, t_9\}$	8	-
16	t_1	$\{t_1, \ldots, t_9\}$	5	42	-	-	9	throw
17	t_1	$\{t_1, \ldots, t_5\}$	0	2	true	$\{t_2, \ldots, t_5\}$	4	-
18	t_1	$\{t_1, \ldots, t_5\}$	0	22	-	-	5	throw
19	t_1	$\{t_1, \ldots, t_5\}$	5	2	true	$\{t_2, \ldots, t_5\}$	4	-
20	t_1	$\{t_1, \ldots, t_5\}$	5	22	-	-	5	throw

Table 26: output for each test cases.