INSTITUTO SUPERIOR TÉCNICO - UL

SOFTWARE TESTING AND VALIDATION

HEALTH CLUB

REPORT - GROUP 20

Authors: *IST ID:*
Vasco FARIA   89559
Pedro PEREIRA   90766

20 of April of 2021

Contents

1 Member Class Scope Test

Test Pattern used: Conformance Testing

We chose this pattern because GroupClass is a modal class (only depends on its contents and its history) and has a fixed constraints on method sequences. The objective of the test suite is to find out which message sequences and state changes may originate error.

We started by building a state machine of the class:
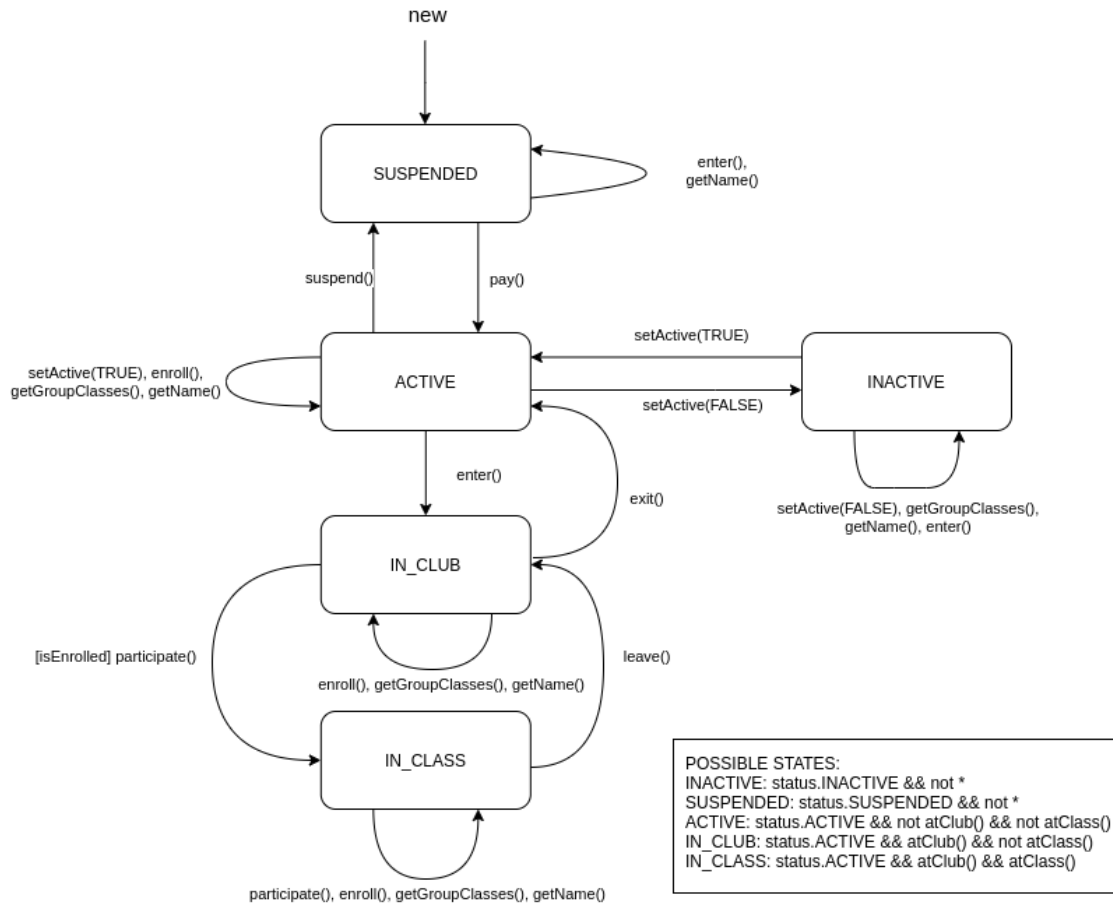

Figure 1: State model for Member class

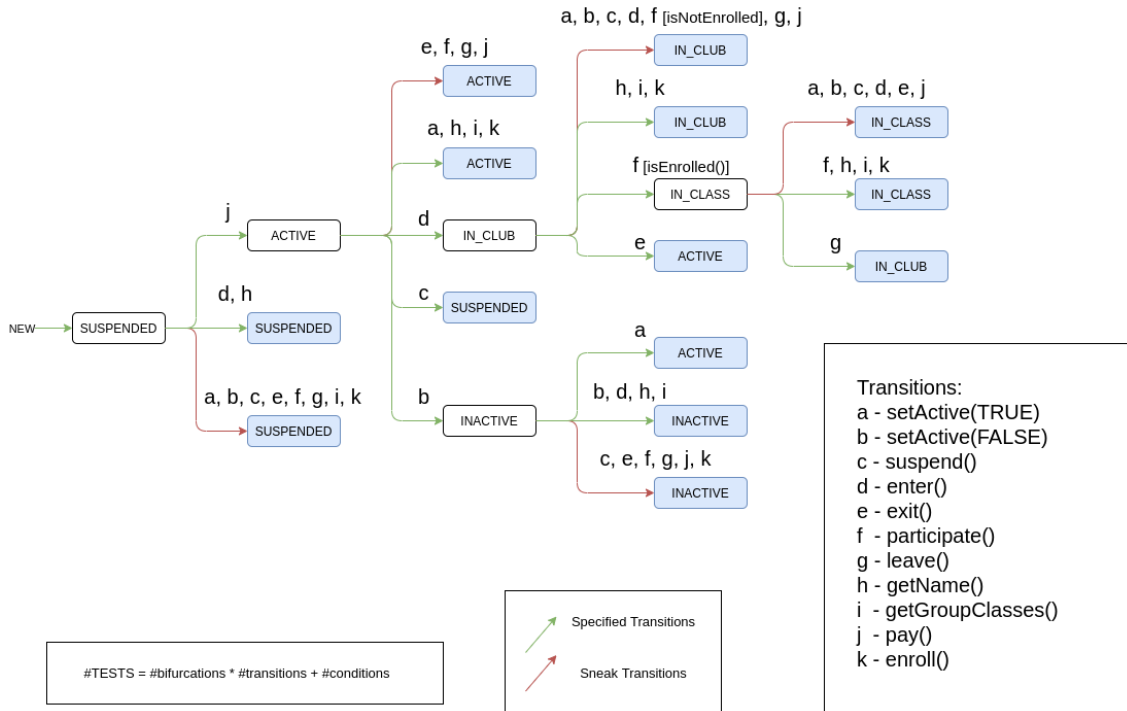From the state machine we built a transition tree:

Figure 2: Transition tree for Member class

And by navigating the branches on the transition tree, we designed the conformance test suite. We discovered the sneak paths through Fig. 3. Tests 1-27 represent the specified behaviour of the class. And tests 28-58 represent the sneak path tests.

Events	States				
	Suspended	Active	Inactive	In Club	In Class
setActive(TRUE)	PSP	✓	✓	PSP	PSP
setActive(FALSE)	PSP	✓	✓	PSP	PSP
suspend()	PSP	✓	PSP	PSP	PSP
enter()	✓	✓	✓	PSP	PSP
exit()	PSP	PSP	PSP	✓	PSP
participate()	✓	PSP	PSP	?	✓
leave()	PSP	PSP	PSP	PSP	✓
getName()	✓	✓	✓	✓	✓
getGroupClasses()	PSP	✓	✓	✓	✓
pay()	✓	✓	PSP	PSP	PSP
enroll()	PSP	✓	PSP	✓	✓

Figure 3: Possible Sneak Path Transition Table

Run	Test Run/Event Path					Expected Terminal State	Exceptions
	lvl1	lvl2	lvl3	lvl4	lvl5		
1	new					SUSPENDED	
2	new	pay()				ACTIVE	
3	new	enter()				SUSPENDED	
4	new	participate()				SUSPENDED	
5	new	getName()				SUSPENDED	
6	new	pay()	setActive(True)			ACTIVE	
7	new	pay()	getName()			ACTIVE	
8	new	pay()	getGroupClasses()			ACTIVE	
9	new	pay()	enroll()			ACTIVE	
10	new	pay()	enter()			IN_CLUB	
11	new	pay()	suspend()			SUSPENDED	
12	new	pay()	setActive(False)			INACTIVE	
13	new	pay()	enter()	getName()		IN_CLUB	
14	new	pay()	enter()	getGroupClasses()		IN_CLUB	
15	new	pay()	enter()	enroll()		IN_CLUB	
16	new	pay()	enter()	[isEnrolled] participate()		IN_CLASS	
17	new	pay()	enter()	exit()		ACTIVE	
18	new	pay()	setActive(False)	setActive(True)		ACTIVE	
19	new	pay()	setActive(False)	setActive(False)		INACTIVE	
20	new	pay()	setActive(False)	enter()		INACTIVE	
21	new	pay()	setActive(False)	getName()		INACTIVE	
22	new	pay()	setActive(False)	getGroupClasses()		INACTIVE	
23	new	pay()	enter()	[isEnrolled] participate()	participate()	IN_CLASS	
24	new	pay()	enter()	[isEnrolled] participate()	getName()	IN_CLASS	
25	new	pay()	enter()	[isEnrolled] participate()	getGroupClasses()	IN_CLASS	
26	new	pay()	enter()	[isEnrolled] participate()	enroll()	IN_CLASS	
27	new	pay()	enter()	[isEnrolled] participate()	leave()	IN_CLUB	
28	new	setActive(True)				SUSPENDED	InvalidOperationException
29	new	setActive(False)				SUSPENDED	InvalidOperationException
30	new	suspend()				SUSPENDED	InvalidOperationException
31	new	exit()				SUSPENDED	InvalidOperationException
32	new	participate()				SUSPENDED	InvalidOperationException
33	new	leave()				SUSPENDED	InvalidOperationException
34	new	getGroupClasses()				SUSPENDED	InvalidOperationException
35	new	enroll()				SUSPENDED	InvalidOperationException
36	new	pay()	exit()			ACTIVE	InvalidOperationException
37	new	pay()	participate()			ACTIVE	InvalidOperationException
38	new	pay()	leave()			ACTIVE	InvalidOperationException
39	new	pay()	pay()			ACTIVE	InvalidOperationException
40	new	setActive(False)	suspend()			INACTIVE	InvalidOperationException
41	new	setActive(False)	exit()			INACTIVE	InvalidOperationException
42	new	setActive(False)	participate()			INACTIVE	InvalidOperationException
43	new	setActive(False)	leave()			INACTIVE	InvalidOperationException
44	new	setActive(False)	pay()			INACTIVE	InvalidOperationException
45	new	setActive(False)	enroll()			INACTIVE	InvalidOperationException
46	new	pay()	enter()	setActive(True)		IN_CLUB	InvalidOperationException
47	new	pay()	enter()	setActive(False)		IN_CLUB	InvalidOperationException
48	new	pay()	enter()	suspend()		IN_CLUB	InvalidOperationException
49	new	pay()	enter()	enter()		IN_CLUB	InvalidOperationException
50	new	pay()	enter()	[isNotEnrolled] participate()		IN_CLUB	InvalidOperationException
51	new	pay()	enter()	leave()		IN_CLUB	InvalidOperationException
52	new	pay()	enter()	pay()		IN_CLUB	InvalidOperationException
53	new	pay()	enter()	[isEnrolled] participate()	setActive(True)	IN_CLASS	InvalidOperationException
54	new	pay()	enter()	[isEnrolled] participate()	setActive(False)	IN_CLASS	InvalidOperationException
55	new	pay()	enter()	[isEnrolled] participate()	suspend()	IN_CLASS	InvalidOperationException
56	new	pay()	enter()	[isEnrolled] participate()	enter()	IN_CLASS	InvalidOperationException
57	new	pay()	enter()	[isEnrolled] participate()	exit()	IN_CLASS	InvalidOperationException
58	new	pay()	enter()	[isEnrolled] participate()	pay()	IN_CLASS	InvalidOperationException

Figure 4: Conformance Matrix for Member Test Suite

2   GroupClass Class Class Scope Test

Test pattern used: Class invariant

We chose this pattern because GroupClass is a non-modal class, i.e., there is no constraints on message sequences. The objective is to find out which sequences may originate error.

Constraint			Test Cases														
Variable	Condition	Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
beginHour	>= 8	ON	8															
		OFF		7														
	<= 22	ON			22													
		OFF				23												
	Typical	IN					9	10	11	12	13	14	15	16	17	18	19	20
duration	> 0	ON					0											
		OFF						1										
	typical	IN	2	3	4	5			6	7	8	9	10	11	12	50	77	88
capacity	>= 5	ON							5									
		OFF								4								
	<= 25	ON									25							
		OFF										26						
	Typical	IN	7	8	9	10	11	12					13	14	15	20	21	22
club	eq null	ON											null					
		OFF												CLUB				
		IN	CLUB	CLUB	CLUB	CLUB	CLUB	CLUB	CLUB	CLUB	CLUB	CLUB			CLUB	CLUB	CLUB	CLUB
minAge	>= 0	ON													0			
		OFF														-1		
	< 20	ON															20	
		OFF																19
	Typical	IN	1	2	3	4	7	9	10	11	12	15	17	18				
Expected Result			✓	✗	✓	✗	✗	✓	✓	✗	✓	✗	✗	✓	✓	✗	✗	✓

Figure 5: GroupClass Test Matrix

Modifier sequences for each test:

1. new GroupClass(8, 2, 7, healthClub, 1, true);

2. new GroupClass(7, 3, 8, healthClub, 2, true);

3. new GroupClass(10, 4, 8, healthClub, 3, true); setBeginHour(22); setCapacity(9);

4. new GroupClass(10, 5, 8, healthClub, 3, true); setBeginHour(23); setMinAge(4);

5. new GroupClass(10, 0, 8, healthClub, 3, true);

6. new GroupClass(10, 1, 8, healthClub, 3, true); setCapacity(12); setMinAge(9);

7. new GroupClass(12, 7, 10, healthClub, 11, true); setCapacity(5);

8. new GroupClass(16, 6, 20, healthClub, 10, true); setBeginHour(11); setCapacity(4);

9. new GroupClass(13, 8, 25, healthClub, 12, true); setBeginHour(13); setCapacity(25); setMinAge(12);

10. new GroupClass(14, 9, 15, healthClub, 15, true); setCapacity(26);

11. new GroupClass(15, 10, 13, healthClub, 17, true);

12. new GroupClass(16, 11, 14, healthClub, 10, true); setCapacity(14); setMinAge(18);

13. new GroupClass(17, 12, 10, healthClub, 10, true); setCapacity(15); setMinAge(0);

14. new GroupClass(13, 12, 10, healthClub, -1, true);

15. new GroupClass(18, 77, 21, healthClub, 19, true); setBeginHour(19); setMinAge(20);

16. new GroupClass(20, 88, 10, healthClub, 19, true); setCapacity(22); setMinAge(19);

3 Compute Membership Cost Method Scope Test

Test pattern used: Combinational Function test

We used this pattern because the computeMembershipCost method relies on a combination of 4 input parameters (the member's age, the number of group classes the member is enrolled in, the number of years of membership of the member, the member's state) in order to compute a member's monthly fee.
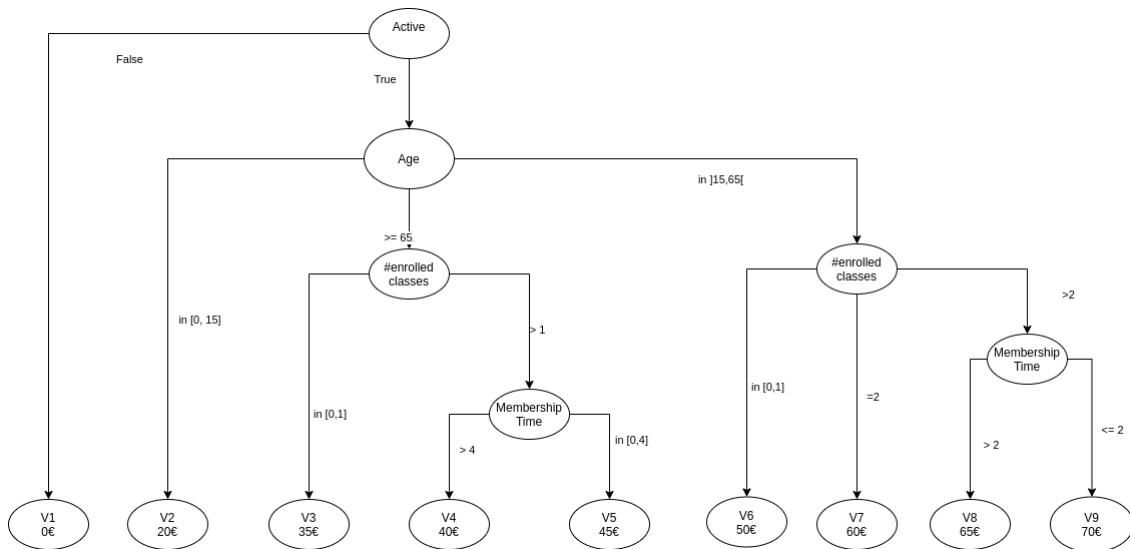
Figure 6: Decision Tree for computeMembershipCost method

We found 9 different variants, and developed an invariant boundary matrix for each one.

Constraint		Type	Test Cases	
Variable	Condition		#1	
active		ON	FALSE	
		OFF		TRUE
age		IN	10	50
#enrolledclasses		IN	0	5
MemberShipTime		IN	0	5
Expected Result			0	V2,V3,V4,V5,V6,V7,V8.V9

Figure 7: Combinational Testing Variant 1

Variable	Condition	Type		#1	#2	#3
Constraint		**Type**		**Test Cases**		
active	eq false	ON	FALSE			
		OFF		TRUE		
		IN			TRUE	TRUE
age	<= 15	ON			15	
		OFF				16
		IN	10	12		
#enrolledclasses		IN	0	2	4	6
MemberShipTime		IN	0	2	4	6
Expected Result			V1	20€	20€	V6,V7,V8,V9

Figure 8: Combinational Testing Variant 2

Variable	Condition	Type		#1	#2		#3	#4
Constraint		**Type**		**Test Cases**				
active	eq false	ON	FALSE					
		OFF		TRUE				
		IN			TRUE	TRUE	TRUE	TRUE
age	>= 65	ON			65			
		OFF				64		
		IN	66	67			70	80
#enrolledclasses	<= 1	ON					1	
		OFF						2
		IN	0	1	0	1		
MemberShipTime		IN	0	2	4	6	8	10
Expected Result			V1	35€	35€	V6,V7,V8,V9	35€	V4,V5

Figure 9: Combinational Testing Variant 3

Constraint		Type	Test Cases							
Variable	Condition			#1	#2			#3		#4
active	eq false	ON	FALSE							
		OFF		TRUE						
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
age	>= 65	ON			65					
		OFF				64				
		IN	66	67			70	80	90	95
#enrolledclasses	> 1	ON					1			
		OFF						2		
		IN	3	4	5	6			7	8
MemberShipTime	> 4	ON							4	
		OFF								5
		IN	6	7	8	9	10	11		
Expected Result			V1	40€	40€	V6,V7,V8,V9	V3	40€	V5	40€

Figure 10: Combinational Testing Variant 4

Constraint		Type	Test Cases							
Variable	Condition			#1	#2			#3	#4	
active	eq false	ON	FALSE							
		OFF		TRUE						
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
age	>= 65	ON			65					
		OFF				64				
		IN	66	67			70	80	90	95
#enrolledclasses	> 1	ON					1			
		OFF						2		
		IN	3	4	5	6			7	8
MemberShipTime	<=4	ON							4	
		OFF								5
		IN	0	0	1	2	3	4		
Expected Result			V1	45€	45€	V6,V7,V8,V9	V3	45€	45€	V4

Figure 11: Combinational Testing Variant 5

Constraint		Type	Test Cases								
Variable	Condition			#1		#2		#3	#4		
active	eq false	ON	FALSE								
		OFF		TRUE							
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	
age	>15	ON			15						
		OFF				16					
	<65	ON					65				
		OFF						64			
		IN	16	17					18	19	
#enrolledclasses	<=1	ON							1		
		OFF								2	
		IN	0	1	0	1	0	1			
MemberShipTime		IN	0	2	4	6	8	10	12	14	
Expected Result			V1	50€	V2	50€	V3,V4,V5	50€	50€	V7	

Figure 12: Combinational Testing Variant 6

Constraint		Type	Test Cases								
Variable	Condition			#1		#2		#3	#4		
active	eq false	ON	FALSE								
		OFF		TRUE							
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	
age	>15	ON			15						
		OFF				16					
	<65	ON					65				
		OFF						64			
		IN	16	17					18	19	
#enrolledclasses	eq 2	ON							2		
		OFF								1	
		IN	2	2	2	2	2	2			
MemberShipTime		IN	0	2	4	6	8	10	12	14	
Expected Result			V1	60€	V2	60€	V3,V4,V5	60€	60€	V6	

Figure 13: Combinational Testing Variant 7

Constraint		Type	Test Cases									
Variable	Condition		#1		#2		#3		#4		#5	
active	eq false	ON	FALSE									
		OFF		TRUE								
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
age	>15	ON			15							
		OFF				16						
	<65	ON					65					
		OFF						64				
		IN	16	17					18	19		
#enrolledclasses	> 2	ON							2			
		OFF								3		
		IN	3	4	5	6	7	8			9	10
MemberShipTime	> 2	ON									2	
		OFF										3
		IN	4	5	6	7	8	9	10	11		
Expected Result			V1	65€	V2	65€	V3,V4,V5	65€	V7	65€	V9	65€

Figure 14: Combinational Testing Variant 8

Constraint		Type	Test Cases									
Variable	Condition		#1		#2		#3		#4		#5	
active	eq false	ON	FALSE									
		OFF		TRUE								
		IN			TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE	TRUE
age	>15	ON			15							
		OFF				16						
	<65	ON					65					
		OFF						64				
		IN	16	17				18	19	20	21	
#enrolledclasses	> 2	ON							2			
		OFF								3		
		IN	3	4	5	6	7	8			9	10
MemberShipTime	<= 2	ON									2	
		OFF										3
		IN	0	1	2	0	1	2	0	1		
Expected Result			V1	70€	V2	70€	V3,V4,V5	70€	V7	70€	70€	V8

Figure 15: Combinational Testing Variant 9

4 Enroll Method Scope Test (Within GroupClass Class)

Test pattern used: Category Partition

We chose this pattern because implements several functions.

Functions of enroll method:

- Primary: properly update the list of member's in the GroupClass (either by adding a new member, adding a new member and removing an existing one or by doing nothing).

- Secondary: Return true if the member is enrolled and false otherwise.

The objective of the test suite is to try to find out which combinations of method parameters and instance variables result in an incorrect input.

Parameter	Category	Choices
Type GroupClass	Standard cases	gold
		silver
Type of Member	Standard cases	gold
		silver
State of the list of already enrolled members	standard size	size $\in$ [0, capacity - 1] members enrolled member regardless of membership type(given member not included)
	Special Cases	full list with only golden members (given member not included)
		full list with some silver members (given member not included)
		member already in list

Figure 16: Enroll Method Categories

Note: we assumed TRUE is returned if the when the call of enroll() returns, the member is in the list of members of the GroupClass, regardless of when we was added (on a previous call or on the current). FALSE is returned, if when enroll() returns, the member is not in the list of members of the GroupClass.

Test	Member age greater or equal than min age	Type of Member	Type Of Class	State of the enrolled member list	Exception	Action (made to list)	Return
1	TRUE	gold	gold	size ∈ [0, capacity - 1]		add member	TRUE
2	TRUE	gold	gold	full w/ golden		unchanged	FALSE
3	TRUE	gold	gold	full w/ some silver		remove one silver and add new member	TRUE
4	TRUE	gold	gold	member in list		unchanged	TRUE
5	TRUE	gold	silver	size ∈ [0, capacity - 1]		add member	TRUE
6	TRUE	gold	silver	full w/ golden		unchanged	FALSE
7	TRUE	gold	silver	full w/ some silver		unchanged	FALSE
8	TRUE	gold	silver	member in list		unchanged	TRUE
9	TRUE	silver	gold	size ∈ [0, capacity - 1]		add member	TRUE
10	TRUE	silver	gold	full w/ golden		unchanged	FALSE
11	TRUE	silver	gold	full w/ some silver		unchanged	FALSE
12	TRUE	silver	gold	member in list		unchanged	TRUE
13	TRUE	silver	silver	size ∈ [0, capacity - 1]		add member	TRUE
14	TRUE	silver	silver	full w/ golden		unchanged	FALSE
15	TRUE	silver	silver	full w/ some silver		unchanged	FALSE
16	TRUE	silver	silver	member in list		unchanged	TRUE
17	FALSE	gold	gold	size ∈ [0, capacity - 1]	InvalidOperation	unchanged	null
18	FALSE	gold	gold	full w/ golden	InvalidOperation	unchanged	null
19	FALSE	gold	gold	full w/ some silver	InvalidOperation	unchanged	null
20	FALSE	gold	gold	member in list	InvalidOperation	unchanged	null
21	FALSE	gold	silver	size ∈ [0, capacity - 1]	InvalidOperation	unchanged	null
22	FALSE	gold	silver	full w/ golden	InvalidOperation	unchanged	null
23	FALSE	gold	silver	full w/ some silver	InvalidOperation	unchanged	null
24	FALSE	gold	silver	member in list	InvalidOperation	unchanged	null
25	FALSE	silver	gold	size ∈ [0, capacity - 1]	InvalidOperation	unchanged	null
26	FALSE	silver	gold	full w/ golden	InvalidOperation	unchanged	null
27	FALSE	silver	gold	full w/ some silver	InvalidOperation	unchanged	null
28	FALSE	silver	gold	member in list	InvalidOperation	unchanged	null
29	FALSE	silver	silver	size ∈ [0, capacity - 1]	InvalidOperation	unchanged	null
30	FALSE	silver	silver	full w/ golden	InvalidOperation	unchanged	null
31	FALSE	silver	silver	full w/ some silver	InvalidOperation	unchanged	null
32	FALSE	silver	silver	member in list	InvalidOperation	unchanged	null

Figure 17: Enroll Method Test Cases