Software Testing and Validation

Project Report 2023/2024

1. Introduction

In this report we will present test cases for some methods and classes and we will justify and explain the reasons why we choose a specific testing pattern according to what we learnt in the theoretical classes. Afterwards we will demonstrate possible tests and the battery of them will be presented by using the TestNG framework written in a java file.

2. Method Scope test cases

2.1 validate() of Class ExamModel

The test pattern we choose for the test cases of the method validade() is the Combinational Function Test, since the validate method has a huge amount of restriction that involves different variables and needs a specific state of those variables to achieve an expected result.

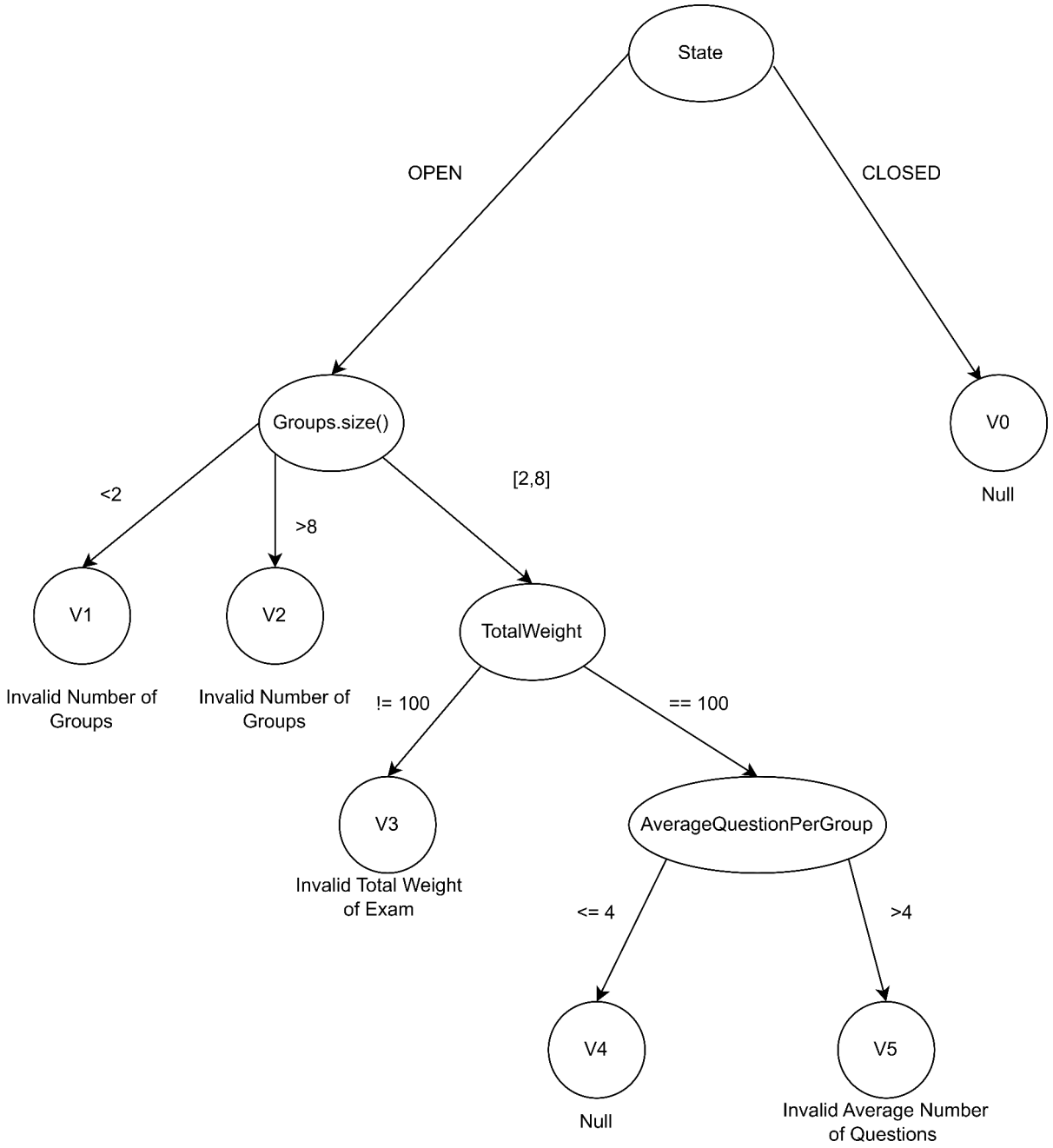
Variables:

- state : State of the exam model
- groups.size(): Number of groups of the exam model
- totalWeight: The cumulative weight that all question of the exam model amount to

- Luís Marques 99265
- José Cardoso 99096
- Yassir Yassin 100611

- averageQuestionPerGroup: Average amount of questions per group

Diagram used to represent the boundary conditions and to determine the variants:



Boundary Conditions for each variant

V0 -> state == CLOSED -> null

V1 -> state == OPEN && groups.size()<2 -> Invalid

V2 > state == OPEN && groups.size()>8 -> Invalid

V3 -> state == OPEN && groups.size() >= 2 && groups.size() <= 8 && TotalWeight != 100 -> Invalid Total Weight of Exam

V4 -> state == OPEN && groups.size() >= 2 && groups.size() <= 8 && TotalWeight == 100 && averageQuestionPerGroup <= 4 -> null

V5 -> state == OPEN && groups.size() >= 2 && groups.size() <= 8 && TotalWeight == 100 && averageQuestionPerGroup > 4 -> Invalid Average Number of Questions

V0	Condition	Type	Test Cases	
			1	-
		ON	CLOSED	
state	"==CLOSED"			
		OFF		OPEN
	TYP	IN		
groups.size()	TYP	IN	3	4
totalWeight	TYP	IN	50	100
averageQuestionsPerGroup	TYP	IN	2	3
Expected Result			null	V3

TVS - Project Report 2023-2024
Group 11

V1	Condition	Type		Test Cases		
			-	2	-	3
		ON	OPEN			
state	"==OPEN"					
		OFF		CLOSE		
	TYP	IN			OPEN	OPEN
		ON			2	
groups.size()	<2					
		OFF				1
	TYP	IN	0	1		
totalWeight	TYP	IN	50	100	100	100
averageQuestionsPerGroup	TYP	IN	2	3	4	5
Expected Result			Invalid Number of Groups	V0	V3	Invalid Number of Groups

TVS - Project Report 2023-2024
Group 11

V2	Condition	Type		Test Cases		
			-	4	-	5
		ON	OPEN			
state	"==OPEN"					
		OFF		CLOSE		
	TYP	IN			OPEN	OPEN
		ON			8	
groups.size()	>8					
		OFF				9
	TYP	IN	9	10		
totalWeight	TYP	IN	50	100	100	100
averageQuestionsPerGroup	TYP	IN	2	3	4	5
Expected Result			Invalid Number of Groups	V0	V3	Invalid Number of Groups

TVS - Project Report 2023-2024
Group 11

V3	Condition	Type				Test Cases					
			-	-	-	-	-	-	-	6	7
		ON	OPEN								
state	"==OPEN"										
		OFF		CLOSE							
	TYP	IN			OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN
		ON			2						
	>= 2										
		OFF				1					
groups.size()											
		ON					8				
	<= 8										
		OFF						9			
	TYP	IN	3	4					4	3	2
		ON							100		
	"==100"										
totalWeight		OFF								99	
		OFF									101
	TYP		100	100	100	100	100	100			
averageQuestionsPerGroup	TYP	IN	2	3	4	5	6	8	6		7
Expected Result			V4	V0	V4	V1	V5	V2	V5	Invalid Total Weight of Exam	Invalid Total Weight of Exam

TVS - Project Report 2023-2024
Group 11

V4	Condition	Type	Test Cases										
			-	8	9	-	10	-	11	-	-	12	-
		ON	OPEN										
State	"==OPEN"												
		OFF		CLOSE									
	TYP	IN			OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN
		ON			2								
	>= 2												
		OFF				1							
groups.size()													
		ON					8						
	<= 8												
		OFF						9					
	TYP	IN	3	4					2	3	4	5	6
		ON							100				
	"==100"												
totalWeight		OFF								99			
		OFF									101		
	TYP	IN	100	100	100	100	100	100				100	100
		ON										4	
averageQuestionsPerGroup	<= 4												
		OFF											5
	TYP	IN	1	0	2	3	0	1	2	3	1		
Expected Result			null	V0	null	V1	null	V2	null	V3	V3	null	V5

TVS - Project Report 2023-2024
Group 11

V5	Condition	Type	Test Cases										
			-	13	14	-	15	-	-	-	-	-	16
		ON	OPEN										
State	"==OPEN"												
		OFF		CLOSE									
	TYP	IN			OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN	OPEN
		ON			2								
	>= 2												
		OFF				1							
groups.size( )													
		ON					8						
	<= 8												
		OFF						9					
	TYP	IN	2	3					2	3	4	5	6
		ON							100				
	"==100"												
totalWeight		OFF								99			
		OFF									101		
	TYP	IN	100	100	100	100	100	100				100	100

		ON										4	
averageQuestionsPerGroup	>4												
		OFF											5
	TYP	IN	5	6	7	8	9	10	11	12	13		
Expected Result			Invalid Average Number of Questions	V0	Invalid Average Number of Questions	V1	Invalid Average Number of Questions	V2	V3	V3	V3	V4	Invalid Average Number of Questions

Description of the test cases

- The total number of tests are 16.

- To obtain these test cases we made a domain matrix for each variant so we could test all the branches of the diagram. In the matrix, each row represents a set of input values and the column represents a test case where the expected result will depend on the combination of the inputs of different variables from the column itself.

- For all conditions we have one ON point and one OFF point, being the exception the equality that has in general on ON point and two OFF points, this was the case for the totalWeight since if its not 100 it can be lower or higher than 100, but the state is different since we only have two states (OPEN, CLOSE).

- The expected results that are variants are tests that belong to the variant referred in the result, this way we do not repeat tests unnecessarily.

2.2 addQuestion() of class ExamModel

The test pattern we choose for the test cases of the method addQuestion is the Category-Partion pattern, since this method only results in a specific state of the variables and other states do not alter the model. Also it is a straightforward function that encompasses a small number of functions.

1. Identify all functions of the MUT

    a. **Primary Functions**:
        i. Add a question to a specified group in the exam model, if invariant is not violated.

    b. Secondary Functions:

        i. Return a boolean indicating if the question was successfully added or not.
        ii. If the model is closed, do not add the question and return false.
        iii. If the specified group is invalid, do not add the question and return false.
        iv. If the group's topic is not one of the question's topics, do not add the question and return false.
        v. If adding the question would exceed the group's maximum number of questions, do not add it and return false.

vi. if the question is added then the size of the set of the question of the group where the question was added is increased.

2. Identify all input and output parameters of each function

    a. **Input**:

        i. Question q
        ii. String q.body
        iii. List<String> q.getChoices
        iv. int q.getChoices.size
        v. int q.correctChoice
        vi. String q.topic
        vii. int q.weight
        viii. int q.getTopics().size
        ix. Int groupId
        x. List<Question> group.questions
        xi. ExamModel modelState

    b. **Output**:
        i. ExamModel - model
        ii. Boolean - return value
        iii. size of the list of question group.questions

3. Identify categories for each input parameter

Parameter	Category
Question q	Valid question Null
String q.body	Not null null
List<String> q.choices	not full full
int q.getChoices.size	Valid number of choices(between 2 and 8 choices) Invalid number of choices(less than 2 choices or more than 8 choices)
int q.correctChoice	Valid index of the list of choices Invalid index of the list of choices
String q.topic	Valid number of characters(more or equal to 6 characters) Invalid number (less than 6 characters)
int q.weight	Valid question Weight (between 0 and 15) Invalid question Weight (less than 0 or more than 15)
int q.getTopics().size	Valid number of topics (between 1 and 5) Invalid number of topics (less than 1 or more than 5)
Int groupId	Valid group ID (between 1 and number

	of groups in model Invalid group ID (less than 1 or greater than number of groups) [error]
List<Question> group.question	Not full full
int group.questions.size	Valid number of questions in a group(between 1 and the max number given when the group is created)
ExamModel model	model.state = Open model.state = Closed

4. Partition each category into choices

Parameter	Category	Choices
Question q	Valid question Null	new Question() null
String q.body	Not null null	“abcdefghi” null
List<String> q.choices	not full full	[C1], [C1,....Cm], 0< m < q.choices.length, [C1,....,C(q.choice.length - 1)] [C1,.....C(q.choice.length)]
int q.getChoices.size	Valid number of choices(between 2 and 8 choices) Invalid number of choices(less than 2 choices or more than 8 choices)	2, 8 1,9
int q.correctChoice	Valid index of the list of choices Invalid index of the list of choices	0, choices.size() - 1 -1, choices.size()
String q.topic	Valid number of characters(more or equal to 6 characters) Invalid number (less than 6 characters)	“abcdef” “abcde”
int q.weight	Valid question Weight (between 0 and 15) Invalid question Weight (less than 0 or more than 15)	1, 15 0,16

int q.getTopics().size	Valid number of topics (between 1 and 5) Invalid number of topics (less than 1 or more than 5)	1, 5 0, 6
Int groupId	Valid group ID (between 1 and number of groups in model Invalid group ID (less than 1 or greater than number of groups) [error]	1, model.groups.size() 0, model.groups.size() + 1 null
List<Question> group.question	Not full full	[], [Q1,....,Qm] -> 0 < m < group.maxNumberOfQuestions - 1 [Q1,.....,Qgroup.maxNumberOfQuestions-1] [Q1,.....,Qgroup.maxNumberOfQuestions]
int group.questions.size	Valid number of questions in a group(between 1 and the max number given when the group is created)	0, group.maxNumberOfQuestions - 1
ExamModel model	model.state = Open model.state = Closed	

Q -> Question

Qgroup -> group of the question

5. Identify constraints on choices

group id null

6. Generate test cases

	Inputs												Ouputs		
Test Cases	Question	q.body	q.choices	q.choices.size	q.correctChoice	q.topic	q.weight	q.getTopics().size	groupId	group.question	group.questions.size	model	ExamModel	size	return value
1	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
2	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[]	0	CLOSED	new model = old model	old size	FALSE
3	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
4	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
5	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,....,Qm]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
6	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,....,Qm]	0	CLOSED	new model = old model	old size	FALSE
7	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,....,Qm]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
8	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,....,Qm]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
9	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
10	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	0	CLOSED	new model = old model	old size	FALSE
11	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
12	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
13	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
14	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions]	0	CLOSED	new model = old model	old size	FALSE
15	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
16	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	1	[Q1,.....,Qgroup.maxNumberOfQuestions]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
17	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
18	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[]	0	CLOSED	new model = old model	old size	FALSE
19	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
20	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
21	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,....,Qm]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
22	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,....,Qm]	0	CLOSED	new model = old model	old size	FALSE
23	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,....,Qm]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
24	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,....,Qm]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
25	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	0	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
26	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	0	CLOSED	new model = old model	old size	FALSE
27	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	group.maxNumberOfQuestions - 1	OPEN	new model.groups[groupID] = old model.groups[groupID] + [q]	old size + 1	TRUE
28	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions-1]	group.maxNumberOfQuestions - 1	CLOSED	new model = old model	old size	FALSE
29	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions]	0	OPEN	new model = old model	old size	FALSE
30	new Question()	“abcdefghi”	[C1]	2	0	"abcdef"	1	1	model.groups.size()	[Q1,.....,Qgroup.maxNumberOfQuestions]	0	CLOSED	new model = old model	old size	FALSE

- Description of the test cases:

    - We have in total 655 360 (2*2*4*4*4*2*4*4*5*4*2*2 = 655 360) combinations, since we verify all the arguments of the question one by one and also we assume that the function also gives as output the new list of questions of the group and the consequences of that behaviour that leads to more tests. We only displayed the first 30 combinations shown above.

    - Each test case has an expected result that will be a consequence of the respective combination.

    - We assume that the choices begin in the index zero so the correct choice can be between 0 and the q.getChoices.lenght - 1

3. Class Scope test cases

3.1 Question class

Even with the fact that the question class has a great number of conditions in terms of its arguments these conditions do not lead to other states. The class question implements basic data types, strings and list of strings and integers. For these reasons we considered that to test the class Question the test pattern Non-modal Class test is the better option.

- *Class Invariant*

  ∀b ∈ body: b != null ∧
  ∀c ∈ choices: c.size() >= 2 ∧ c.size() <= 8 ∧
  ∀cC ∈ correctChoice: cC >= 0 ∧ cC < c.size() ∧
  ∀t ∈ topics: t != null ∧ t.size() > 0 ∧ t.size() <= 5 ∧
  ∀t1, t2 ∈ topic: t1 != t2 ∧ t.length() >= 6 ∧
  ∀w ∈ weight: w >= 1 ∧ w <= 15

cC -> correctChoice

TVS - Project Report 2023-2024
Group 11

Boundary			Test Cases																							
Variable	Condition	Type	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24
q.body	!= null	ON	null																							
		OFF		"abcdef"																						
	TYP	IN			"a"	"ab"	"abc"	"abcd"	"abcde"	"b"	"bc"	"bcd"	"bcde"	"c"	"cd"	"cde"	"cdef"	"d"	"de"	"def"	"defg"	"e"	"ef"	"efg"	"apsij"	"asaspk"
q.choices.size()	>= 2	ON			2																					
		OFF				1																				
	<= 8	ON					8																			
		OFF						9																		
	TYP	IN	3	4					4	7	5	5	6	7	8	4	6	7	4	3	4	1	2	6	4	5
q.correctChoices	>=0	ON							0																	
		OFF								-1																
	< choices.size()	ON									5															
		OFF										4														
	TYP	IN	1	2	1	0	4	5					3	4	4	2	3	4	2	2	2	1	1	3	2	3
q.weight	>0	ON											0													
		OFF												1												
	<=15	ON													15											
		OFF														16										
	TYP	IN	1	2	3	4	5	6	7	8	9	10					1	2	3	4	5	6	7	8	9	10
q.topics	!= null	ON															null									
		OFF																4								
	topics.size() > 0	ON																	0							
		OFF																		1						
	topics.size() <= 5	ON																			5					
		OFF																				6				
	topics.stream() .allMatch(t -> t.length() >= 6)	ON																					["abcdef"]			
		OFF																						["abcde"]		
	topics.stream() .allMatch(t is unique)	ON																							2 unique	
		OFF																								2 equal
	TYP	IN	["abcde"]	["abcdwe", "avctbag"]	["abcdef"]	["abcdefg"]	["abcdefgj"]	["abcdef", "abcdefg"]	["cmutbagh", "abcdswe", "abcdefgh"]	["fsdfdfd", "asdsdfdf", "sdfdsf", "svjutimae"]	["fsdfdfd", "asdsdfdf", "sdfdsf", "svjutimae", "dasdfsdfw"]	["alçsd*asf*ld"]	["as«pasdk"]	["a+*asw+d"]	["a0pisojojd"]	["asw0s9opjd"]										
Expected Result			x	✓	✓	x	✓	x	✓	x	x	✓	x	✓	✓	x	x	✓	x	✓	✓	x	✓	x	✓	x

X -> IUT rejects this input ✔ -> IUT accepts this value and produces correct output

Description of the test cases

- In total we have 24 test cases.

- In our interpretation we assume that the correctChoice is an index of the list choice and for that reason the value should be between 0 and the size of the list.

- We have 11 constraints each having an ON and OFF point.

- Each column represents a test case and in them we have the inputs given to the parameters that are referred to in the first column

- To represent the input given to verify the constraint of the length of the string topic we gave a list of strings one with a string with 6 characters for the On point and another list with a string with 5 characters.

- To represent the input given to verify the constraint that refers the uniqueness of the topics in the list of topics that the question belongs to we decided to exemplify the ON point by saying that the list of topics have 2 topics that are unique (2 unique) and for the OFF point we referred has input a list of topics where both topics are the same (2 topics equal).

- To represent the two constraints referred before we represented them with pseudocode to simplify the table itself.

- From these 24 test cases we decided to implement the following:

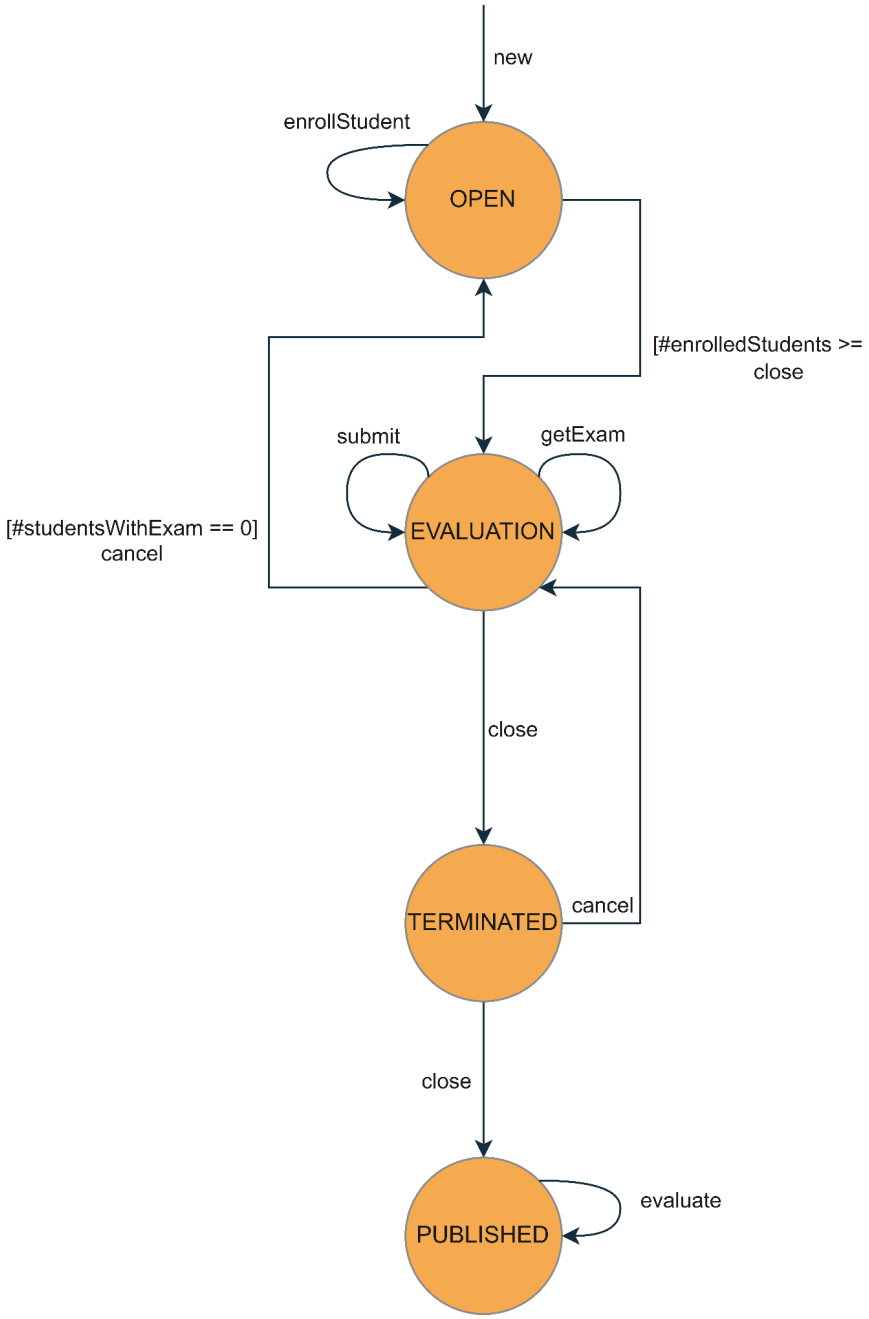
  Pass: TC2, TC7, TC21, TC23
  Not Pass:TC6, TC9, TC22, TC24.

  These test cases are implemented in a separate java file.

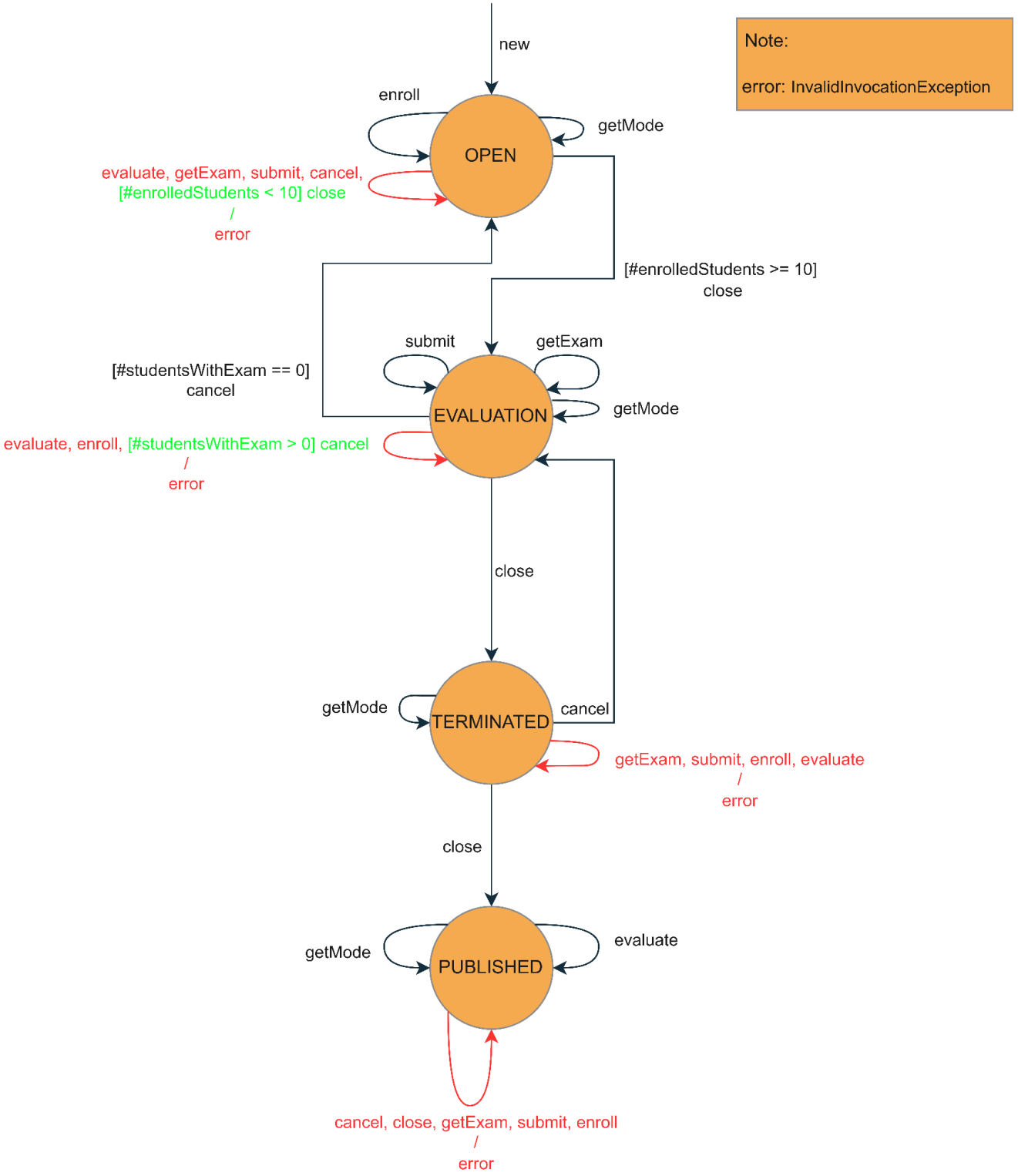
3.2 ExamManager class

For the class ExamManager we decided to take the Model Class Test Pattern with the Finite State Machine Test Pattern since it has a range of states and presents some constraints.

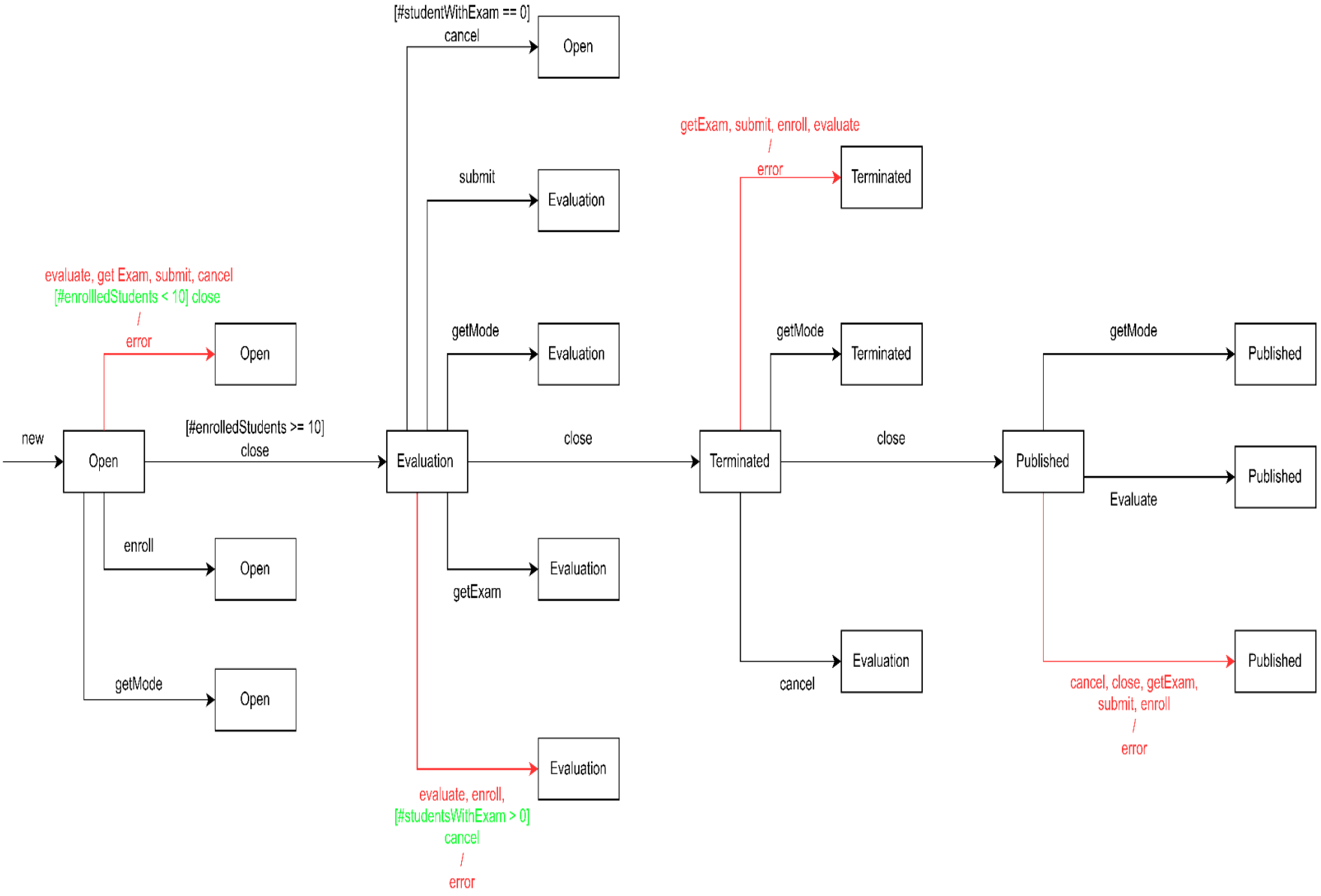
1º Step: Generating the state machine for CUT



2º Step: Full expansion of conditional transition variants applying it to the diagram



3º Step: Generate Transition Tree variants



4º Step: Generate Conformance Test Suite

Test Case	Level 0	Level 1	Level 2	Level 3	Level 4	Expected State	Exception
1	new	-	-	-	-	OPEN	-
2	new	enroll	-	-	-	OPEN	-
3	new	getMode	-	-	-	OPEN	-
4	new	[#enrolledStudent >= 10] close	-	-	-	CLOSE	-
5	new	[#enrolledStudents < 10] close	-	-	-	OPEN	✔
6	new	[#enrolledStudent >= 10] close	[#studentWithExam == 0] cancel	-	-	OPEN	-
7	new	[#enrolledStudent >= 10] close	[#studentWithExam > 0] cancel	-	-	EVALUATION	✔
8	new	[#enrolledStudent >= 10] close	submit	-	-	EVALUATION	-
9	new	[#enrolledStudent >= 10] close	getMode	-	-	EVALUATION	-
10	new	[#enrolledStudent >= 10] close	[#studentWithExam > 0] cancel	-	-	TERMINATED	-
11	new	[#enrolledStudent >= 10] close	close	get Mode	-	TERMINATED	-
12	new	[#enrolledStudent >= 10] close	close	cancel	-	EVALUATION	-
13	new	[#enrolledStudent >= 10] close	close	close	-	PUBLISHED	-
14	new	[#enrolledStudent >= 10] close	close	close	get Mode	PUBLISHED	-
15	new	[#enrolledStudent >= 10] close	close	close	evaluate	PUBLISHED	-

5º Step: Test data for each path using Invariant Boundaries

OPEN	Condition	ON	OFF
close	[#enrolledStudent >= 10]	10	9
	[#enrolledStudent < 10]	10	11

EVALUATION	Condition	ON	OFF
cancel	[#studentWithExam == 0]	0	213
	[#studentWithExam > 0]	0	1

6º Step: Transition Table for Sneak Path

	OPEN	EVALUATION	TERMINATED	PUBLISHED
enroll	✔	PSP	PSP	PSP
close	?	✔	✔	PSP
cancel	PSP	?	✔	PSP
getMode	✔	✔	✔	✔
getExam	PSP	✔	PSP	PSP
evaluate	PSP	PSP	PSP	✔
submit	PSP	✔	PSP	PSP

✓ = Valid Transition; PSP = Possible sneak path; ? = Conditional Transition

Test Case	Level 0	Level 1	Level 2	Level 3	Level 4	Expected State	Exception
16	new	cancel	-	-	-	OPEN	✔
17	new	getExam	-	-	-	OPEN	✔
18	new	evaluate	-	-	-	OPEN	✔
19	new	submit	-	-	-	OPEN	✔
20	new	[#enrolledStudent >= 10] close	enroll	-	-	EVALUATION	✔
21	new	[#enrolledStudent >= 10] close	evaluate	-	-	EVALUATION	✔
22	new	[#enrolledStudent >= 10] close	close	enroll	-	TERMINATED	✔
23	new	[#enrolledStudent >= 10] close	close	getExam	-	TERMINATED	✔
24	new	[#enrolledStudent >= 10] close	close	evaluate	-	TERMINATED	✔
25	new	[#enrolledStudent >= 10] close	close	submit	-	TERMINATED	✔
26	new	[#enrolledStudent >= 10] close	close	close	enroll	PUBLISHED	✔
27	new	[#enrolledStudent >= 10] close	close	close	close	PUBLISHED	✔
28	new	[#enrolledStudent >= 10] close	close	close	cancel	PUBLISHED	✔
29	new	[#enrolledStudent >= 10] close	close	close	getExam	PUBLISHED	✔
30	new	[#enrolledStudent >= 10] close	close	close	submit	PUBLISHED	✔

7º Step: Develop Sneak Path Test Suite

Description of the test:

- The conformance test suite has in total 30 tests being the last 15 of them Sneak Path tests. Testing all the possible transitions, and their respective expected behaviors of the class.

- The Sneak Path Test Suite tests all the paths that are not supposed to happen, being in this case impossible, these are represented in the state machines and in the transition tree in red.

- In the Sneak Path Test suite the exception is specified but it is InvalidInvocationException for all test cases.

- Each row in the Conformance Test Suite are test cases, and each column is the level or more precisely the number of transitions made and below the column we can see the [condition] method that leads to that level in the test case.

- Before the Sneak Path Tests , the first 15 tests were made in the following way, in each level we first put the tests that pass and afterwards we put the tests that will throw an exception being the ones that are expansion of the guards.

- The expansion of conditional transition variants is represented in the 2º diagram and in the 3º step in the transition tree by the color green