TESTES DE CAIXA PRETA

TEST DESIGN SPECIFICATION

Version 2.0

24/11/2019

Grupo_TP1_Q:	
Rui Bessa	8150521
Miguel Guimarães	8150520
José Fernando	8150498

Histórico de Versões

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.0	*Grupo TP1 Q*	12/11/2019	*Grupo TP1 Q*	*Grupo TP1 Q*	1º versão
2.0	*Grupo TP1 Q*	24/11/2019	*Grupo TP1 Q*	*Grupo TP1 Q*	Entrega

Tabela de Conteúdos

1. Introdução

1.1. Identificador do documento

Test_Case_Specification_TP1_Q

1.2. Âmbito

Este relatório foi elaborado para a disciplina de Engenharia de Software II, e tem como objetivo apresentar e explicar os testes de software, caixa preta, realizados pelo grupo “TP1 Q” no primeiro trabalho prático. Pretende-se, também, apresentar a documentação detalhada de todas as definições, planos, abordagens efetuadas e os resultados retornados.

O trabalho desenvolvido passou por realizar os testes unitários de software fornecido pelo docente da disciplina. Este software, consiste na realização de um teste, ou seja, adicionar perguntas a uma coleção (Test) para formar um conjunto de questões. Estas podem ser de três tipos, questões numéricas, questões de verdadeiro ou falso e questões de múltipla escolha.

Para a realização dos testes, foi necessário planear e testar com testes de caixa preta, equivalência e limites, para os métodos addQuestion, numberQuestions, removeQuestion, isComplete, correctAnswers, incorrectAnswers, correctAnswerPecentage, incorrectAnswerPecentage, evaluateAnswer e answer.

Serão de seguida, apresentados todos os dados para o desenvolvimento dos testes de software, tais como, inputs para cada método, outputs esperados e os obtidos, e uma avaliação feita pelos membros do grupo e em alguns casos, acrescentadas observações para possíveis resoluções ou melhorias de problemas encontrados.

1.3. Repositório

A hiperligação para o repositório do gitlab, onde foi efetuado o desenvolvimento deste trabalho, é o seguinte:

http://gitlab.estg.ipp.pt/8150521/esii_tp1_q.git

1.4. Referências

- Slides disponibilizados na plataforma moodle da unidade curricular ESII;
- IEEE Standard for Software Unit Testing;
- S.W.E.B.O.K.;

1.5 Glossário

ECP (equivalency class partitioning) – este método procura definir um caso de teste com o intuito de descobrir classes com erros, reduzindo assim o número tota de casos de teste a serem desenvolvidos.

Uma classe de equivalência representa um conjunto de estados válidos para condições de entrada.

BVA (boundary value analysis) – esta análise leva à escolha de casos de teste que põem à prova os valores limites. Serve de complemento aos testes de partição de equivalência. Em vez de selecionar qualquer elemento de uma classe de equivalência, o BVA leva à seleção de casos de testes nos extremos da classe.

2. Features/Itens a testar

Item a testar	Descrição	Requisitos	Responsabilidade
addQuestion	Adicionar uma questão ao teste	Existência de uma coleção de perguntas previamente instanciada	Rui / Fernando
removeQuestion	Remover questão de uma determinada posição no teste	Existência de uma coleção de perguntas previamente instanciada	Rui
numberQuestions	Retorno o número de questões existentes no teste	Existência de uma coleção de perguntas previamente instanciada	Fernando/ Miguel
isComplete	Retorna o valor true se a questão estiver todas respondidas, se não retorna false.	Existência de uma coleção de perguntas previamente instanciada	Fernando / Rui
correctAnswers	Retorna uma coleção de respostas corretas do teste	Existência de uma coleção de perguntas previamente instanciada	Miguel / Rui
incorrectAnswers	Retorna uma coleção de respostas incorretas do teste	Existência de uma coleção de perguntas previamente instanciada	Miguel
correctAnswersPercentage	Retorna uma percentagem de número de questões corretas	Existência de uma coleção de perguntas previamente instanciada	Fernando
incorrectAnswersPercentage	Retorna uma percentagem de número de questões incorretas	Existência de uma coleção de perguntas previamente instanciada	Fernando
evaluteAnswer	Retorna o valor true, se a resposta do Utilizador for igual a respota correta da questão, se não, retorna false	correct_answer != null e número de caracteres de correct_answer > 0	Miguel

answer	Método que atribui a resposta do utilizador à questão	correct_answer != null e número de caracteres de correct_answer > 0	Rui

3. Detalhes da abordagem aos testes

De forma a assegurar a deteção de erros de software, com o objetivo de alcançar a qualidade de software, seguimos a abordagem de deteção de intervalos e de casos especiais, bem como os limites dos intervalos, através de técnicas conhecidas como "Input Domain-Based Techniques".

Mais concretamente, através das técnicas conhecidas como "Equivalence Class Partitioning" (ECP) e "Boundary Value Analysis" (BVA) para posteriormente detetar casos de teste suficientes que cubram todas os intervalos detetados através de testes JUnit.

Em primeiro lugar realizamos para cada método de forma isolada a tabela ECP para identificação dos vários intervalos possíveis e não possíveis, identificados através de classes válidas e classes inválidas.

Após a tabela ECP, realizamos a tabela BVA para identificar os limites dos vários intervalos identificados anteriormente, de realçar, que é sempre testado o limite inferior, limite inferior menos um, limite inferior mais um, limite máximo, limite máximo menos um e limite máximo mais um. Para além dos limites dos intervalos, na tabela BVA incluimos também os chamados casos especiais, como por exemplo string vazia ("") e string nula (null).

Por último, mas não menos importante, será elaborada uma tabela de testes para cada teste realizado, com os inputs introduzidos, outputs esperados, outputs obtidos e indicação se o teste passou ou não. Bem como observações pelos elementos do grupo em alguns casos em que o teste não passou, para por exemplo, indicar qual foi o motivo da falha.

Com esta abordagem, pretendemos detetar o maior número possível de erros com o menor número de casos de teste.

Estes testes serão executados no IDE IntelliJ usando a ferramenta de testes unitários JUnit, assim como todas as dependências utilizando o Gradle para assegurar que todos os elementos do grupo usam as mesmas versões.

Ao longo do desenvolvimento dos testes foi usado um repositório no GitLab para facilitar o trabalho em grupo e ter registado e controlado as várias versões e alterações do projeto.

4. Identificação dos Testes

De seguida são ilustradas as várias tabelas ECP elaboradas pelo grupo, foi uma para cada método, assim como as respetivas tabelas BVA, que foi uma para cada tabela ECP.

Devido a algumas imagens poderem não estar bastante percetiveis, enviamos em anexo as respetivas tabelas em formato excel.

ECP (Equivalence Class Partitioning) Método addQuestion()

	Pré-suposto	O teste só pode ter no máximo 100 perguntas
	Pré-condições Gerais	Existência de uma coleção de perguntas previamente instanciada

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
TestException	Nº argumentos entrada	1	[ECP1]	1	[ECP2]
	Tipo argumentos entrada	IQuestion		IQuestion	
	Restrições de entrada	q = null		q != null	
	Resultado Esperado	Throws: TestException		!= (Throws: TestException)	
	Exemplo	q = null		Q1	
True	Pré-condições Específicas	!= TestException	[ECP3]	!= TestException	[ECP4]
	Nº argumentos entrada	1		1	
	Tipo argumentos entrada	IQuestion		IQuestion	
	Restrições de entrada	q != null		q != null	
	Critérios de validade	-		-	
	Resultado Esperado	true		false	
	Exemplo	Q1		Q1	
False	Pré-condições Específicas	!= TestException	[ECP5]	!= TestException	[ECP6]
	Nº argumentos entrada	1		1	
	Tipo argumentos entrada	IQuestion		IQuestion	
	Restrições de entrada	q != null		q != null	
	Critérios de validade	-		-	
	Resultado Esperado	false		true	
	Exemplo	numberQuestions() = 100		q = null	
q1	Iquestion q1 = new QuestionYesNo("title", "testDescription", "no")				

BVA (Boundary Value Analysis) método addQuestion()

BVA	ECP	Inputs	Resultado Esperado
(BVA1)	(ECP2, ECP3, ECP6)	adicionar a primeira questão	True
(BVA2)		adicionar a segunda questão	
(BVA3)		adicionar a questão 99	
(BVA4)		adicionar a questão 100	
(BVA5)	(ECP2, ECP4, ECP5)	adicionar a questão 101	False

ECP (Equivalence Class Partitioning) Método removeQuestion()

	Pré-suposto	O teste só pode ter no máximo 100 perguntas
	Pré-condições Gerais	Existência de uma coleção de perguntas previamente instanciada

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
True	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	1		!= 1	
	Tipo argumentos entrada	int		!= int	
	Restrições de entrada	pos >= 0 && pos < 100		pos < 0 || pos >= 100	
	Critérios de validade	-		-	
	Resultado Esperado	true		ArrayIndexOutOfBoundsException	
	Exemplo	pos = 50 & Test().getQuestion(pos) != null		pos = 100	
False	Pré-condições Específicas	-	[ECP3]	-	[ECP4]
	Nº argumentos entrada	1		!= 1	
	Tipo argumentos entrada	int		!= int	
	Restrições de entrada	pos >= 0 && pos < 100		pos < 0 || pos >= 100	
	Critérios de validade	-		-	
	Resultado Esperado	false		ArrayIndexOutOfBoundsException	
	Exemplo	pos = 50 & Test().getQuestion(pos)=null		pos = 100	

BVA (Boundary Value Analysis) método removeQuestion()

BVA	ECP	Inputs	Resultado Esperado
(BVA1)	(ECP1)	inserir pos = 0	True
(BVA2)		inserir pos = 1	
(BVA3)		inserir pos = 98	
(BVA4)		inserir pos = 99	
(BVA5)	(ECP2, ECP4)	inserir pos = -1	ArrayIndexOutOfBoundsException
(BVA6)	(ECP2, ECP4)	inserir pos = 100	

ECP (Equivalence Class Partitioning) Método correctAnswers()

	Pré-suposto	O teste só pode ter no máximo 100 perguntas
	Pré-condições Gerais	Existência de uma coleção de perguntas previamente instanciada

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
correctAnswer	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	1		1	
	Tipo argumentos entrada	IQuestion[]		IQuestion[]	
	Restrições de entrada	IQuestions[{0<=x<=100}]		IQuestions[{x<0 || x>100}]	
	Critérios de validade	-		-	
	Resultado Esperado	IQuestions[{0<=x<=100}]		Exception	
	Exemplo	-		-	

BVA (Boundary Value Analysis) método correctAnswers()

BVA	ECP	Inputs	Resultado Esperado
(BVA1)	(ECP1)	nenhuma questão certa	Iquestion[0]
(BVA2)		apenas primeira questão certa	Iquestion[1]
(BVA3)		apenas segunda questão certa	Iquestion[1]
(BVA4)		apenas a questão 99 certa	Iquestion[1]
(BVA5)		apenas a questão 100 certa	Iquestion[1]
(BVA6)		primeiras 99 questões certas	Iquestion[99]
(BVA7)		últimas 99 questões certas	Iquestion[99]
(BVA8)		todas as questões certas	Iquestion[100]
(BVA9)	(ECP2)	adicionar a questão 101	Protegida pela linguagem

ECP (Equivalence Class Partitioning) Método incorrectAnswers()

Pré-suposto	O teste só pode ter no máximo 100 perguntas
Pré-condições Gerais	Existência de uma coleção de perguntas previamente instanciada

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
incorrectAnswers	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	1		1	
	Tipo argumentos entrada	IQuestion[]		IQuestion[]	
	Restrições de entrada	IQuestions[{0<=x<=100}]		IQuestions[{x<0 || x>100}]	
	Critérios de validade	-		-	
	Resultado Esperado	IQuestions[{0<=x<=100}]		Exception	
	Exemplo	-		-	

BVA (Boundary Value Analysis) método incorrectAnswers()

BVA	ECP	Inputs	Resultado Esperado
(BVA1)	(ECP1)	nenhuma questão errada	Iquestion[0]
(BVA2)		apenas primeira questão errada	Iquestion[1]
(BVA3)		apenas segunda questão errada	Iquestion[1]
(BVA4)		apenas a questão 99 errada	Iquestion[1]
(BVA5)		apenas a questão 100 errada	Iquestion[1]
(BVA6)		primeiras 99 questões erradas	Iquestion[99]
(BVA7)		últimas 99 questões erradas	Iquestion[99]
(BVA8)		todas as questões erradas	Iquestion[100]
(BVA9)	(ECP2)	adicionar a questão 101 errada	Protegida pela linguagem

Método EvaluateAnswer()

Pré-suposto	limite len(String) = 26
Pré-condições Gerais	correct_answer != null & length(correct_answer) > 0

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
Exceção	Nº argumentos entrada	2 ("correct_answer" e "user_answer")	[ECP1]	2	[ECP2]
	Tipo argumentos entrada	String		String	
	Restrições de entrada	user_answer = null		user_answer != null	
	Resultado Esperado	Throws: Exception		!= excecao	
	Exemplo	user_answer = null & correct_answer = "Polymorphism"		user_answer = "no" & correct_answer = "no"	
True	Pré-condições Específicas	!= excecao	[ECP2]	!= excecao	[ECP3]
	Nº argumentos entrada	2		2	
	Tipo argumentos entrada	String		String	
	Restrições de entrada	length(String) >= 1 & length(String) <= 26		length(String) >= 1 & length(String) <= 26	
	Critérios de validade	user_answer = correct_answer		user_answer != correct_answer	
	Resultado Esperado	true		false	
	Exemplo	user_answer = "no" & correct_answer = "no"		user_answer = "2" & correct_answer = "1"	
False	Pré-condições Específicas	!= excecao	[ECP3]	!= excecao	[ECP2]
	Nº argumentos entrada	2		2	
	Tipo argumentos entrada	String		String	
	Restrições de entrada	length(String) >= 1 & length(String) <= 26		length(String) >= 1 & length(String) <= 26	
	Critérios de validade	user_answer != correct_answer		user_answer = correct_answer	
	Resultado Esperado	false		true	
	Exemplo	user_answer = "2" & correct_answer = "1"		user_answer = "no" & correct_answer = "no"	

BVA (Boundary Value Analysis) Método: EvaluateAnswer

Casos de uso	ECP	BVA	Inputs	Resultado Esperado
Exceção	1	(BVA1)	correct_answer = "Polymorphism" & user_answer = ""	Throws: QuestionException
		(BVA2)	correct_answer = "1" & user_answer = ""	
True	2	(BVA3)	correct_answer = "a" & user_answer = "a"	True
		(BVA4)	correct_answer = "a" & user_answer = "A"	
		(BVA5)	correct_answer = "a...z" & user_answer = "a...z"	
		(BVA6)	correct_answer = "0" & user_answer = "0"	
		(BVA7)	correct_answer = "1" & user_answer = "1"	
		(BVA8)	correct_answer = "2,147,483,647" & user_answer = "2,147,483,647"	
		(BVA9)	correct_answer = "no" & user_answer = "NO"	
False	3	(BVA10)	correct_answer = "a" & user_answer = "b"	False
		(BVA11)	correct_answer = "a" & user_answer = "B"	
		(BVA12)	correct_answer = "a...z" & user_answer = "a...y"	
		(BVA13)	correct_answer = "1" & user_answer = "0"	
		(BVA14)	correct_answer = "2,147,483,646" & user_answer = "2,147,483,647"	
		(BVA15)	correct_answer = "no" & user_answer = "YES"	

Método answer()

Pré-suposto	limite length(String) = 26
Pré-condições Gerais	user_answer != null

Casos de uso	Critério	Classe Válida	[ECP]	Classe Inválida	[ECP]
Exceção	Nº argumentos entrada	1	[ECP1]	!1	[ECP2]
	Tipo argumentos entrada	String		!String	
	Restrições de entrada	user_answer != [0-9] && QuestionNumeric		!QuestionNumeric	
	Resultado Esperado	Throws: QuestionException		!= excecao	
	Exemplo	user_answer = "no"		user_answer = 8	
QuestionYesNo	Pré-condições Específicas	-	[ECP3]	-	[ECP4]
	Nº argumentos entrada	1		!1	
	Tipo argumentos entrada	String		!String	
	Restrições de entrada	length(String) >= 1 & length(String) <= 26		length(String) < 1 & length(String) > 26	
	Critérios de validade	-		-	
	Resultado Esperado	setUserAnswer(userAnswer) && setDone(true)		!setUserAnswer(userAnswer) && !setDone(true)	
	Exemplo	user_answer = "no"		user_answer = 2	
QuestionNumeric	Pré-condições Específicas	!= excecao	[ECP5]	-	[ECP6]
	Nº argumentos entrada	1		!1	
	Tipo argumentos entrada	String		!String	
	Restrições de entrada	user_answer = [0-9]		user_answer != [0-9]	
	Critérios de validade	-		-	
	Resultado Esperado	setUserAnswer(userAnswer) && setDone(true)		QuestionException	
	Exemplo	user_answer = "2"		-	
QuestionMultipleChoice	Pré-condições Específicas	-	[ECP7]	-	[ECP8]
	Nº argumentos entrada	1		!1	
	Tipo argumentos entrada	String		!String	
	Restrições de entrada	length(String) >= 1 & length(String) <= 26		length(String) < 1 & length(String) > 26	
	Critérios de validade	-		-	
	Resultado Esperado	setUserAnswer(userAnswer) && setDone(true)		!setUserAnswer(userAnswer) && !setDone(true)	
	Exemplo	user_answer = "no"		user_answer = 2	

BVA (Boundary Value Analysis) Método: answer

Casos de uso	BVA	ECP	Inputs	Resultado Esperado	Resultado obtido
Exceção	BVA1	1	user_answer = ""	Throws: QuestionException	java.lang.NumberFormatException
	BVA2	1	user_answer = null	Throws: QuestionException	java.lang.NullPointerException
QuestionYesNo	BVA3	3	user_answer = "YES"	True	Fail
	BVA4	3	user_answer = "NO"	True	Fail
	BVA5	3	user_answer = "yes minusculo"	True	Pass
	BVA6	3	user_answer = "no minusculo"	True	Pass
	BVA7	4, 8	user_answer = 2	False	Protegido pela linguagem
QuestionNumeric	BVA8	2, 5	user_answer = String.ValueOf(Double.MIN_VALUE – 1)	True	Pass
	BVA9	5	user_answer = String.ValueOf(Double.MIN_VALUE)	True	Pass
	BVA10	5	user_answer = String.ValueOf(Double.MIN_VALUE + 1)	True	Pass
	BVA11	5	user_answer = String.ValueOf(Double.MAX_VALUE – 1)	True	Pass
	BVA12	5	user_answer = String.ValueOf(Double.MAX_VALUE)	True	Pass
	BVA13	2, 5	user_answer = String.ValueOf(Double.MAX_VALUE + 1)	True	Pass
	BVA14	6	user_answer = "palavra"	False	Protegido pela linguagem
QuestionMultipleChoice	BVA15	7	user_answer = "polymorphism"	True	Pass
	BVA16	7	user_answer = "POLYMORPHISM"	True	Fail

Método: correctAnswersPercentage();

Pré-condições gerais: Coleção de perguntas previamente instanciada

		Classe Válida		Classe Inválida	
Casos de uso	**Critério**		**[ECP]**		**[ECP]**
correctAnswersPercentage()	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	ITest		ITest	
	Tipo argumentos entrada	0<=questions<=100		Questions<0 || questions>100	
	Restrições de entrada	isComplete=true		IsComplete!=true	
	Critérios de validade	0.0%<=double<=100.0%		TestException	
	Resultado Esperado	90.0%		isComplete=false	
	Exemplo	-		addQuestion(2)	
TestException	Pré-condições Específicas	-	[ECP3]	-	[ECP4]
	Nº argumentos entrada	ITest		ITest	
	Tipo argumentos entrada	Questions<0 || questions>100		0<=questions<=100	
	Critérios de validade	TestException		0.0%<=double<=100.0%	
	Resultado Esperado	-		90.0%	
	Exemplo	AddQuestion(2)		addQuestion(questionyesno)	

BVA	ECP	Inputs	Resultado Esperado
1	2,3	correctAnswersPercentage(1,2);	*"Build failed with an exception"*
2	2,3	isComplete=false	TestException
3	1,4	CorrectAnswer=0 && numberQuestions=100	0.0
4	1,4	CorrectAnswer=1 && numberQuestions=100	0.01
5	1,4	CorrectAnswer=50 && numberQuestions=100	0.5
6	1,4	CorrectAnswer=99 && numberQuestions=100	0.99
7	1,4	CorrectAnswer=100 && numberQuestions=100	1

Método: incorrectAnswersPercentage();

Pré-condições gerais: Coleção de perguntas previamente instanciad

		Classe Válida		Classe Inválida	
Casos de uso	**Critério**		**[ECP]**		**[ECP]**
ncorrectAnswersPercentage(	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	ITest		0, >1	
	Tipo argumentos entrada	0<=questions<=100		ITest	
	Restrições de entrada	isComplete=true		Questions<0 || questions>100	
	Critérios de validade	0.0%<=double<=100.0%		IsComplete!=true	
	Resultado Esperado	80.0%		TestException	
	Exemplo	-		isComplete=false	
TestException	Pré-condições Específicas	-	[ECP3]	-	[ECP4]
	Nº argumentos entrada	0, >1		1	
	Tipo argumentos entrada	ITest		ITest	
	Restrições de entrada	Questions<0 || questions>100		0<=questions<=100	
	Critérios de validade	IsComplete!=true		isComplete=true	
	Resultado Esperado	TestException		0.0%<=double<=100.0%	
	Exemplo	-		50.0%	

BVA	ECP	Inputs	Resultado Esperado
1	2,3	incorrectAnswersPercentage(1,2);	*"Build failed with an exception"*
2	2,3	isComplete=false	TestException
3	1,4	incorrectAnswer=0 && numberQuestions=100	0.0
4	1,4	incorrectAnswer=1 && numberQuestions=100	0.01
5	1,4	incorrectAnswer=50 && numberQuestions=100	0.5
6	1,4	incorrectAnswer=99 && numberQuestions=100	0.99
7	1,4	incorrectAnswer=100 && numberQuestions=100	1

Método: isComplete();

Pré-condições gerais: Coleção de perguntas previamente instanciada

		Classe Válida		Classe Inválida	
Casos de uso	Critério		[ECP]		[ECP]
True	Pré-condições Específicas	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	1		0, >1	
	Tipo argumentos entrada	ITest		ITest	
	Restrições de entrada	0<=questions<=100		questions<0 || questions>100	
	Critérios de validade	done=true		done=false	
	Resultado Esperado	true		false	
	Exemplo	-		-	
False	Pré-condições Específicas	-	[ECP3]	-	[ECP4]
	Nº argumentos entrada	0, >1		1	
	Tipo argumentos entrada	ITest		ITest	
	Restrições de entrada	0<=questions<=100		0<=questions<=100	
	Critérios de validade	done=false		done=true	
	Resultado Esperado	false		true	
	Exemplo	-		-	

BVA	ECP	Inputs	Resultado Esperado
1	-	isComplete(1,2);	*"Build failed with an exception"*
2	2,3	questions=10&&done==true(0)	false
3	1,4	questions=10&&done==true(1)	false
4	1,4	questions=10&&done==true(5)	false
5	1,4	questions=10&&done==true(9)	false
6	1,4	questions=10&&done==true(10)	true

Método: numberQuestions();					
Pré-condições gerais: Coleção de perguntas previamente instanciada					
		Classe Válida		**Classe Inválida**	
Casos de uso	**Critério**		**[ECP]**		**[ECP]**
True	**Pré-condições Específicas**	-	[ECP1]	-	[ECP2]
	Nº argumentos entrada	1		0, >1	
	Tipo argumentos entrada			ITest	
	Restrições de entrada	ITest		Questions<0 || questions>100	
	Critérios de validade	0<=questions<=100		-	
	Resultado Esperado	-		Excecão	
	Exemplo	0<=int<=100		101	

BVA	ECP	Inputs	Resultado Esperado
1	2	Questions=-1	*"Build failed with an exception"*
2	2	Questions=101	*" ArrayIndexOutOfBoundsException"*
3	2	berQuestions(1	*"Build failed with an exception"*
4	1	uestions=0 (nu	0
5	1	Questions=1	1
6	1	Questions=50	50
7	1	Questions=99	99
8	1	Questions=100	100

5. Critérios de passagem ou falha das features

Como critérios de passagem para os testes definidos anteriormente, considerando se o resultado esperado fosse igual ao output, o teste passava, caso contrario, não passava, assumindo uma falha no método e consequentemente irá ser reportado, para que a correção do método seja realizada. Caso contrário poderá levar a diversas consequências como falha de segurança e performance, assim como erros inesperados.