P.PORTO

ESCOLA
SUPERIOR
DE TECNOLOGIA
E GESTÃO

Engenharia de Software 2

Trabalho pratico 2

Módulo de Transações

<ES2_2021_GRUPO_11>

Version <2.0>

<20/01/2022>

Realizado por:

8180378-Hugo Silva

8200306-Fábio Costa

Histórico de Versões

Version #	Implemented By	Revision Date	Approved By	Approval Date	Reason
1.1	Fabio	3-12-2021	Todos	3-12-2021	Adição de uses cases
1.2	Hugo	4-12-2021	Todos	4-12-2021	Tests BVA BasicEntity
1.3	Hugo	4-12-2021	Todos	4-12-2021	Todos os tests BVA
1.4	Fabio	8-12-2021	Todos	8-12-2021	Alteração dos tests BVA
1.5	Erros Corrigidos	8-1-2022	Todos	8-1-2022	Alteração dos tests BVA

Índice

1.Introdução

Identificador do documento

TestCaseSpecificationMóduloTransações.

Âmbito

Este documento refere-se a um relatório de testes desenvolvido para a disciplina de Engenharia de Software II do curso de Engenharia Informática.

O relatório é realizado no âmbito do trabalho prático 2 e o objetivo é testar a *API* do trabalho 1, mas desta vez com os erros identificados corrigidos.

A estratégia de testes abordada é a estratégia de testes de caixa preta, aplicando técnicas ECP E BVA.

Glossário

BVA – *Boundary Value Analysis*

ECP – *Equivalence Class Partitioning*

TC – *Teste Case*

BVA - Esta Técnica foca-se na análise dos limites do domínio, ou seja, limites superiores e inferiores dos valores fronteira. Além disso, focam-se também em testar valores especiais (por exemplo: *null*).

ECP - Esta Técnica foca-se em dividir os dados de input de uma classe de software em conjuntos de dados equivalentes, esses conjuntos podem ser usados para criar casos de teste.

TC-Grupo de condições utilizadas para testar software.

Referências

IEEE Std 829™-2008

2.Features/Itens a testar

Na seguinte tabela são apresentados os métodos que foram identificados para teste

Item a testar	Descrição	Requisitos	Responsabilidade
Construtor BasicEntity	Constructor que permite criar uma entidade	R8	Hugo
Construtor BasicTransaction	Constructor que permite criar uma transação	R1	Fábio
Método addTransactionLine	Método que adiciona uma linha a uma transação	R2	Fábio
Método removeTransactionLine	Método que remove uma linha de uma transação	R3	Fábio
Método getTransactionLine	Método que devolve uma linha de uma transação	R4	Fábio
Método addTokens	Método que adiciona tokens a uma entidade	R8	Fábio
Método addTransaction	Método que adiciona uma transação a um *ledger*	R11	Fábio
Método removeTransaction	Método que remove uma transação a um *ledger*	R13	Fábio
Método getTransaction	Método que permite obter uma transação de um *ledger*	R12	Fábio
Método getBlock	Método que permite obter um block	R16	Fábio
Construtor BasicTransactionLine	Constructor que permite criar uma linha para uma transação	R18	Hugo
Método registerTransactionsInLedger	Método que regista uma transação no ledger	R6	Hugo
Método registerTransactionsInLedger /getBlockCount	Método que obtém o número de blocos de um *ledger*	R6/R15	Hugo

Método isValidEdger	Método que verifica se os blocos do livro não foram adulterados	17	Hugo
Método addTransactionLine/getTotalValue	Método que retorna o valor total de uma transação	12	Hugo

3. Detalhes da abordagem aos testes

3.1 Construtor BasicEntity

Este construtor tem como responsabilidade garantir que é criada uma entidade válida e está associado ao Use Case "Criar Entidade".

Técnica BVA

A tabela da figura 6 corresponde aos casos de teste com técnica BVA.

Inputs	Estado
name	" "
name	null
district	null

Figura 1 Tabela BVA BasicEntity

São considerados dois valores limites para o name (tipo String), null ou vazio. Para a enumeração é apenas considerado o *input* null.

Técnica ECP

Critérios	Classe Válida	Classe Inválida
Nº inputs	2	!=2
Tipo inputs	String name, District distric	!=String || !=District
Pré-Condições	Sendo um construtor de um objeto não existe pré-condições	
Restrições Inputs (Objeto válido)	name != null && name != " " && district != null	name == null || name == " " || district == null
Pós-Condições	Uma entidade é criada com sucesso	IllegalArgumentException é returnada
Exemplos	("entidade1",district.valueof("Beja"))	("entidade1",District.valueof("Lixa"))

Figura 2 Tabela ECP construtor BasicEntity

Para a técnica ECP, figura 7, foram definidas as classes de dados válidas e as invalidas. Como é um método procede à criação de um objeto não existem pré-condições.

3.2 Construtor BasicTransaction

Este construtor tem como responsabilidade garantir que é criada uma transação válida e está associado ao Use Case "Criar Transação".

Técnica BVA

A tabela (figura 8) corresponde aos casos de teste com técnica BVA.

Inputs	Estado
sender	null
receiver	null

Figura 3 Tabela BVA BasicTransaction

São considerados para teste os dois *inputs* no estado *null*.

Técnica ECP

Critérios	Classe Válida	Classe Inválida
Nº inputs	2	!=2
Tipo inputs	(Entity sender,Entity receiver)	!= (Entity sender,Entity receiver)
Pré-Condições	Dois objetos do tipo Entity corretamente criados	Não tem
Restrições Inputs (Objeto válido)	**sender** != null && **receiver** != null	**sender** == null || **receiver** == null
Pós-Condições	Uma transição é criada com sucesso	IllegalArgumentException é returnada
Exemplos	BasicTransaction (sender, receiver)	BasicTransaction (sender, null)

Figura 4 Tabela ECP construtor BasicTransaction

Para a técnica ECP foram definidas as classes de dados válidas e as inválidas. A criação de uma BasicTransaction tem como pré-condição a instanciação de dois objetos do tipo Entity.

3.3 Método addTransactionLine

Este método tem como responsabilidade garantir que é adicionada uma linha válida à transação e está associado ao Use Case "adicionar linha à transação".

O método addTransactionLine recebe um parâmetro do tipo TransactionLine.

Técnica BVA

A tabela da figura 10 corresponde ao input que foi caso de teste com técnica BVA.

Inputs	Estado
transactionLine	null

Figura 5 Tabela input BVA addTransactionLine

Nos testes BVA para este método foi considerado o cenário em que o input é null.

Inputs	Min	Min blow	MID	MAX	Max above	Special Case
transactionLine	N	N	N	N	N	null

Figura 6 Cenário BVA addTransactionLine

Para a realização dos dois testes BVA deste método é necessário instanciar uma transação.

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#1	É adicionada linha a uma transação	Nº inputs	1	!=1
		Tipo inputs	TransactionLine	!= TransactionLine -> ex: string
		Pré-Condições	Transação Criada (BasicTransaction)	Transação Criada (BasicTransaction)
		Restrições Inputs	O input não pode corresponder a uma linha já adicionada na transição	Não tem
		Pós-Condições	Return true	N
		Exemplos	addTransactionLine (transacao)	N

Figura 7 Tabela ECP Metodo addTransactionLine test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#2	É adicionada linha já existente na transação.	Nº inputs	1	!=1
		Tipo inputs	TransactionLine	! = TransactionLine -> ex: string
		Pré-Condições	Transação Criada (BasicTransaction) com linhas adicionadas	Transação Criada (BasicTransaction)
		Restrições Inputs	O input tem de corresponder a uma linha já adicionada na transição	Não tem
		Pós-Condições	Return False	N
		Exemplos	addTransactionLine (transacao)	N

Figura 8 Tabela ECP Metodo addTransactionLine test case 2

Para o ECP#01 foi escolhido inserir uma linha que não exista no objeto do tipo BasicTransaction. O objetivo é testar o funcionamento do método em condições normais.

Para o ECP#02 foi escolhido inserir uma linha que já existe no objeto do tipo BasicTransaction. O objetivo é testar se o método faz a verificação correta de linhas existentes e retorna o esperado.

3.4 Método removeTransactionLine

Este método permite remover uma linha de uma transação e está associado ao Use Case "remover linha da transação".

O método removeTransactionLine recebe um parâmetro do tipo TransactionLine.

Técnica BVA

A tabela da figura 13 corresponde ao *input* que foi caso de teste com técnica BVA.

Para a técnica BVA foram ponderados dois cenários(figura 14), sendo necessário existir uma transação válida criada:

-Test case a remover passando o parâmetro null;

-Test case remover num array vazio com um parâmetro válido;

Inputs	Min	Min blow	MID	MAX	Max above	Special Case
transactionLine	N	N	N	N	N	null
transactionLine	Remover com array vazio	N	N	N	N	N

Figura 9 TestS Cases BVA removeTransactionLine

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#1	Remover uma linha da transação	Nº inputs	1	!= 1
		Tipo inputs	TransactionLine	! = TransactionLine (Ex: null)
		Pré-Condições	Transação Criada (BasicTransaction) com linhas adicionadas	Transação Criada (BasicTransaction)
		Restrições Inputs	O input tem de corresponder a uma linha adicionada na transição	Não tem
		Pós-Condições	Return true	Return IllegalArgumentException
		Exemplos	removeTransactionLine (transacao)	removeTransactionLine (null)

Figura 10 Tabela ECP removeTransactionLine test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#2	Remover uma linha que não existe na transação	Nº inputs	1	!= 1
		Tipo inputs	TransactionLine	! = TransactionLine (Ex: null)
		Pré-Condições	Transação Criada (BasicTransaction)	Transação Criada (BasicTransaction)
		Restrições Inputs	O input não pode corresponder a uma linha existente na transação	Não tem
		Pós-Condições	Return false	Return IllegalArgumentException
		Exemplos	removeTransactionLine (transacao)	removeTransactionLine (null)

Figura 11 Tabela ECP removeTransactionLine test case 2

Para o ECP#01 foi escolhido remover uma linha que existe no objeto do tipo BasicTransaction. O objetivo é testar o funcionamento do método em condições normais.

Para o ECP#02 foi escolhido remover uma linha que não existe no objeto do tipo BasicTransaction. O objetivo é testar se o método faz a verificação correta das linhas existentes e retorna o esperado.

3.5 Método getTransactionLine

Técnica BVA

Este método permite obter uma linha de uma transação e está associado ao Use Case "obter linha da transação".

O método getTransactionLine recebe um parâmetro do tipo TransactionLine.

A tabela da figura 17 corresponde ao caso de teste.

Inputs	Min	Min blow	MID	MAX	Max above	Special Cases
transactionLine	N	N	N	N	N	Null

Figura 12 Test Cases Bva getTransactionLine

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#1	É obtida de uma transação uma linha com o mesmo hash que a transação passada como parametro	Nº inputs	1	!= 1
		Tipo inputs	TransactionLine	! = TransactionLine -> ex: string
		Pré-Condições	Transação Criada (BasicTransaction) com linhas adicionadas	Transação Criada (BasicTransaction)
		Restrições Inputs	O input tem de corresponder a uma linha adicionada na transição	Não tem
		Output esperado	Objeto do tipo TransactionLine	UnHashableException
		Exemplos	getTransactionLine (transacao)	getTransactionLine (null)

Figura 13 Tabela ECP getTransactionLine case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#2	Tentar obter uma linha com o hash que não existe na transação	Nº inputs	1	!= 1
		Tipo inputs	TransactionLine	! = TransactionLine -> ex: string
		Pré-Condições	Transação Criada (BasicTransaction)	
		Restrições Inputs	O input não deve corresponder a uma linha adicionada na transição	Não tem
		Output esperado	null	UnHashableException
		Exemplos	getTransactionLine (transacao)	getTransactionLine (null)

Figura 14 Tabela ECP getTransactionLine case 2

Para o ECP#01 foi escolhido obter uma linha que existe no objeto do tipo BasicTransaction. O objetivo é testar o funcionamento do método em condições normais.

Para o ECP#02 foi escolhido obter uma linha que não existe no objeto do tipo BasicTransaction. O objetivo é testar se o método faz a verificação correta de linhas existentes e retorna o esperado.

3.6 Método addTokens

Este método permite adicionar tokens a uma entidade e está associado ao Use Case "adicionar tokens".

Técnica BVA

A tabela da figura 22 corresponde aos *inputs* que foram caso de teste com técnica BVA para o método.

Para executar os testes BVA para este método é necessário criar uma entidade.

Os inputs são do tipo *int*.

Inputs	Valor
tokens	-1
tokens	0
tokens	2147483647

Figura 15 Inputs BVA addTokens

Nota: Foi considerado um limite máximo para teste com o valor de 2147483647. Este valor é o maximo que o tipo int aceita como parâmetro(linguagem java)**.**

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
#1	Adiciona Tokens a uma entidade	Nº inputs	1	!=1
		Tipo inputs	int	int
		Pré-Condições	Entidade Criada (BasicEntity)	
		Restrições Inputs	Input >= 0	Input < 0
		Output esperado	Número de tokens adicionados	IllegalStateException
		Exemplos	addTokens (2) addTokens (1900)	addTokens (-3)

Figura 16 Tabela ECP Método addTokens teste case 1

Para este método foi assumido que o elemento neutro da adição pertence à classe de equivalência valida sendo que o valor de tokens deve permancer inalterado quando o 0 é passado como *input*.

3.7 Método addTransaction

Este método adiciona uma transação a um *ledger* e está associado ao Use Case "Adicionar transação".

Técnica BVA

Para executar os testes BVA para este método é necessário ser instanciada uma organização.

A tabela da figura 22 corresponde ao *input* que foi caso de teste com técnica BVA para o método.

O *input* é do tipo Transaction.

Inputs	Valor
transaction	null

Figura 17 Valor do input BVA addTransaction

A tabela da figura 23 mostra o test case realizado. Neste teste é inserido no índice 0 do array uma transação com o estado null.

Case ID	Inputs	Min	Min blow	MID	MAX	Max above	Special Case
1	transactionLine	Posição 0 do array	N	N	N	N	null

Figura 18 Test Case BVA método addTransaction

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Adiciona uma transação a um *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Instanciar objeto tipo BasicOrganization	
		Restrições Inputs	• a transação do input não pode já estar inserida no ledger	• Não tem
		Pós-Condições	Adiciona uma transação ao ledger com sucesso	Throws IllegalArgumentException
		Resultado Esperado	True	IllegalArgumentException
		Exemplos	addTransaction (transation)	addTransaction ("hello")

Figura 19 Tabela ECP addTransaction test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
2	Adiciona uma transação já existente no *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado	
		Restrições Inputs	• a transação do input deve estar inserida no *ledger*	• Não tem
		Pós-Condições	Transação não é adicionada	Throws IllegalArgumentException
		Resultado Esperado	False	IllegalArgumentException
		Exemplos	addTransaction (transation)	(null)

Figura 20 Tabela ECP addTransaction test case 2

Para o ECP#01 foi escolhido inserir uma transação que não exista no objeto do tipo BasicOrganization. O objetivo é testar o funcionamento do método em condições normais.

Para o ECP#02 foi escolhido inserir uma transação que existe no objeto do tipo BasicOrganization. O objetivo é testar se o método faz a verificação correta das transações existentes e retorna o resultado esperado.

Case	Descrição	Critérios	Classe Válida	Classe Inválida
3	Adiciona uma transação já registada num *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Instanciar Objeto do tipo BasicOrganization . Adicionar Transação (válida pra registo) ao *ledger*. Registar transação no *legder*.	Instanciar Objeto do tipo BasicOrganization .
		Restrições Inputs	• a transação do *input* já deve ter sido registada no ledger	• Não tem
		Pós-Condições	Adiciona uma transação ao ledger com sucesso	Throws IllegalArgumentException
		Resultado Esperado	True	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransaction() addTransaction (transation)	addTransaction (null)

Figura 21 Tabela ECP addTransaction test case 3

Para o ECP#03 foi escolhido inserir uma transação já registada no *ledger*. O objetivo é testar se a API quando regista uma transação a remove da lista das transações por registar.

Case	Descrição	Critérios	Classe Válida	Classe Inválida
4	Adiciona uma transação já registada num *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Instanciar Objeto do tipo BasicOrganization . Adicionar Transação (inválida pra registo) ao *ledger*. Tentar registar transação no *legder*.	Instanciar Objeto do tipo BasicOrganization .
		Restrições Inputs	• a transação do *input* não deve ser registada no *ledger*	• Não tem
		Pós-Condições	Adiciona uma transação ao ledger com sucesso	Throws IllegalArgumentException
		Resultado Esperado	false	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransaction() addTransaction (transation)	addTransaction (null)

Figura 22 Tabela ECP test case 4

Para o ECP#04 foi escolhido inserir uma transação já adicionada que não foi registada. O objetivo é testar se a API guarda as transações não registadas para avaliações posteriores.

3.8 Método removeTransaction

Este método remove uma transação a um *ledger* e está associado ao Use Case "Remover transação".

Técnica BVA

Para executar os testes BVA para este método é necessário ser criada uma organização.

A tabela da figura 28 corresponde ao *input* que foi caso de teste com técnica BVA para o método.

O *input* é do tipo Transaction.

Inputs	Valor
transaction	null

Figura 23 Input BVA removeTransaction

A tabela da figura 29 mostra os test cases realizados onde é passado uma transação nula como parâmetro e onde é passado uma transação válida para um *array* vazio.

CaseID	Inputs	Min	Min blow	MID	MAX	Max above
1	transaction	array vazio	N	N	N	N
2	null	N	N	N	N	N

Figura 24 Test Case BVA removeTransaction

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Remove uma transação a um *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado EX: BasicOrganization basicOrganization = new BasicOrganization()	
		Restrições Inputs	• a transação do input tem de estar inserida no *ledger*	• Não tem
		Pós-Condições	Transação é removida do *ledger* com sucesso	Throws IllegalArgumentException
		Resultado Esperado	True	IllegalArgumentException
		Exemplos	removeTransaction (transation)	removeTransaction (null)

Figura 25 Tabela ECP removeTransaction test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
2	Remove transação não existente no *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado	
		Restrições Inputs	• a transação do input não pode estar inserida no *ledger*	• Não tem
		Pós-Condições	Transação não removida	Throws IllegalArgumentException
		Resultado Esperado	False	IllegalArgumentException
		Exemplos	removeTransaction (transation)	removeTransaction (null)

Figura 26 Tabela ECP removeTransaction test case 2

Case	Descrição	Critérios	Classe Válida	Classe Inválida
3	Remove transação registada num bloco.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado com transações registadas	
		Restrições Inputs	• a transação do input deve estar registada num bloco	• Não tem
		Pós-Condições	Transação não removida	Throws IllegalArgumentException
		Resultado Esperado	False	IllegalArgumentException
		Exemplos	removeTransaction (transation)	removeTransaction (null)

Figura 27 Tabela ECP removeTransaction test case 3

Para o ECP#01 foi escolhido remover uma transação que existe no objeto do tipo BasicOrganization. O objetivo é testar o funcionamento do método em condições normais.

Para o ECP#02 foi escolhido remover uma transação que não existe no *ledge*. O objetivo é testar se o método faz a verificação correta das transações existentes e retorna o esperado.

Para o ECP#03 foi escolhido remover uma transação que foi registada num bloco. O objetivo é testar se o método remove transações registadas.

3.9 Método getTransaction

Este método retorna (caso exista) uma transação de um *ledger* e está associado ao Use Case "Obter transação".

Técnica BVA

Para executar os testes BVA para este método é necessário criar uma organização.

A tabela da figura 33 corresponde ao *input* que foi caso de teste com técnica BVA para o método.

O input é do tipo Transaction.

Inputs	Valor
transaction	null

Figura 28 Input BVA método getTransaction

A tabela da figura 34 mostra o test case realizado onde é passado uma transação nula como parâmetro.

CaseID	Inputs	Min	Min blow	MID	MAX	Max above	Special Cases
#1	transaction	N	N	N	N	N	Null

Figura 29 Test Case BVA getTransaction

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Obter uma transação existente no *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado	
		Restrições Inputs	• a transação do input tem de estar inserida no *ledger*	• Não tem
		Pós-Condições	Retorna a transição com o mesmo hash	Throws IllegalArgumentException
		Resultado Esperado	Objeto do tipo Transaction	IllegalArgumentException
		Exemplos	getTransaction (transation)	removeTransaction (null)

Figura 30 Tabela ECP método getTransaction test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
2	Obter uma transação não existente no *ledger*.	Nº inputs	1	!= 1
		Tipo inputs	BasicTransaction	!= BasicTransaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado	
		Restrições Inputs	• a transação do input tem de estar inserida no *ledger*	• Não tem
		Pós-Condições	Return null	Throws IllegalArgumentException
		Resultado Esperado	null	IllegalArgumentException
		Exemplos	getTransaction (transation)	getTransaction (null)

Figura 31 Tabela ECP método getTransaction test case 2

3.10 Método getBlock

Este método retorna um bloco de um *ledger* através de um índice passado como parâmetro *e* está associado ao Use Case "Obter Bloco".

Técnica BVA

Para executar os testes BVA para este método é necessário ser criada uma organização.

A tabela da figura 37 corresponde aos *inputs* que foram caso de teste com técnica BVA.

Os inputs são do tipo *int*.

Inputs	Valor
index	-1
index	0
index	getBlockCount()

Figura 32 Inputs BVA getBlock

A tabela da figura 38 mostra os TCs realizados.

- O #1 é passado o index com o valor 0.

- O #2 é passado um valor limite definido pelo método getBlockCount.

- O #3 é passado o valor -1.

CaseID	Inputs	Min	Min blow	MID	MAX	Max above	Other
#1	index	0	N	N	N	N	N
#2	index	N	N	N	getBlockCount()	N	N
#3	index	-1	N	N	N	N	N

Figura 33 Tabela BVA método getBlock

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Obter bloco de um *ledger* através de um indice	Nº inputs	1	!=1
		Tipo inputs	(Int index)	(Int index)
		Pré-Condições	Objeto do tipo BasicOrganization instanciado	
		Restrições Inputs	index >= 0 && index < getBlockCount () *	index < 0 || index >= getBlockCount ()*
		Pós-Condições	É retornado o bloco no index correspondente	Throws IndexOutOfBoundsException
		Resultado Esperado	Objeto do tipo Block	IndexOutOfBoundsException
		Exemplos	getBlock (0)	getBlock (-3)

Figura 34 Tabela ECP metodo getBlock test case 1

* método getBlockCount() -> método que retorna o número de blocos existentes no ledger.

3.11 Construtor BasicTransactionLine

Este construtor permite criar uma tinha que pode ser adicionada à transação. Está inserido no Use case “Adicionar Linha à transação”.

Técnica BVA

Para testar este construtor são necessários três inputs:

-String itemDescription;

−int quantity;

-double unitPrice;

A tabela figura 40 mostra os valores dos *inputs* que foram usados em Tests Cases.

Inputs	Valor	Valor2
item Description	null	“ ”(empty)
quantity	-1	0
unitPrice	-0.1	0

Figura 35 Valores BVA BasicTransactionLine

Serão realizados 6 testes, sendo que em cada um é passado como parâmetro apenas um valor limite da tabela.

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Criar uma linha para adicionar a uma transação	Nº inputs	3	!=3
		Tipo inputs	(String itemDescription, int quantity, double unitPrice)	(String itemDescription, int quantity, double unitPrice)
		Pré-Condições	Não existem	
		Restrições Inputs	itemDescription != null && itemDescription != " " && quantity > 0 && unitPrice >= 0,0	itemDescription == null || itemDescription == " " || quantity <= 0 || unitPrice < 0,0
		Pós-Condições	Objeto é criado com sucesso	Throws IllegalArgumentException
		Resultado Esperado	Objeto do tipo Block	IllegalArgumentException
		Exemplos	BasicTransactionLine("Huo",2, 2) BasicTransactionLine("Hugo",1,1)	BasicTransactionLine("Hugo",2, -2) BasicTransactionLine("Hugo",-2, 2) BasicTransactionLine(" ",2, 2)

Figura 36 Tabela ECP Construtor BasicTransactionLine test case 1

3.12 Método registerTransactionsInLedger

Este método foi usado para testar se são registadas o número de transações válidas. Está inserido no use case "registar transações no ledger".

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Obter número de transações válidas registadas.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado. (EX: basicOrganization != null)	
		Restrições Inputs	• a transação do input não pode já estar inserida no *ledger* e tem de ser válida pra registo	• Não tem
		Pós-Condições	Adiciona uma transação é registada *ledger* com sucesso	Throws IllegalArgumentException
		Resultado Esperado	1	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransactionsInLedger()	addTransaction ("hello")

Figura 37 Tabela ECP método registerTransactionsInLedger case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
2	Obter número de transações inválidas registadas.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado. (EX: basicOrganization != null)	
		Restrições Inputs	• a transação do input não pode já estar inserida no *ledger* e não pode ser válida pra registo	• Não tem
		Pós-Condições	Transação não é registada no *ledger*	Throws IllegalArgumentException
		Resultado Esperado	0	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransactionsInLedger()	addTransaction ("hello")

Figura 38 Tabela ECP método registerTransactionsInLedger case 2

3.12.1 Método registerTransactionsInLedger/ getBlockCount

Estes métodos foram usados para testar o número de blocos criados depois de registada uma transação. Está inserido no use case "obter número de blocos do *ledger*".

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Obter número de blocos depois de registar uma transação.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado. (EX: basicOrganization != null)	
		Restrições Inputs	• a transação do input não pode já estar inserida no *ledger*	• Não tem
		Pós-Condições	Uma transação é registada *ledger* com sucesso	Throws IllegalArgumentException
		Resultado Esperado	2	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransactionsInLedger() getBlock()	addTransaction ("hello")

Figura 39 Tabela ECP método registerTransactionsInLedger/ getBlockCount test case 1

Case	Descrição	Critérios	Classe Válida	Classe Inválida
2	Obter número de blocos depois tentar registar uma transação inválida.	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado. (EX: basicOrganization != null)	
		Restrições Inputs	• a entidade "sender" da transação adicionada pra registo não deve possuir tokens	• Não tem
		Pós-Condições	Transação não é registada e o número de blocos não é alterado	Throws IllegalArgumentException
		Resultado Esperado	1	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransactionsInLedger() getBlock()	addTransaction (null)

Figura 40 Tabela ECP método registerTransactionsInLedger/ getBlockCount test case 2

Para o ECP01 foi escolhido registar uma transação (que possui as condições para ser registada) no *ledger*. O objetivo é testar o funcionamento do método em condições normais (novo bloco adicionado ao livro).

Para o ECP02 foi escolhido registar uma transação que não tem as condições necessárias. O objetivo é testar se o método não adiciona blocos vazios*.

*Apenas o primeiro bloco deve ser vazio.

Resultados esperados:

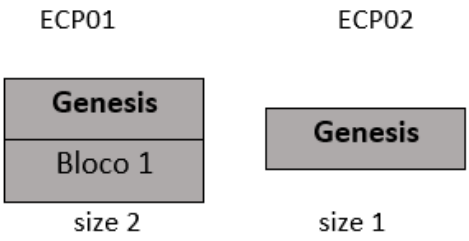

Figura 41 Resultados esperados registerTransaction/getBlockCount

Nota: Não foram realizados teste com a tecnica BVA pois já foram realizados no método addTransaction (método que recebe o *input*).

3.13 Método registerTransactionsInLedger/ isValidEdger

Técnica ECP

Neste método o objetivo é testar se a API faz a verificação correta dos blocos(válido/inválido) do *ledger* e está inserido no use case "verificar se os blocos são válidos".

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Verificar se os blocos são válidos	Nº inputs	1	!= 1
		Tipo inputs	Transaction	!= Transaction
		Pré-Condições	Objeto do tipo BasicOrganization instanciado. (EX: basicOrganization != null)	
		Restrições Inputs	• a transação do input não pode já estar inserida no *ledger* e tem de ser válida pra registar	• Não tem
		Pós-Condições	É retornado que o ledger é invalido (return false)	Throws IllegalArgumentException
		Resultado Esperado	false	IllegalArgumentException
		Exemplos	addTransaction (transation) registerTransactionsInLedger() transaction.add(transationline) organization.isValidEdger()	addTransaction (null)

Figura 42 Tabela ECP Método registerTransactionInLedger

Para testar este método é necessário passar uma transação válida para registo. A transação deve ser registada no *ledger*.

Com a transação registada deve ser alterado o seu valor (por exemplo removendo uma linha).

Ao alterar a transação o bloco irá ficar adulterado pois o *hash* atual será diferente do *hash* de criação.

Ao invocar o método isValidEdger deve ser retornado false, dado que o livro possui um bloco alterado.

Bloco Válido	**Bloco adulterado**	Bloco Válido

Para o test ECP#01 foi adulterado um bloco no "meio" do *array,* depois é verificado se o método faz a correta verificação e retorna os resultados esperados.

Técnica BVA

Para testar este método (figura 48) será usado o valor limite minimo do *array*, ou seja, a posição 0 a posição getBlockCount()-1 .

Test case 1, bloco adulterado na posição 0 do array(sendo que é a única posição do *array*).

Bloco adulterado

Test case 2, o bloco adulterado estará na posição getBlockCount – 1(última posição do *array*).

Bloco Válido	**Bloco adulterado**

Teste case 3, o *ledger* não tem blocos com transações logo não existe possibilidade de existir nenhum adulterado e o livro deve ser válido, ou seja, ou seja, retornado *true*.

Case ID	Inputs	Min	Min blow	MID	MAX	Max above	Special Case
1	transaction	0	N	N	N	N	N
2	transaction	N	N	N	getBlock()-1	N	N
3	transaction	arrayVazio	N	N	N	N	N

Figura 43 Tabela test cases BVA registerTransactionsInLedger/ isValidEdger

3.14 Método addTransactionLine/ getTotalValue

Para estes métodos o objetivo é testar se os valores das linhas correspondem ao valor total da transação.

Técnica ECP

Case	Descrição	Critérios	Classe Válida	Classe Inválida
1	Obter valor de uma transação	Nº inputs	1	!= 1
		Tipo inputs	TransactionLine	!=TransactionLine
		Pré-Condições	Instanciar objeto do tipo BasicOrganization Instanciar 3 objetos BasicTransactionLines	
		Restrições Inputs	• a linha não deve já estar inserida na transação	• Não tem
		Pós-Condições	É retornado o valor total da transação	Throws IllegalArgumentException
		Resultado Esperado	6.0	IllegalArgumentException
		Exemplos	addTransactionLine (transationLine)	(null)

Figura 44 Tabela ECP método addTransactionLine / BasicTransactionLine teste case 1

Para executar o teste serão inseridas 3 linhas na transação e obtido o valor total. Os *inputs* definidos serão utilizados no método addTransactionLine.

Técnica BVA

Não foram executados testes BVA uma vez que estes já foram realizados nos métodos addTransactionLine e BasicTransactionLine.

4. Identificação dos Testes

Construtor BasicEntity

Nome	Tecnica
testBasicEntityBVA01	BVA
testBasicEntityBVA02	BVA
testBasicEntityECP01	ECP

Construtor BasicTransaction

Nome	Tecnica
testBasicTransactionBVA01	BVA
testBasicTransactionECP01	ECP

Método addTransactionLine

Nome	Tecnica
testAddTransactionLineBVA01	BVA
testAddTransactionLineECP01	ECP
testAddTransactionLineECP02	ECP

Método removeTransactionLine

Nome	Tecnica
testRemoveTransactionLineBVA01	BVA
testRemoveTransactionLineBVA02	BVA
testRemoveTransactionLineECP01	ECP
testRemoveTransactionLineECP02	ECP

Método getTransactionLine

Nome	Tecnica
testGetTransactionLineBVA01	BVA
testGetTransactionLineECP01	ECP
testGetTransactionLineECP02	ECP

Método addTransaction

Nome	Tecnica
testAddTransactionBVA01	BVA
testAddTransactionECP01	ECP
testAddTransactionECP02	ECP
testAddTransactionECP03	ECP
testAddTransactionECP04	ECP

Método addToken

Nome	Tecnica
testAddTokenBVA01	BVA
testAddTokenBVA02	BVA
testAddTokenBVA03	BVA
testAddTokenECP01	ECP

Método getBlock

Nome	Tecnica
testGetBlockBVA01	BVA
testGetBlockBVA02	BVA
testGetBlockECP01	ECP

Método getBlockCount

Nome	Tecnica
testGetBlockCountECP01	ECP
testGetBlockCountECP02	ECP

Método registerTransactionInLedger

Nome	Tecnica
registerTransactionInLegderECP01	ECP
registerTransactionInLegderECP02	ECP

Método IsValidEdger

Nome	Tecnica
isValidEdgerECP01	ECP
isValidEdgerBVA01	BVA
isValidEdgerBVA02	BVA
isValidEdgerBVA03	BVA

Método BasicTransactionLine

Nome	Tecnica
testBasicTransactionLineBVA01	BVA
testBasicTransactionLineBVA02	BVA
testBasicTransactionLineBVA03	BVA
testBasicTransactionLineBVA04	BVA
testBasicTransactionLineBVA05	BVA
testBasicTransactionLineBVA06	BVA
TestBasicTransactionLineECP01	ECP

Método addTransactionLine/ getTotalValue

Nome	Tecnica
testGetTransactionCountECP01	ECP
testGetTotalValueECP01	ECP

5. Critérios de passagem ou falha das features

Classe	Nome	Passagem(Resultado obtido)	Falha(Resultado obtido)
BE	testAddTokenBVA01	IllegalArgumentException	!= IllegalArgumentException
BE	testAddTokenBVA02	0	!= 0
BE	testAddTokenBVA03	2147483647	!=2147483647
BE	testAddTokenECP01	3	!= 3
BE	testBasicEntityBVA01	IllegalArgumentException	!= IllegalArgumentException
BE	testBasicEntityBVA02	IllegalArgumentException	!= IllegalArgumentException
BE	testBasicEntityECP01	true	!= true
BT	testBasicTransactionBVA01	IllegalArgumentException	!= IllegalArgumentException
BT	testBasicTransactionECP01	true	!= true
BT	testAddTransactionLineBVA01	IllegalArgumentException	!= IllegalArgumentException
BT	testAddTransactionLineECP01	true	!= true
BT	testAddTransactionLineECP02	false	!=false
BT	testRemoveTransactionLineBVA01	IllegalArgumentException	!= IllegalArgumentException
BT	testRemoveTransactionLineBVA02	false	!=false
BT	testRemoveTransactionLineECP01	true	!= true
BT	testRemoveTransactionLineECP02	false	!=false
BT	testGetTransactionLineBVA01	UnHashableException	!= UnHashableException
BT	testGetTransactionLineECP01	true	!= true
BT	testGetTransactionLineECP02	null	!=null
BO	testAddTransactionBVA01	IllegalArgumentException	!= IllegalArgumentException
BO	testAddTransactionECP01	true	!= true
BO	testAddTransactionECP02	false	!=false
BO	testAddTransactionECP03	true	!= true
BO	testAddTransactionECP04	false	!=false
BO	testRemoveTransactionBVA01	IllegalArgumentException	!= IllegalArgumentException
BO	testRemoveTransactionECP01	true	!= true
BO	testRemoveTransactionECP02	false	!=false
BO	testRemoveTransactionECP03	false	!=false

BO	testGetTransactionBVA01	IllegalArgumentException	!= IllegalArgumentException
BO	testGetTransactionECP01	true	!= true
BO	testGetTransactionECP02	null	!=null
BO	testgetBlockCountECP01	2	!= 2
BO	testgetBlockCountECP02	1	!= 1
BO	testRegisterTransactionsInLedgerECP01	1	!= 1
BO	testRegisterTransactionsInLedgerECP02	0	!= 0
BO	testGetBlockBVA01	true	!= true
BO	testGetBlockBVA02	IllegalArgumentException	!= IllegalArgumentException
BO	testGetBlockBVA03	IllegalArgumentException	!= IllegalArgumentException
BO	testGetBlockECP01	true	!= true
BO	isValidEdgerBVA01	false	!=false
BO	isValidEdgerBVA02	false	!=false
BO	isValidEdgerBVA03	true	!= true
BO	isValidEdgerECP01	false	!=false
BTL	testBasicEntityBVA01	IllegalArgumentException	!= IllegalArgumentException
BTL	testBasicEntityBVA02	IllegalArgumentException	!= IllegalArgumentException
BTL	testBasicEntityBVA03	IllegalArgumentException	!= IllegalArgumentException
BTL	testBasicEntityBVA04	IllegalArgumentException	!= IllegalArgumentException
BTL	testBasicEntityBVA05	true	!= true
BTL	testBasicEntityBVA06	true	!= true
BTL	testBasicEntityECP01	true	!= true

Legenda Classes: BO- BasicOrganization, BTL-BasicTransactionLine, BE-BasicEntity;