Unitary test design

Nombre	Clase	Escenario
setup1	**HashTable**	**Add 4 customers to the database** **name= “Carlos Marti”** **ID: “253221”** **name= “Manuelo Coco”** **ID: “253331”** **name= “Pepe Willys”** **ID: “830173”** **name= “Alison Robert”** **ID: “984573”**
setup2	**HashTable**	**The database is empty.**
setup3	**Queue**	**Filled queue with 4 unregistered clients** **1. Name: “Alberto Aristizabal”** **ID: “1233450”** **2. Name: “Michael Martinez”** **ID: “1223344”** **3. Name: “Libertario Estratega ”** **ID: “6543210”** **4. Name: “Yotoko Matsutaka”** **ID: “6523410”**
setup4	**Queue**	**Filled queue with 2 registered clients** **1. Name: “Vallermo Azerduban”** **ID: “1233450”** **2. Name: “Martina Callizos”** **ID: “3652478”**
setup5	**Queue**	**Empty Queue.**
setup6	**Stack**	**Stack filled with 4 “undo” actions:** **name= “Carlos Marti”** **ID: “253221”** **priority: 0** **name= “Manuelo Coco”** **ID: “253331”** **priority: 0** **name= “Pepe Willys”**

		ID: “830173” **priority: 0** **name= “Alison Robert”** **ID: “984573”** **priority: 0**
setup7	**Stack**	**Empty Stack**
setup8	**HashTable**	**HashTable with 2 clients added with the following info:** **name= “Carlos Marti”** **ID: “253221”** **priority = 0** **name= “Manuelo Coco”** **ID: “253331”** **priority= 0**
setup9	**PriorityQueue**	**PriorityQueue with 3 clients of different priority** **name= “Carlos Marti”** **ID: “253221”** **priority = 20** **name= “Manuelo Coco”** **ID: “253331”** **priority= 1** **name= “Cumbia Marina”** **ID: “123456”** **priority = 5**
setup10	**PriorityQueue**	**Empty PriorityQueue.**

Test Cases Design

QUEUE

Class	Method	Scenario	Entry Values	Result
Bank	**enQueueTest**	**setup5**	**name = “ä§-↕à0↕”** **ID: “123456789”**	**Invalid input**

Class	Method	Scenario	Entry Values	Result

Bank	**enQueueTest**	**setup5**	**name = “David Montoya” ID: “123456789”**	**User added to the unregistered users queue.**

Class	Method	Scenario	Entry Values	Result
Bank	**enQueueTest**	**setup3**	**name = “Camilo Quiñonez” ID=”372904”**	**Exception, user cant be added because the queue is already full.**

Class	Method	Scenario	Entry Values	Result
Bank	**deQueueTest**	**setup5**	**none**	**Exception, the queue is empty**

Class	Method	Scenario	Entry Values	Result
Bank	**deQueueTest**	**setup3**	**none**	**Successful, it returns the user with name = “Alberto Aristizabal”**

Class	Method	Scenario	Entry Values	Result
Bank	**deQueueTest**	**setup3**	**none**	**dequeue all the users in the queue, and then validate if the queue is empty**

Class	Method	Scenario	Entry Values	Result
Bank	**isEmptyTest**	**setup5**	**none**	**False**

Class	Method	Scenario	Entry Values	Result
Bank	**isEmptyTest**	**setup3**	**none**	**True**

Class	Method	Scenario	Entry Values	Result
Bank	isEmptyTest	setup5	name=”Wilson Jaramillo” ID=”843965”	First check if the queue is empty, then enqueue the user, and check that the queue is not empty, finally dequeue and check that the queue is empty

Class	Method	Scenario	Entry Values	Result
Bank	peekTest	setup5	none	it returns null, because the queue is empty

Class	Method	Scenario	Entry Values	Result
Bank	peekTest	setup5	name = “Rodrigo Perez” ID = “463224”	Check the method returns null because the queue is empty, then enqueue the user and validate that it returns the only user, finally dequeue and check the null return.

Class	Method	Scenario	Entry Values	Result
Bank	peekTest	setup3	none	It returns the user with name = “Alberto Aristizabal”

STACK

Class	Method	Scenario	Entry Values	Result
Bank	pushStackTest	setup7	name= “Carlos Mora” ID = “346328”	User is stored in the stack

Class	Method	Scenario	Entry Values	Result
Bank	pushStackTest	setup6	none	Exception, the stack cant

				afford more elements.

Class	Method	Scenario	Entry Values	Result
Bank	**pushStackTest**	**setup7**	**name= “Carlos Mora” ID = “346328” name= “Raul Tare” ID = “432534”**	**The users are stored sucessfully.**

Class	Method	Scenario	Entry Values	Result
Bank	**popStackTest**	**setup6**	**none**	**Pops the user with name = “Alison Robert”**

Class	Method	Scenario	Entry Values	Result
Bank	**popStackTest**	**setup6**	**none**	**Pops the stack until it’s no longer full.**

Class	Method	Scenario	Entry Values	Result
Bank	**popStackTest**	**setup7**	**name=”rodrigo ramos” ID=”1212134”**	**Check the exception is thrown then add a costumer pop the stack and finally check the exception again.**

Class	Method	Scenario	Entry Values	Result
Bank	**peekTest**	**setup7**	**none**	**It returns null, because the stack is empty**

Class	Method	Scenario	Entry Values	Result
Bank	**peekTest**	**setup6**	**none**	**It returns the user with name = “Alison Robert”**

Class	Method	Scenario	Entry Values	Result
Bank	**peekTest**	**setup7**	**name = “Zion Lopez” ID = “534522”**	**Check the peek is null, then push the user and it returns the right user, finally pop the user and**

				validate the null return.

Class	Method	Scenario	Entry Values	Result
Bank	**isStackEmptyTest**	**setup6**	**none**	**False**

Class	Method	Scenario	Entry Values	Result
Bank	**isStackEmptyTest**	**setup7**	**none**	**True**

Class	Method	Scenario	Entry Values	Result
Bank	**isStackEmptyTest**	**setup6**	**none**	**Empties stack and then checks if it is empty, result should be true if the test is correct.**

HASHTABLE

Class	Method	Scenario	Entry Values	Result
Bank	**getTest**	**setup8**	**key**	**Checks for a key value equal to the given one in the HashTable and returns a boolean with the result of the search**

Class	Method	Scenario	Entry Values	Result
Bank	**getTest**	**setup8**	**key**	**Checks for a key value equal to the given one in the HashTable and returns a boolean with the result of the search**

Class	Method	Scenario	Entry Values	Result
Bank	**getTest**	**setup8**	**name = “Alberto Caranchoa”** **id = “123456”**	**Checks for a key value equal to the given one in the HashTable and returns a boolean with the**

			priority = 3 key	result of the search

Class	Method	Scenario	Entry Values	Result
Bank	addElementTest	setup8	name = “Alberto Caranchoa” id = “123456” priority = 3	Adds the element into the HashTable using linear probing as its search method.

Class	Method	Scenario	Entry Values	Result
Bank	addElementTest	setup8	name = “Alberto Caranchoa” id = “123456” priority = 3	Adds the element into the HashTable using linear probing as its search method.

Class	Method	Scenario	Entry Values	Result
Bank	addElementTest	setup8	name = “Amigable Albera” id = “325148” priority = 0 name = “Martino Espinoza” id = “123455” priority = 2 name = “Palermo Vidal” id = “354128” priority = 3	Adds the element into the HashTable using linear probing as its search method.

Class	Method	Scenario	Entry Values	Result
Bank	removeElementTest	setup8	key	Removes an element from the HashTable

Class	Method	Scenario	Entry Values	Result
Bank	removeElementTest	setup8	name= “Carlos Marti” id: “253221” priority = 0 name = “Palermo Vidal” id = “354128” priority = 3	Removes an element from the HashTable

Class	Method	Scenario	Entry Values	Result
Bank	removeElementTest	setup8	name= “Carlos Marti” id: “253221” priority = 0 name = “Palermo Vidal” id = “354128” priority = 3	Removes an element from the HashTable

PRIORITYQUEUE

Class	Method	Scenario	Entry Values	Result

Bank	**insertValueTest**	**setup10**	**name= “Padrino”** **id: “353622”** **priority = 20**	**Inserts a value to the PriorityQueue**

Class	Method	Scenario	Entry Values	Result
Bank	**insertValueTest2**	**setup9**	**name= “Marino”** **id: “353622”** **priority = 50**	**Inserts a value to the PriorityQueue**

Class	Method	Scenario	Entry Values	Result
Bank	**maximumTest**	**setup9**		**Gets the maximum value of the PriorityQueue**

Class	Method	Scenario	Entry Values	Result
Bank	**maximumTest2**	**setup10**		**adds a client in the empty priorityQueue and extracts it**

Class	Method	Scenario	Entry Values	Result
Bank	**extractMax**	**setup10**		**adds a client in the empty priorityQueue, extracts it and tries to extract again catching an exception in the process**

Class	Method	Scenario	Entry Values	Result
Bank	**extractMax2**	**setup9**		**Extracts the maximum priority client of the list**

Class	Method	Scenario	Entry Values	Result
Bank	**extractMax3**	**setup10**		**adds a client in the empty priorityQueue and extracts it**