

1. Functional requirements

Name:	R. #1. Assign turns at the time of admission
Summary:	Assign turns to users with name and ID data
Input:	String name, int id
Results:	Turn assigned to user
	Incomplete data, turn not assigned

Name:	R. #2. Place a user in the queue
Summary:	Place a user in the queue. There are two queue, the customer and the priority
Input:	
Results:	User located in queue

Name:	R. #3. Search a user in data base
Summary:	Search efficiently a user in data base to get information about user before it reaches the office where it will be attended
Input:	int id
Results:	User was found
	User wasn't found

Name:	R. #4. Change the amount of the customer's saving account
Summary:	Change the amount of the customer's saving account when requesting a withdrawal or consignment
Input:	int id
Results:	Changed the amount of the customer
	User does not have a saving account

Name:	R. #5. Cancellation of a customer's account
Summary:	Delete client data from database and add it to an exclusive database for those who desert
Input:	
Results:	Changed the amount of the customer
	User does not have a saving account

Name:	R. #6. Card payment
Summary:	User can pay amount used by credit card until now. Payment can be by cash or saving account
Input:	int id
Results:	Paid card
	User does not have debts

Name:	R. #7. Undo functionality
Summary:	Undo functionality works to fix mistakes, even after save those
Input:	int id
Results:	Paid card
	User does not have debts

2. Non-functional requirements

- Implement a ordering method for each of the chosen parameters, with the restriction that only one of them can have average temporal complexity of n^2
- Users information must be displayed in a spreadsheet-type table
- Must have a graphical interface to display users information and make operations, spreadsheet-type table and status of queues

3. Test documentation

Nombre	Clase	Escenario
setUp1	CashierModuleTest	Un objeto de la clase CashierModule sin clientes
setUp2	CashierModuleTest	<p>Un objeto de la clase CashierModule con 4 clientes con atributos:</p> <p>name = "Andrea" id = 001 genero = true condición especial = 0 monto = 322 cupo = 0 fecha = "29/09/2020"</p> <p>name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"</p> <p>name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020"</p> <p>name = "Aristizabal" id = 666 genero = true condición especial = 0 monto = 0 cupo = 0 fecha = "29/09/2020"</p>

Objetivo de la prueba: Determinar si el método signUpClient agrega correctamente a un cliente nuevo

Clase	Método	Escenario	Valores de Entrada	Resultado
CashierModuleTest	signUpClient	signUp1	<p>Objeto cliente con los atributos</p> <p>name = "Andrea"</p> <p>id = 001</p> <p>genero = true</p> <p>condición especial = 0</p> <p>monto = 322</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p> <p>name = "Danna"</p> <p>id = 257</p> <p>genero = true</p> <p>condición especial = 2</p> <p>monto = 2</p> <p>cupo = 3</p> <p>fecha = "29/09/2020"</p> <p>name = "Camilo"</p> <p>id = 999</p> <p>genero = false</p> <p>condición especial = 0</p> <p>monto = 10000</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p> <p>name = "Aristizabal"</p> <p>id = 666</p> <p>genero = true</p> <p>condición especial = 0</p> <p>monto = 0</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p>	Los clientes se agregan correctamente

Objetivo de la prueba: Determinar si el método unifyClient unifica correctamente y de manera ordenada los clientes de las diferentes particiones de la base de datos en el ArrayList

Clase	Método	Escenario	Valores de Entrada	Resultado
CashierModuleTest	unifyClientTest	signUp2	<p>Objeto cliente con los atributos</p> <p>name = "Andrea"</p> <p>id = 001</p> <p>genero = true</p> <p>condición especial = 0</p> <p>monto = 322</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p> <p>name = "Danna"</p> <p>id = 257</p> <p>genero = true</p> <p>condición especial = 2</p> <p>monto = 2</p> <p>cupo = 3</p> <p>fecha = "29/09/2020"</p> <p>name = "Camilo"</p> <p>id = 999</p> <p>genero = false</p> <p>condición especial = 0</p> <p>monto = 10000</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p> <p>name = "Aristizabal"</p> <p>id = 666</p> <p>genero = true</p> <p>condición especial = 0</p> <p>monto = 0</p> <p>cupo = 0</p> <p>fecha = "29/09/2020"</p> <p>Un objeto ArrayList de tipo cliente sin ningún cliente</p>	Los clientes se unifican correctamente y de forma ordenada en el ArrayList

Objetivo de la prueba: Determinar si el método search busca correctamente a un cliente

Clase	Método	Escenario	Valores de Entrada	Resultado
CashierModuleTest	searchTest	signUp2		El cliente se busca de manera correcta y retorna el cliente que se buscaba

Objetivo de la prueba: Determinar si el método deleteClientAccount elimina correctamente a un cliente

Clase	Método	Escenario	Valores de Entrada	Resultado
CashierModuleTest	deleteClientAccountTest	signUp2		El cliente se elimina correctamente y se elimina el cliente que se necesitaba eliminar

Nombre	Clase	Escenario
setUp1	QueueModuleTest	Un objeto de la clase QueueModuleTest sin usuarios
setUp2	QueueModuleTest	<p>Un objeto de la clase QueueModuleTest con 3 usuarios con atributos:</p> <p>name = "Andrea" id = 001 genero = true condición especial = 0 monto = 322 cupo = 0 fecha = "29/09/2020"</p> <p>name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"</p> <p>name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020"</p>

Objetivo de la prueba: Determinar si el método receivePerson recibe correctamente un usuario y lo asigna en la fila que le corresponde dependiendo de su condición especial

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueModuleTest	receivePersonTest	setUp1		Los usuarios se reciben correctamente en las filas que le corresponden dependiendo de su condición especial

Nombre	Clase	Escenario
setUp1	QueueTest	Un objeto de la clase QueueTest con un solo usuario con los atributos name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"
setUp2	QueueTest	Un objeto de la clase QueueTest con 2 usuarios con atributos: name = "Andrea" id = 001 genero = true condición especial = 0 monto = 322 cupo = 0 fecha = "29/09/2020" name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"

Objetivo de la prueba: Determinar si el método dequeueTestLimits desencola correctamente un usuario en el caso límite de que la cola no tenga ningún usuario

Clase	Método	Escenario	Valores de Entrada	Resultado
queueTest	dequeueTestLimits	setUp1		El método controla correctamente la Excepcion cuando se intenta desencolar un elemento que es null

Objetivo de la prueba: Determinar si el método dequeue desencola correctamente un usuario

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueTest	dequeueTestCaseBase	setUp2		El método desencola correctamente el usuario que se le solicita

Objetivo de la prueba: Determinar si el método enqueue atrapa correctamente la Excepcion al intentar encolar un elemento nulo

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueTest	enqueueTestInterestingCase	setUp1		El método controla correctamente la Excepcion

Objetivo de la prueba: Determinar si el método enqueue verifica que el último método encolado sea el último elemento en la fila correctamente

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueTest	enqueueLastTest	setUp1	Objeto usuario con los atributos name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020"	El método controla correctamente la Excepcion

Objetivo de la prueba: Determinar si el método isEmpty retorna correctamente si la cola está vacía

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueTest	isEmptyTest	setUp1		El método determina correctamente que la cola esta vacía

Objetivo de la prueba: Determinar si el método size retorna correctamente el tamaño de la cola

Clase	Método	Escenario	Valores de Entrada	Resultado
QueueTest	sizeTest	setUp1		El método retorna correctamente el tamaño de la cola

Nombre	Clase	Escenario
setUp1	HashTableTest	Un objeto de la clase MyHashTable sin clientes
setUp2	HashTableTest	Un objeto de la clase MyHashTable con 1 clientes con atributos: name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020"
setUp3	HashTableTest	Un objeto de la clase MyHashTable con 2 clientes con atributos: name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020" name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"

Objetivo de la prueba: Determinar si el método isEmpty retorna correctamente si la hash table está vacía

Clase	Método	Escenario	Valores de Entrada	Resultado
HashTableTest	isEmptyTest	setUp1		El método determina correctamente que la hash table esta vacía
HashTableTest	isEmptyTest2	setUp2		El método determina correctamente que la hash table no esta vacía

Objetivo de la prueba: Determinar si el método get retorna correctamente el elemento en la posición solicitada

Clase	Método	Escenario	Valores de Entrada	Resultado
HashTableTest	getTest	setUp3		El método retorna correctamente el cliente que se le solicita
HashTableTest	getTestInterestingCase	setUp1		El método retorna null ante un elemento que no existe dado que la hash table se encuentra vacía

Objetivo de la prueba: Determinar si el método remove elimina correctamente el elemento solicitado en la hash table

Clase	Método	Escenario	Valores de Entrada	Resultado
HashTableTest	removeTestSpecialCase	setUp1		El método determina correctamente si al eliminar en una hash table vacía retorna null
HashTableTest	removeTestBaseCase	setUp2		El método elimina correctamente el cliente que se solicita de la tabla hash

Objetivo de la prueba: Determinar si el método add agrega correctamente los clientes

Clase	Método	Escenario	Valores de Entrada	Resultado
HashTableTest	addTest	setUp3		El método agrega correctamente el cliente que se solicita
HashTableTest	addTestInterestingCase	setUp1		El método agrega correctamente el cliente que se solicita

Nombre	Clase	Escenario
setUp	MyDoublyLinkedListTester	<p>Tres objetos de la clase MyDoublyLinkedListTester con 3 clientes con los siguientes atributos:</p> <p>name = "Andrea" id = 001 genero = true condición especial = 0 monto = 322 cupo = 0 fecha = "29/09/2020"</p> <p>name = "Danna" id = 257 genero = true condición especial = 2 monto = 2 cupo = 3 fecha = "29/09/2020"</p> <p>name = "Camilo" id = 999 genero = false condición especial = 0 monto = 10000 cupo = 0 fecha = "29/09/2020"</p>

Objetivo de la prueba: Determinar si el método get retorna correctamente el elemento que se le pide

Clase	Método	Escenario	Valores de Entrada	Resultado
MyDoublyLinkedListTester	testGet	setUp		El método retorna correctamente el elemento que se le pide

Objetivo de la prueba: Determinar si el método remove elimina correctamente el elemento que se le pide

Clase	Método	Escenario	Valores de Entrada	Resultado
MyDoublyLinkedListTester	testRemove	setUp		El método elimina correctamente el elemento que se le pide

Objetivo de la prueba: Determinar si el método size retorna correctamente el tamaño de la lista

Clase	Método	Escenario	Valores de Entrada	Resultado
MyDoublyLinkedListTester	testSize	setUp		El método retorna correctamente el tamaño de la lista

Nombre	Clase	Escenario
setUp1	PriorityQueueHeapTest	Un objeto de la clase MyPriorityQueueHeap sin usuarios
setUp2	PriorityQueueHeapTest	<p>Un objeto de la clase MyPriorityQueueHeap con 4 usuarios con atributos:</p> <p>name = "Andrea" id = 100 genero = true condición especial = 0</p> <p>name = "Danna" id = 2 genero = true condición especial = 2</p> <p>name = "Camilo" id = 4 genero = false condición especial = 2</p> <p>name = "Cristhian" id = 6 genero = false condición especial = 1</p>
setUp3	PriorityQueueHeapTest	<p>Un objeto de la clase MyPriorityQueueHeap con 3 clientes con atributos:</p> <p>name = "Andrea" id = 100 genero = true condición especial = 0</p> <p>name = "Danna" id = 2 genero = true condición especial = 2</p> <p>name = "Camilo" id = 4 genero = false condición especial = 2</p>

Objetivo de la prueba: Determinar si el método enqueue agrega correctamente los usuarios

Clase	Método	Escenario	Valores de Entrada	Resultado
PriorityQueueHeap Test	enqueueTest1	setUp1	Objeto de la clase Usuario con los atributos: name = "Andrea" id = 100 genero = true condición especial = 0	El método agrega correctamente el usuario que se solicita
PriorityQueueHeap Test	enqueueTest2	setUp2	Objeto de la clase usuario con los atributos name = "Pinina" id = 4 genero = false condición especial = 3 name = "Andrea" id = 100 genero = true condición especial = 0	El método agrega correctamente el cliente que se solicita

Objetivo de la prueba: Determinar si el método dequeue agrega correctamente los usuarios

Clase	Método	Escenario	Valores de Entrada	Resultado
PriorityQueueHeap Test	dequeueTest1	setUp1		El método retorna correctamente null al solicitarse desencolar en una fila vacía
PriorityQueueHeap Test	dequeueTest2	setUp3		El método desencola correctamente el usuario que se solicita