Design of the test cases

Test Objective: Verify that a game is correctly added to the Game store

Class	Method	Scenary	Input	Output
Game	addGame	setupScenary 1	Name: Fifa21 review: Soccer. price: 45000 Shelve name: B Amount: 2	They were added with 2 games with the name "Fifa 21". Price 45000. And the name of the shelf is "B"
	Setters and getters	setupScenary 1	Name:Fifa 21 Review:Soccer Price: 45000 Shelve name: B Amount: 2	setters and getters work fine

Test Objective: Verify that a shelve is correctly added to the Game store

Class	Method	Scenary	Input	Output
shelf	addShelf	setupScenary 1	Shelve name: B Size Shelve slots: 2	Shelf "B" 2 spaces successfully added to the play store.
	Setters and getters	setupScenary 1	Shelve name: B Size Shelve slots: 2	setters and getters work fine

Test Objective: Verify that a costumer is correctly added to the Game store

Class	Method	Scenary	Input	Output
Custumer	Custumer	setupScenary 1	Id: 1237 Name: juan Code: 001 Wishlist Fifa 21, far cry	The client "Juan" with id 1237, code 001 and his wish list fifa 21 and far cry, has entered
	Setters and getters	setupScenary 1	Id: 1237 Name: juan Code: 001 Wishlist Fifa 21, far cry	setters and getters work fine

Test Objective: Verify if one shelve doesn't exist in the game store

Class	Method	Scenary	Input	Output
GameStore	getShelfLocation	setupScenary 1	Shelve name: b	The shelve name "b" already exist in the game store,

Test Objective: Search for the customer who has entered the store

Class	Method	Scenary	Input	Output
GameStore	searchCostumer	setupScenary 1	Name: juan	The costumer “juan” was found

Test Objective: amount of cashiers in the play store.

Class	Method	Scenary	Input	Output
GameStore	amountCashiers	setupScenary 1	Null	The amount of the cashiers are “0”.

Test Objective: Add element to the hash table

Class	Method	Scenary	Input	Output
HashTable	insert	setupScenary 1	new Game(1,"Warzone ",”War”,20000,"C",2);	The game1 was added successfully.

Test Objective: Remove element to the hash table

Class	Method	Scenary	Input	Output
HashTable	delete	setupScenary 1	new Game(1,"Warzone ",”War”,20000,"C",2);	The game1 was deleted successfully.

Test Objective: Get the values on the hash table

Class	Method	Scenary	Input	Output
HashTable	Search	setupScenary 1	new Game(1,"Warzone ",”War”,20000,"C",2);	Return the values of the game1.

Test Objective: Validate if the queue is empty.

Class	Method	Scenary	Input	Output
Queque	isEmpty()	setupScenary 1	None	The queue is empty.

Test Objective: Obtain the data of the element in one queue

Class	Method	Scenary	Input	Output
Queque	front()	setupScenary 1	costumer1 = new Costumer(5,"sebastian",”game1);	Return the value of the costumer1. .

Test Objective: Add one value to the stack.

Class	Method	Scenary	Input	Output
Stack	push()	setupScenary 1	game1= new Game(1,"Warzone ",”War”,20000,"c",2);	The game1 was added successfully to the stack. .

Test Objective: Delete one value to the stack

Class	Method	Scenary	Input	Output
Stack	pop()	setupScenary 1	game1	The game1 was deleted successfully. Return game1.

Test Objective: Validate if the stack is empty.

Class	Method	Scenary	Input	Output
Stack	isEmpty()	setupScenary 1	None	The stack is empty.

Test Objective: Obtain the data of the element in one stack.

Class	Method	Scenary	Input	Output
Stack	top()	setupScenary 1	game1= new Game(1,"Warzone ",”War”,20000,"C",2);	Return the value of the game1.

Test Objective: Add one value to the queue.

Class	Method	Scenary	Input	Output
Queque	enqueue()	setupScenary 1	new Client (1, "Daniel", ”zula, mario bros”);	has been added

Test Objective: Delete one value to the queue

Class	Method	Scenary	Input	Output
Queque	dequeue()	setupScenary 1	costumer1	Sebastian 2552 21351351 : Customer 1. has been deleted

Test Objective: Obtain the last element in the queue

Class	Method	Scenary	Input	Output
Queque	Front	setupScenary 1		new Client (1, "Daniel", ”zula, mario bros”);

Test Objective: Search one element in the queue.

Class	Method	Scenary	Input	Output
Queque	search	setupScenary 1	new Client (1, "Daniel", ”zula, mario bros”);	True

Testing two test cases

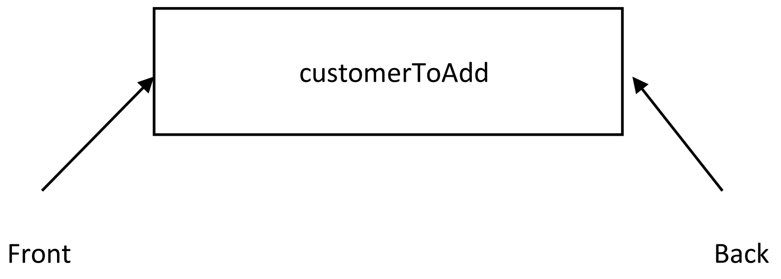
Queue:

Test Objective: Add one value to the queue.

Class	Method	Scenary	Input	Output
Queque	enqueue()	setupScenary 1	new Client (1, "Daniel", ”zula, mario bros”);	has been added

Step by Step

For the resolution of this case, the Queue class must first be initialized with the setupScenary method that invokes the constructor of this class, then within the method that will test the method to add element to the queue, a new type of Client object is created, this will contain the values contained in the Input box, then this object is added to the customer queue with the enqueue (customerToAdd) method, the front of the queue and the back would be updated and would point to this first and only element for the moment added and internally the tail would be affected in this way

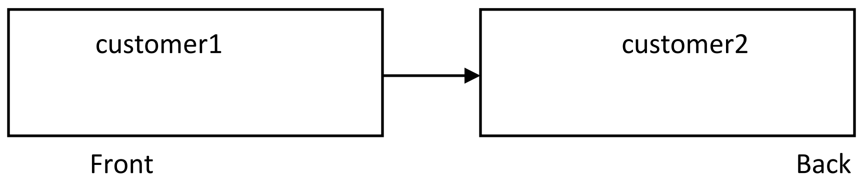


Test Objective: Delete one value to the queue

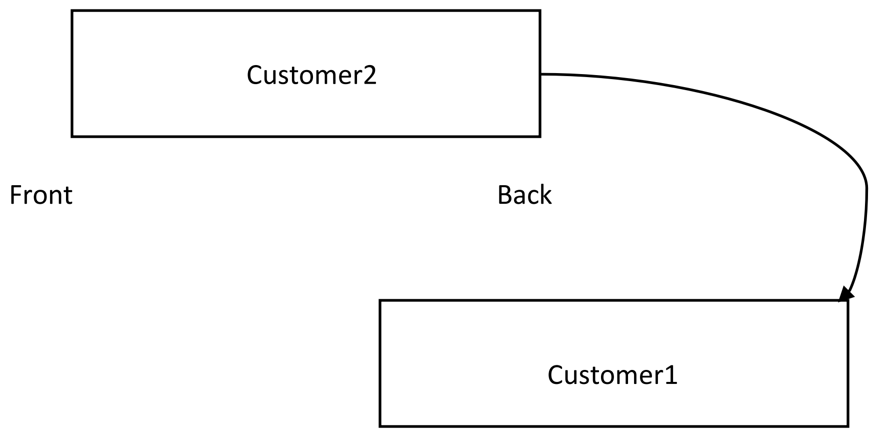
Class	Method	Scenary	Input	Output
Queque	dequeue()	setupScenary 1	costumer1	Sebastian 2552 21351351 : Customer 1. has been deleted

Step by Step

Following the previous process, to test this method we proceed to add two clients, these would be the following



Customer 1 was added first and after this, customer 2, now we proceed to invoke the dequeue method which what it does is eliminate the object that is at the beginning, it returns this and also updates the pointer called front so that its next is the new front, this operation performed on this queue of customers would result in the following graphic

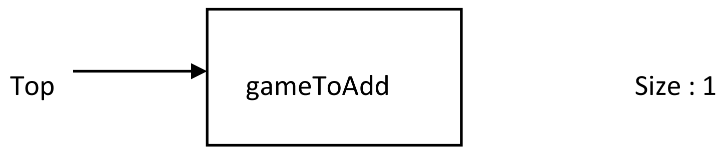


Stack:

Test Objective: Add one value to the stack.				
Class	**Method**	**Scenary**	**Input**	**Output**
Stack	push()	setupScenary 1	game1= new Game(1,"Warzone","War",20000,"c",2);	The game1 was added successfully to the stack. .

Test

To test this test case, first an element is added to the stack, for this the setupScenary that contains the constructor method of the Stack class is invoked, after that an object of type Game is created with the attributes described in Input, once this is done, the stack is used to invoke the method called push and the previously created object of type Game (push (gameToAdd)) is passed to it in its parameters, the pointer called Top is updated by the new object and the variable is updated size that indicates the current size of the stack, graphically this would be the representation



Test Objective: Validate if the stack is empty.

Class	Method	Scenary	Input	Output
Stack	isEmpty()	setupScenary 1	None	The stack is empty.

Test:

We proceed to create a new stack in which no type of object will be added, this will be done by invoking the setupScenary1 method that contains the initialization of the constructor method of the Stack class, once this process is carried out, the isEmpty () method is invoked belonging to the Stack class, this method returns true if the stack is empty and it will return false if the list has 1 element or more, in this case as the stack is empty the method will return true because top = null

$$\begin{tikzcd}\text{Top} \arrow[r] & \text{null}\end{tikzcd}$$

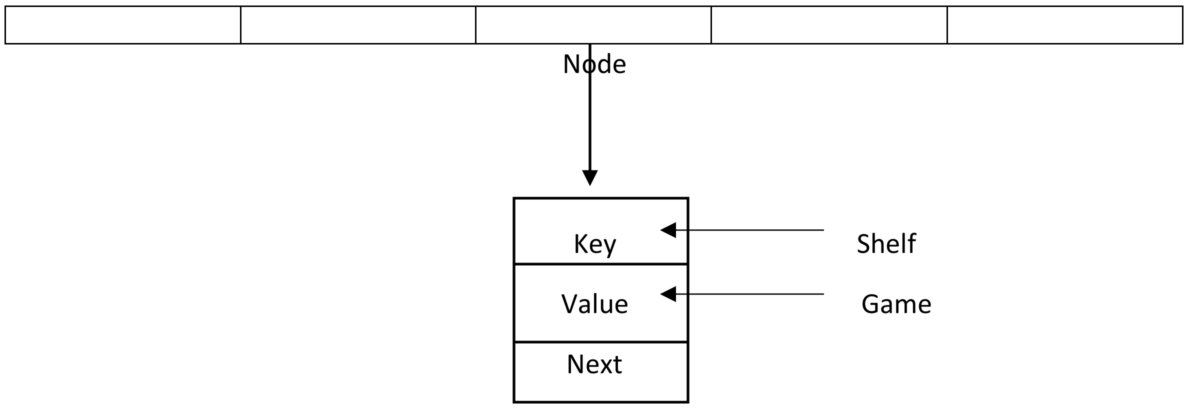
HashTable

Test Objective: Add element to the hash table

Class	Method	Scenary	Input	Output
HashTable	insert	setupScenary 1	new Game(1,"Warzone ",”War”,20000,"C",2);	The game1 was added successfully.

Test

To test this method we first proceed to create a game type object that will contain the attributes shown in input, in addition to that, we proceed to create a Shel type object that will be the shelf where the game will be contained, this object will be the key and the set will be the value of this key, after that an object of type HashTable is created invoking the setupScenary method that contains the creation of the HashTable class (invocation constructor method) after that, we proceed to invoke the insert method of the HashTable class and this element is added, internally, the calculation is performed with hashFunction to know in which position this object will be inserted, the hashTable will have a fixed value and the object will be inserted depending on the key (Shelf), the hash function is obtain the hash value of the key and to this apply the mod operator of the size of the hashTable to proceed to insert it in the array, graphically it would look like this



Test Objective: Remove element to the hash table

Class	Method	Scenary	Input	Output
HashTable	delete	setupScenary 1	new Game(1,"Warzone ",”War”,20000,"C",2);	The game1 was deleted successfully.

Test:

With the same previous procedure, two objects are created, a key that will be of type Shelf and a value that will be of type Game, these two are inserted in the HashTable and are located in the same way since they will contain the same values, to eliminate This same pair of objects proceeds to enter a key to the delete method of the HashTable, this key must coincide with the key of the object already created in the table and thus eliminate both the key of said object and the value of this