Test Cases Integrative Task 1

Samuel Navia Quiceno (A00405006)
Simón García (A00371828)
Juan Camilo Criollo Cifuentes(A00402515)

Test Cases for HashTable structure:

Scenario configurations:

Name	Class	Scenario
defaultTest	HashTable Test()	hashTable=new HashTable

Test Cases Design

Test add method

Test's objective: The objective is to evaluate and determine the limits that the add method has, so that when testing this it reacts in an expected way either when throwing an exception (when we break the limit of the method) or when we use a correct operation of this one.				
Class	Method	Scenario	Input	Expected Output
HashTable	add()	defaulTest()	hashTable.add("xry",5);	5

HashTable	add()	defaultTest()	hashTable.add("xry",0);
			hashTable.add("rxy",0);
			hashTable.add("xsx",0);
			hashTable.add("rxy",0);	

Test delete method

Test's objective: The objective of this test is to verify that the implementation of the delete method works correctly inside the hashTable even if we exceed the limits of this method.				
Class	Method	Scenario	Input	Expected Output
HashTable	delete()	defaulTest()	hashTable.add("xry",5);	null

			hashTable.add("rxy",5);
			hashTable.add("xsx",5);
			hashTable.delete("xry");	

HashTable	delete()	defaultTest()	hashTable.add("xry",5);
			hashTable.delete("xry");
			hashTable.delete("xry");
				HashTable Exception

HashTable	delete()	defaultTest()	hashTable.add("xry",5);
			hashTable.add("rxy",5);
			hashTable.add("xsx",5);
			hashTable.add("rxz",5);
			hashTable.delete("xry");
			hashTable.delete("rxy");
			hashTable.delete("xsx");
			hashTable.delete("rxz");
				null

Test Search method

Test's objective: The objective of this test of the search method is to verify that it works in a proper way, in such a way that when searching for a key it finds its value correctly, as in other cases where it does not find the key either because it does not exist or because of some other case.				
Class	Method	Scenario	Input	Expected Output
HashTable	search()	defaulTest()	hashTable.add("xry",5);	5
HashTable	search()	defaultTest()	hashTable.add("xry",5);	null

			hashTable.delete("xry");	

HashTable	search()	defaultTest	hashTable.add("xry",5);
			hashTable.add("rxy",7);
			hashTable.add("xsx",5);
				7

Test Cases for Stack structure:

Scenario configurations:

Name	Class	Scenario
defaultTest()	Stack	stack=new Stack

Test Cases Design

Test push method

Test's objective: The purpose of testing the push method in the Stack class is that in this way we can verify its correct operation respecting the logic of the stack itself first in last out, using this we use the top method to observe the operation of push either in a successful or unsuccessful case.				
Class	Method	Scenario	Input	Expected Output

Stack	push()	defaultTest()	stack.push(1);
			stack.push(2);
			stack.push(3);
				3

Stack	push()	defaultTest()	stack.push(null);	null
Stack	push()	defaultTest()	for(int i=0;i<999;i++){ stack.push(4+i);	777

			}
			
			stack.push(777);
			int value=
			stack.top();	

Test pop method

Test's objective: The purpose of the pop test method is to verify the correct functioning of the test, as well as to have different cases when the user enters a correct value or not.				
Class	Method	Scenario	Input	Expected Output

Stack	pop()	defaultTest()	stack.push(-1);
			stack.push(2);
			stack.pop();
				-1

Stack	pop()	defaultTest()	stack.push(i); from i=0 to i=100 stack.pop(i); from i=0 to i=99	0

Stack	pop()	defaultTest()	stack.push(-1);
			stack.push(2);
			
			stack.push(2450000);
			stack.pop();
			stack.pop();
			stack.pop();
			stack.pop();
				StackException
				e.message="The Stack is empty."

Test top method

Test's objective: The purpose of the top method test is to verify that the last method added, or rather the top, is shown, so that we can use it in the rest of the methods as a verification of the previous ones.				
Class	Method	Scenario	Input	Expected Output
Stack	top()	defaultTest()	stack.push(-1); stack.push(2);	-1

			stack.push(2450000);
			stack.top(); stack.pop();
			stack.pop();
			int value=stack.top();	

Stack	top()	defaultTest()	stack.push(-1);
			stack.pop();
			stack.top();
				StackException
				message=e.message
				"Nothing is saved in the stack.

Stack	top()	defaultTest()	stack.push(-1);
			stack.push(5);
			int value= stack.top();
			
			Assert.assertEquals(5,value);
			stack.pop();
			int value2= stack.top();
			
			Assert.assertEquals(-1, value2);
				5
				-1

Test Cases for Queue structure:

Scenario configurations:

Name	Class	Scenario
defaultTest()	Queue	queue=new Queue

Test Cases Design

Test enqueue method

Test's objective: The purpose of the enqueue method is to verify that an element is added correctly into the Queue, including possible exceptions.				
Class	Method	Scenario	Input	Expected Output
Queue	enqueue()	defaultTest()	queue.enqueue(1);	1

			queue.enqueue(2);
			
			queue.enqueue(3); int value= queue.front();	

Queue	enqueue()	defaultTest()	queue.enqueue(i) from i=1 to i=0 queue.front();	1
Queue	enqueue()	defaultTest()	queue.enqueue(null); queue.front();	null

Test dequeue method

Test's objective: The objective of doing this test of the dequeue method is to verify that it returns the correct value of the Queue and that this same one is eliminated of the Queue, without breaking the logical and theoretical operation.				
Class	Method	Scenario	Input	Expected Output
Queue	dequeue()	defaultTest()	queue.enqueue(1); queue.enqueue(2); queue.enqueue(3); int value=queue.dequeue();	1
Queue	dequeue()	defaultTest()	queue.enqueue(1); queue.dequeue(); int value=queue.dequeue();	1
Queue	dequeue()	defaultTest()	queue.enqueue(null); queue.dequeue();	null

Test empty method

Test's objective: The objective of this test method is verify that the isEmpty method works correctly when adding or removing products, so this method is used in enqueue and dequeue.

Class	Method	Scenario	Input	Expected Output
Queue	isEmpty()	defaultTest()	queue.isEmpty();	true
Queue	isEmpty()	defaultTest()	queue.enqueue(1); queue.isEmpty();	false

Queue	isEmpty()	defaultTest()	queue.enqueue(1);
			queue.enqueue(2);
			queue.enqueue(3);
			queue.enqueue(4);
			queue.enqueue(5);
			int value=queue.dequeue()+queue.dequeue()+queue.dequeue()+queue.dequeue();
				10

Test cases for the controller:

Scenario configurations:

Name	Class	Scenario
defaultTest()	Controller	controller=new Controller

Test addCustomer Method

Test's objective:The purpose of testing the addCustomer method in the CustomerController class is to verify the correct operation of adding a customer, ensuring that valid customers are saved properly, and that the system handles cases such as empty inputs and duplicate customers appropriately.

Class	Method	Scenario	Input	Expected Output
CustomerController	addCustomer()	addCustomerSuccessTest()	“John Doe”, “123 Elm Street”, “johndoe@example.com”,”password123”	"Saved customers:\n Email: johndoe@example.com - Name: John Doe"
CustomerController	addCustomer()	addEmptyCustomer	"", "", "", ""	"Saved customers:\n Email: - Name: "
CustomerController	addCustomer()	addExistingCustomerTest()	"Jane Doe", "456 Oak Avenue", "johndoe@example.com", "anotherpassword"	"Customer already exists! Try another email."

Test createProduct Method

The goal of testing the createProduct method in the ProductController class is to confirm that products are added correctly to the system, while also checking for scenarios where the product code already exists or the input values are empty, ensuring the proper error handling.				
Class	Method	Scenario	Input	Expected Output
ProductController	createProduct()	createProductSuccessTest()	"P001", "Toy Car", "A small toy car", 19.99, 1	"Saved products:\nCode: P001 - Name: Toy Car"
ProductController	createProduct()	createProductExistingCodeTest()	"AA00", "Toy Car", "A small toy car", 19.99, 1	"Product already exists! Try another code."

ProductController	createProduct()	createEmptyProductTest()	"", "", "", 0, 0	"Saved products:\nCode: - Name: "

Test deleteProduct Method

The purpose of the deleteProduct test is to verify that the method correctly removes a product from the system when a valid code is provided and manages cases where the product does not exist or the input code is empty.				
Class	Method	Scenario	Input	Expected Output
ProductController	deleteProduct()	deleteProductSuccessTest()	"AA00"	"There are no registered products."
ProductController	deleteProduct()	deleteProductNonExistingTest()	"0000"	"Product not found! Try another code."
ProductController	deleteProduct()	deleteProductEmptyCodeTest()	“”	"There are no registered products to delete."

Test updateProduct Method

The purpose of testing the updateProduct method in the ProductController class is to validate the ability to update a product's code correctly, ensuring that valid updates are processed as expected. It also tests scenarios such as trying to update a non-existing product or using a code that is already in use, verifying the appropriate error handling.

Class	Method	Scenario	Input	Expected Output
ProductController	updateProduct()	updateProductNameSuccessTest()	(1, "AA00", "AAAA")	"Saved products:\nCode: AAAA - Name: Soap"
ProductController	updateProduct()	updateProductNonExistingCodeTest()	(1, "P999", "0000")	"Product was not found! Try another code."
ProductController	updateProduct()	updateProductPreexistingCodeTest()	(1, "AA00", "BB11")	"That product code is already in use. Try another one."

Test updateProduct Method

The objective of the test is to update the prices of a product with positive, negative and 0.

Class	Method	Scenario	Input	Expected Output
ProductController	updateProduct()	updateProductPriceSuccessTest()	("AA00", 100.0)	100.0 (price updated successfully)
ProductController	updateProduct()	updateProductNegativePriceTest()	("AA00", -100.0)	-100.0 (price updated to negative)
ProductController	updateProduct()	updateProduct0PriceTest()	("AA00", 0.0)	0.0 (price updated to zero)

Test updateProduct Method

Objective of the test is update the priority while checking the changes and also change the structural changes inside the priority queue.

Class	Method	Scenario	Input	Expected Output
ProductController	updateProduct()	updateProductPrioritySuccessTest()	("AA00", 5)	5 (priority updated successfully)
ProductController	updateProduct()	updateProductNegativePriorityTest()	("AA00", -1)	-1 (priority updated to negative)
ProductController	updateProduct()	updateProductPriorityChangeOrderTest()	("BB11", 5)	"BB11" (product remains in the correct order)

Test placeOrder Method

Objective of the test was to check the method to add orders.

Class	Method	Scenario	Input	Expected Output
OrderController	placeOrder()	placeOrderSuccessTest()	("AA00", 5)	5 (priority updated successfully)
OrderController	placeOrder()	placeOrderCustomerNotFoundTest()	("AA00", -1)	-1 (priority updated to negative)
OrderController	placeOrder()		("BB11", 5)	"BB11" (product remains in the correct order)