

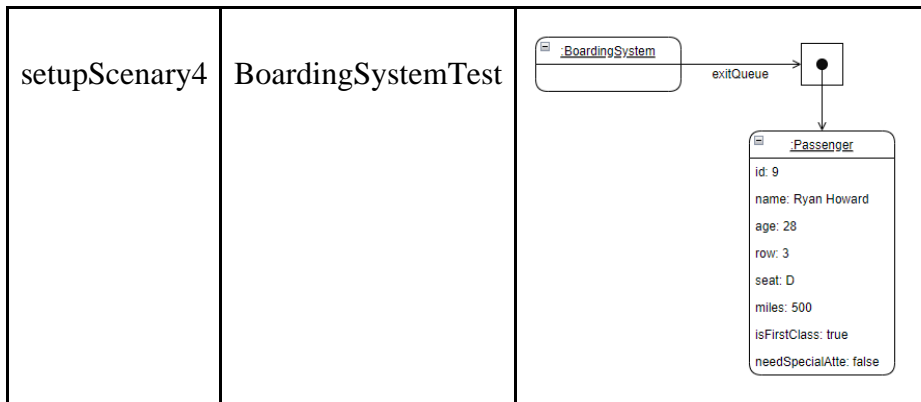
# Tarea Integradora I

Gloria Vanesa Vicuña - A00369332  
Ricardo Medina Sterling - A00369009  
Alejandro Osejo Ochoa - A00372469

# DESIGN

C. Design of test cases, including scenarios, for the structures and the system.  
**Configuration of BoardingSystem scenarios**

Name	Class	Scenery
setupScenary1	BoardingSystemTest	<pre>classDiagram     class BoardingSystem {     }     class Passenger {         id: 9         name: Ryan Howard         age: 28         row: 3         seat: D         miles: 500         isFirstClass: true         needSpecialAtte: false     }     BoardingSystem --&gt; Passenger : passengers</pre>
setupScenary2	BoardingSystemTest	<pre>classDiagram     class BoardingSystem {     }     class Passenger {         id: 9         name: Ryan Howard         age: 28         row: 3         seat: D         miles: 500         isFirstClass: true         needSpecialAtte: false     }     BoardingSystem --&gt; Passenger : arrivalQueue</pre>
setupScenary3	BoardingSystemTest	<pre>classDiagram     class BoardingSystem {     }     class Passenger {         id: 9         name: Ryan Howard         age: 28         row: 3         seat: D         miles: 500         isFirstClass: true         needSpecialAtte: false     }     BoardingSystem --&gt; Passenger : boardingQueue</pre>


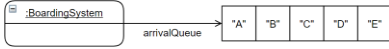
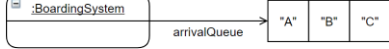



### Test Cases Design

Test objective: Test the correct operation of the BoardingSystem class.				
Class	Method	Scenery	Inputs	Resultado
BoardingSystem	loadPassengers	setupScenary1	filename= passengers.txt	Each passenger's information has been correctly stored in the HashTable.
BoardingSystem	getPassenger	setupScenary1	id= 9	The information corresponding to a passenger is obtained

BoardingSystem	addToArrivalQueue	setupScenary2	id= 9	A passenger's id is added to the arrival queue at the boarding lounge.
BoardingSystem	getBoardingQueue	setupScenary3	empty	A passenger's id is added to the boarding list.
BoardingSystem	getExitQueue	setupScenary4	empty	A passenger's id is added to the exit list.

### Configuration of QueueTest scenarios

Name	Class	Scenery
setupScenary1	QueueTest	 <pre> sequenceDiagram     participant BS as :BoardingSystem     participant AQ as arrivalQueue     BS-&gt;&gt;AQ:      AQ--&gt;&gt;BS: empty empty empty empty empty           </pre>
setupScenary2	QueueTest	 <pre> sequenceDiagram     participant BS as :BoardingSystem     participant AQ as arrivalQueue     BS-&gt;&gt;AQ:      AQ--&gt;&gt;BS: "A" "B" "C" "D" "E"           </pre>
setupScenary3	QueueTest	 <pre> sequenceDiagram     participant BS as :BoardingSystem     participant AQ as arrivalQueue     BS-&gt;&gt;AQ:      AQ--&gt;&gt;BS: "A" "B" "C"           </pre>
setupScenary4	QueueTest	 <pre> sequenceDiagram     participant BS as :BoardingSystem     participant AQ as arrivalQueue     BS-&gt;&gt;AQ:      AQ--&gt;&gt;BS: "A" "B" "C" "D"           </pre>

### Test Cases Design

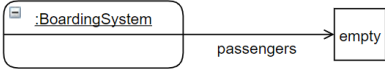
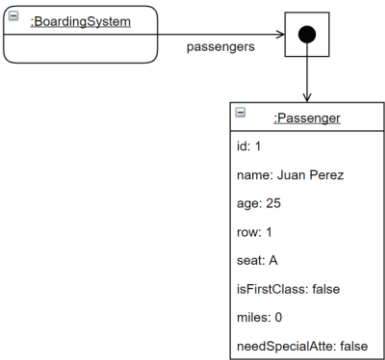
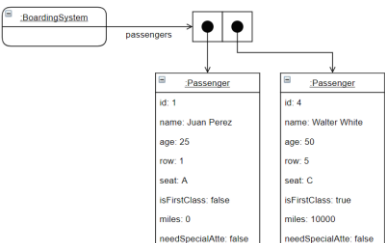
**Test objective:** Test the correct operation of the Queue class.

Class	Method	Scenery	Inputs	Result
Queue	enqueue	setupScenary1	data= "A"  data= "B"  data= "C"	The test must allow entering 3 Data which are in the queue
Queue	enqueue	setupScenary2	data= "A"  data= "B"  data= "C"  data= "D"  data= "E"	The test should allow data entry until the queue is full
Queue	enqueue	setupScenary1	data= "A"  data= "B"	The test must allow entering 2 Data which are in the queue
Queue	dequeue	setupScenary3	empty	The test must allow data to be removed from a queue that has data

Queue	dequeue	setupScenary1	empty	The test must remove data from a queue that is empty
Queue	dequeue enqueue	setupScenary4	data= "A"  data= "B"  data= "C"  data= "D"	The test should add data to a queue and remove data from the queue, as well as add other data to the queue
Queue	peek	setupScenary3	empty	The test is used to examine the next item in the queue.
Queue	peek	setupScenary1	empty	The test is used to examine the next item in the queue but it is an empty queue
Queue	peek enqueue	setupScenary2	data= "A"  data= "B"  data= "C"  data= "D"	The test is used to add data to the queue, also it is used to examine the next item in the queue but it is an empty queue

#### Configuration of HashTableTest scenarios

Name	Class	Scenery
------	-------	---------

setupScenary1	HashTableTest	
setupScenary2	HashTableTest	
setupScenary3	HashTableTest	

### Test Cases Design

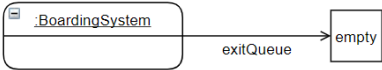
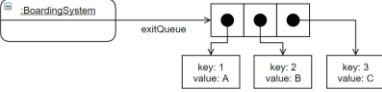
Test objective: Test the correct operation of the HashTable class.				
Class	Method	Scenery	Inputs	Resultado
HashTable	put	setupScenary1	key= 1  value= "Juan Perez", 25, 1, 'A', false, 0, false	The test serves to insert elements inside the Hashtable
HashTable	put	setupScenary1	key= 1  value= "Juan Perez", 25, 1, 'A', false, 0, false	The test serves to insert two elements into the Hashtable

			key= 4  value= "Walter White", 50, 5, 'C', true, 10000, false	
HashTable	put	setupScenary1	key= 1  value= "Juan Perez", 25, 1, 'A', false, 0, false  key= 4  value= "Walter White", 50, 5, 'C', true, 10000, false  key= 5  value= "Jesse Pinkman", 30, 2, 'B', true, 5000, false	The test serves to insert three elements into the Hashtable
HashTable	get	setupScenary2	key= 1	The test is used to get the value in the HashTable
HashTable	get	setupScenary1	key= 1	The test is used to obtain the value in the HashTable but it looks for an element that is not in the HashTable
HashTable	get	setupScenary3	key= 1  key= 4	The test verifies that the returned object is equal to the second Passenger object inserted into the table, which is what you



				would expect to get when searching for its id.
HashTable	remove	setupScenary2	key= 1	The test is used to remove a value from the HashTable
HashTable	remove	setupScenary1	key= 1	The test is used to remove a value from the HashTable but that value is not found in the table
HashTable	remove	setupScenary3	key= 1 key= 4	It is a test for a HashTable in which two colliding objects are inserted at the same table index. Then, an attempt is made to delete one of the objects using its key, and it is verified that the deleted object is the one that was expected.

### Configuration of MinPriorityQueueTest scenarios

Name	Class	Scenery
setupScenary1	MinPriorityQueueTest	
setupScenary2	MinPriorityQueueTest	

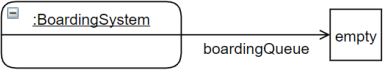
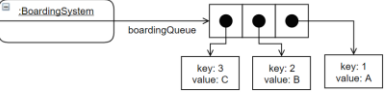
### Test Cases Design

**Test objective:** Test the correct operation of the MinPriorityQueue class.

Class	Method	Scenery	Inputs	Resultado
MinPriorityQueue	minInsert	setupScenary1	key= 3 value= "C" key= 2 value= "B" key= 1 value= "A"	The test verifies that inserting elements with different properties into the minimum priority queue works correctly and that the queue returns the elements in the expected order.
MinPriorityQueue	minInsert	setupScenary2	key= 3 value= "C" key= 2 value= "B" key= 1 value= "A"	This last call to minInsert is expected to throw a RuntimeException, since the priority queue is full and no more elements can be inserted.
MinPriorityQueue	minInsert	setupScenary2	key= 1 value= "A" key= 2 value= "B" key= 3	This test verifies the behavior of the minInsert() operation on a MinPriorityQueue when trying to insert an element with a duplicate key into the structure.

			value= "C"	
MinPriorityQueue	extractMin	setupScenary2	empty	The test extracts the minimum value on the heap and ensures that the data structure remains a valid heap after extraction.
MinPriorityQueue	extractMin	setupScenary1	empty	Verifies that a RuntimeException is thrown when an attempt is made to fetch a minimum element from an empty priority queue.
MinPriorityQueue	extractMin	setupScenary2	empty	verifies that after inserting multiple elements into the minimum priority queue (minPriorityQueue), extracting the minimum returns the expected values.
MinPriorityQueue	getMin	setupScenary2	empty	The test is verifying that the getMin() method works correctly and returns the minimum element of the heap.
MinPriorityQueue	getMin	setupScenary1	empty	This method tests the case where it tries to obtain the maximum element of an empty priority queue, that is, it does not have any elements.
MinPriorityQueue	getMin extractMin	setupScenary2	empty	The testgetMax() should throw a RuntimeException if the priority queue is empty, so the first test case checks if this happens correctly.

### Configuration of MaxPriorityQueueTest scenarios

Name	Class	Scenery
setupScenary1	MaxPriorityQueueTest	
setupScenary2	MaxPriorityQueueTest	

### Test Cases Design

<b>Test objective:</b> Test the correct operation of the MaxPriorityQueue class.				
Class	Method	Scenery	Inputs	Resultado
MaxPriorityQueueTest	maxInsert	setupScenary1	key= 1 value= "A" key= 2 value= "B" key= 3 value= "C"	This test verifies the correct operation of the maxInsert method of a maximum priority queue
MaxPriorityQueueTest	maxInsert	setupScenary2	key= 1 value= "A" key= 2	The behavior of the top priority queue is tested when trying to add a new item when the queue is already full.

			value= "B"  key= 3  value= "C"	
MaxPriorityQueueTest	maxInsert	setupScenary2	key= 1  value= "A"  key= 2  value= "B"  key= 3  value= "C"	The test verifies that a RuntimeException is thrown when an attempt is made to insert a duplicate element with the same priority.
MaxPriorityQueueTest	extractMax	setupScenary2	empty	Verifies that the priority queue implementation can correctly handle the max-item fetch operation and can maintain the heap property of max after fetch.
MaxPriorityQueueTest	extractMax	setupScenary1	empty	This test checks that an exception is thrown when trying to fetch an element from an empty priority queue.
MaxPriorityQueueTest	extractMax	setupScenary2	empty	Suite of unit tests to verify the correct operation of the "Top Priority Queue" data structure

MaxPriorityQueueTest	getMax	setupScenary2	empty	This test verifies that the getMax() method correctly returns the element with the highest priority value from the maximum priority queue when the queue contains multiple elements.
MaxPriorityQueueTest	getMax	setupScenary1	empty	This tests the behavior of the priority queue when the getMax() function is called on an empty queue.
MaxPriorityQueueTest	getMax extractMax	setupScenary2	empty	This test verifies the correct operation of the getMax() and extractMax() methods of the MaxPriorityQueue class.