**HashTable Tests**

**Scenarios**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setupStage1 | HashTableTest | An empty hashTable with generics <Integer, String> |
| setupStage2 | HashTableTest | An empty hashTable with generics <Integer, Passenger> |

**Tests**

| **Test objective: Verify that the insert method of the hash table works correctly and doesn’t cause any execution errors or exceptions** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| HashTable | insert | setupStage1 | key = 3, value = “Santiago” | The key and value are added to the hashTable successfully |
| HashTable | insert | setupStage1 | 10 different keys and 10 different values | All the keys and values are added to the HashTable successfully |
| HashTable | insert | setupStage2 | key = 3, value =  Passenger(“F1”, “Santiago”, “no”, “no”, 10, 10) | The key and value are added to the hashTable successfully |

| **Test objective: Verify that the search method of the hash table works correctly and finds the correct keys** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| HashTable | search | setupStage1 | Add one node to the hashtable with key = 1 and value = “Alejandro”  search key = 1 | The node that the method returns has the same key and value as the one we inserted before |
| HashTable | search | setupStage1 | Add one node to the hashtable with key = 2, value = “B”  search key = 3 | The method returns null |
| HashTable | search | setupStage2 | Add one node to the hashtable with key = 3, value =  Passenger(“F1”, “Santiago”, “no”, “no”, 10, 10)  search key = 3 | The method finds a node with the corresponding key = 3 and the passenger value |

| **Test objective: Verify that the searchValue method of the hash table works correctly and finds the correct keys** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| HashTable | searchValue | setupStage1 | Add one node to the hashtable with key = 1 and value = “Alejandro”  searchValue key = 1 | The vale that the method returns is = “Alejandro” |
| HashTable | searchValue | setupStage1 | Add one node to the hashtable with key = 2, value = “B”  searchValue key = 3 | The method returns null |
| HashTable | searchValue | setupStage2 | Add one node to the hashtable with key = 3, value =  Passenger(“F1”, “Santiago”, “no”, “no”, 10, 10)  searchValue key = 3 | The method returns the corresponding passenger value |

| **Test objective: Verify that the delete method of the hash table works correctly by removing certains key from the hash table** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| HashTable | delete | setupStage1 | Add one node to the hashtable with key = 1 and value = “Alejandro”  delete key = 1 | The node is deleted from the hashtable |
| HashTable | delete | setupStage1 | Add three nodes to the hashtable  key = 1, value = “A”  key = 2, value = “B”  key = 3, value = “C”  delete key = 2 | The node with key 2 is deleted from the hashtable and the other two nodes remain |
| HashTable | delete | setupStage2 | Add one node to the hashtable with key = 2, value = “B”  delete key = 3 | The method doesn’t delete any nodes |

**Priority Queue Tests**

**Scenarios**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setupStage1 | PriorityQueueTest | An empty priority queue with generics <Integer, String> |
| setupStage2 | PriorityQueueTest | An empty priority queue with generics <InputPriority, Passenger> |

**Tests**

| **Test objective: Verify that the insert method of the priority queue works correctly and doesn’t cause any execution errors or exceptions** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| PriorityQueue | insert | setupStage1 | key = 3, value = “Alejandro” | The key and value are added to the priority queue successfully |
| PriorityQueue | insert | setupStage1 | 10 different keys and then different values | All the keys and values are added to the priority queue successfully |
| PriorityQueue | insert | setupStage2 | key = InputPriority("no","no",10F,"F5",10), value = “F5” | The input priority key and value are added to the priority queue successfully |

| **Test objective: Verify that the extract max method of the priority queue works correctly and returns the node with the highest priority** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| PriorityQueue | extractMax | setupStage1 | Add one node to the priority queue with key = 3, value = “Alejandro” | The extract max returns a node with key = 3 and value = “Alejandro” |
| PriorityQueue | extractMax | setupStage1 | Add 9 nodes with different keys and values  Add 1 node with a higher number than all the others | The extract max returns the node with they key with the highest number |
| PriorityQueue | extractMax | setupStage2 | Add 3 nodes with different input priority as keys and the seat of the input priority as value | The extract max returns the node with the highest input priority |

**Queue Tests**

**Scenarios**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setupStage1 | QueueTest | An empty queue with generics <Integer, String> |
| setupStage2 | QueueTest | An empty queue with generics <String, String> |

**Tests**

| **Test objective: Verify that the enqueue method of the queue works correctly and doesn’t cause any execution errors or exceptions** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Queue | enqueue | setupStage1 | key = 2, value = “Julian” | The key and value are added to the queue successfully |
| Queue | enqueue | setupStage1 | 10 different keys and then different values | All the keys and values are added to the queue successfully |
| Queue | enqueue | setupStage2 | key = “A”, value = “A” | The String key and value are added to the queue successfully |

| **Test objective: Verify that the dequeue method of the queue works correctly and returns the first person in the queue and removes it from the queue** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Queue | dequeue | setupStage1 | add a node with key = 2, value = “Julian” | The dequeue method returns a node with key = 2 and value = “Julian” , removes it from the queue and then the queue is empty |
| Queue | dequeue | setupStage1 | Add three nodes, the first one to add should have key = 2, value = “Julian” | The dequeue method returns the node with key = 2, and value = “Julian”, removes it from the queue and the queue remains not empty |
| Queue | dequeue | setupStage2 | Add a node with key = “A”, value = “Alejo” | The dequeue method returns the node with key = “A” and value = “Alejo”, removes it from the queue and then the queue is empty |

| **Test objective: Verify that the peek method of the queue works correctly and returns the first person in the queue** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Queue | peek | setupStage1 | add a node with key = 2, value = “Julian” | The peek method returns a node with key = 2 and value = “Julian” and the queue remains not empty |
| Queue | peek | setupStage1 | Add three nodes, the first one to add should have key = 2, value = “Julian” | The peek method returns the node with key = 2, and value = “Julian” and the queue remains not empty |
| Queue | peek | setupStage2 | Add a node with key = “A”, value = “Alejo” | The peek method returns the node with key = “A” and value = “Alejo” and then the queue is not empty |

**Queue Tests**

**Scenarios**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setupStage1 | StackTest | An empty stack with generics <Integer, String> |
| setupStage2 | StackTest | An empty stack with generics <String, String> |

**Tests**

| **Test objective: Verify that the enqueue push of the stack works correctly and doesn’t cause any execution errors or exceptions** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Stack | push | setupStage1 | key = 2, value = “Julian” | The key and value are added to the stack successfully |
| Stack | push | setupStage1 | 10 different keys and then different values | All the keys and values are added to the stack successfully |
| Stack | push | setupStage2 | key = “A”, value = “A” | The String key and value are added to the stack successfully |

| **Test objective: Verify that the dequeue method of the queue works correctly and returns the first person in the queue and removes it from the queue** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Stack | pop | setupStage1 | add a node with key = 2, value = “Julian” | The pop method returns a node with key = 2 and value = “Julian” , removes it from the stack and then the stack is empty |
| Stack | pop | setupStage1 | Add three nodes, the last one to add should have key = 2, value = “Julian” | The pop method returns the node with key = 2, and value = “Julian”, removes it from the stack and the stack remains not empty |
| Stack | pop | setupStage2 | Add a node with key = “A”, value = “Alejo” | The pop method returns the node with key = “A” and value = “Alejo”, removes it from the stack and then the stack is empty |

| **Test objective: Verify that the peek method of the queue works correctly and returns the first person in the queue** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Stack | top | setupStage1 | add a node with key = 2, value = “Julian” | The top method returns a node with key = 2 and value = “Julian” and the stack remains not empty |
| Stack | top | setupStage1 | Add three nodes, the last one to add should have key = 2, value = “Julian” | The top method returns the node with key = 2, and value = “Julian” and the stack remains not empty |
| Stack | top | setupStage2 | Add a node with key = “A”, value = “Alejo” | The top method returns the node with key = “A” and value = “Alejo” and then the stack is not empty |

**Controller Tests**

**Scenarios**

| **Name** | **Class** | **Scenario** |
| --- | --- | --- |
| setupStage1 | ControllerTest | Initialization of the controller  The controller should read the file src/data/file.txt  file.txt contains:  5  8  B4;Alejandro Londoño Bermúdez;yes;no;10  A1;Santiago Valencia García;yes;yes;10  C5;Julián Andrés Mendoza Castro;yes;no;10  The hashtable of the controller  The queue of the controller |

**Tests**

| **Test objective: Verify that the controller loads correctly the passengers from the file.txt file and adds them to the hashtable and the priority queues** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenario** | **Inputs** | **Awaited response** |
| Controller | hashtable.search | setupStage1 | key = “A1” | The method returns a Node with a passenger class, the passenger class contains  seat = “A1”  fullName = “Santiago Valencia García”  highClass = “yes”  vulnerable = “yes”  miles = 10 |
| Controller | hashtable.search | setupStage1 | Three searches  key = “B4”  key = “A1”  key = “C5” | The method returns a different node for each of the searches  For the search “B4”, it returns a node with a passenger with name = “Alejandro Londoño Bermudez”  For the search “A1”, it returns a node with a passenger with name = “Santiago Valencia Garcia”  For the search “C5”, it returns a node with a passenger with name = “Julian Andres Mendoza Castro” |
| Controller | inputQueue.extractMax() | setupStage1 | One extract max | The method should return one node with value = “C5” and a key = input priority with  seat = “C5”  highClass = “yes”  vulnerable = “no”  miles = 10 |
| Controller | inputQueue.extractMax() | setupStage1 | Three extract max | The method returns a different node for each extract max  The first extract max returns a node with value “C5”  The second extract max returns a node with value “B4”  The third extract max returns a node with value “A1” |
| Controller | outputQueue.extractMax() | setupStage1 | One extract max | The method should return one node with value = “A1” and a key OutputPriority with seat = “A1” |
| Controller | outputQueue.extractMax() | setupStage1 | Three extract max | The method returns a different node for each extract max  The first extract max returns a node with value “A1”  The second extract max returns a node with value “B4”  The third extract max returns a node with value “C5” |