| Name | Class | Scenarios (Examples) |
| --- | --- | --- |
| SetupScenaryConstructorCOntroller | ControllerTaskTest | empty |
| SetupScenaryConstructorHash | HashTableTest | empty |
| SetupScenaryConstructorQueue | QueueTest | empty |
| SetupScenaryConstructorStack | StackTest | empty |
| SetupScenaryAddTaskWithNullValues | ControllerTaskTest | Add task with null values |
| SetupScenaryAddTask | ControllerTaskTest | Add task |
| SetupScenaryRemoveTask | ControllerTaskTest | Remove task |
| SetupScenaryRemoveNonExistentTask | ControllerTaskTest | Remove Non-Existent Task |
| SetupScenaryModifyTest | ControllerTaskTest | Modify Task |
| SetupScenaryTaskWithNullValues | ControllerTaskTest | Modify Task with Null Values |
| SetupScenaryGetNonExistentTask | ControllerTaskTest | Get Non-Existent Task |
| SetupScenaryConstructorHash | HashTableTest | Initialize an empty hash table |
| SetupScenaryPutAndGet | HashTableTest | Put and get elements in hash table |
| SetupScenaryPutWithCollision | HashTableTest | Place items with collision in hash table |
| SetupScenaryRemove | HashTableTest | Delete an element from the hash table |
| SetupScenaryRemoveNonExistent | HashTableTest | Delete an element that does not exist in the hash table |
| SetupScenaryRemoveWithSameHash | HashTableTest | Delete elements with the same key in hash table |
| SetupScenaryContainsKey | HashTableTest | Check if a key exists in the hash table |
| SetupScenaryContainsKeyLimit | HashTableTest | Check the existence of keys in a full hash table |
| SetupScenaryCollissionHandling | HashTableTest | Place and handle collisions in hash table |
| SetupScenaryGetWithNonExistentKey | HashTableTest | Get an element with a key that does not exist in the hash table |
| SetupScenaryGetWithSameHash | HashTableTest | Get an element with the same key in the hash table |
| SetupScenaryInsertStandard | HeapTest | Insert elements into an empty heap |
| SetupScenaryInsertManyElements | HeapTest | Insert many elements into a heap |
| SetupScenaryInsertEdgeCase | HeapTest | Inserting elements into a heap in a borderline case |
| SetupScenaryInsertMinCase | HeapTest | Insert elements into a heap for minimum value |
| SetupScenatyHeapify | HeapTest | Perform a heapify operation on the mound |
| SetupScenaryHeapifyEdgeCase | HeapTest | Performing heapify on a mound in a borderline case |
| SetupScenaryHeapifyMinScenary | HeapTest | Heapify on a heap with the minimum value |
| SetupScenaryExtractMin | HeapTest | Extract minimum value from a heap |
| SetupScenaryExtractMinOrder | HeapTest | Extract elements in order from a heap |
| SetupScenaryExtractMinEmptyHeap | HeapTest | Trying to extract the minimum value from an empty heap |
| SetupScenaryEmptyScenary | HeapTest | Check if a heap is empty |
| SetupScenaryEmptyEdgeCase | HeapTest | Check if an empty mound in a borderline case |
| SetupScenaryEmptyAfterInsertion | HeapTest | Check if a heap is empty after inserting and removing elements |
| SetupScenaryEnqueue | PriorityQueueTest | Paste items into a priority queue |
| SetupScenaryEnqueueLimit | PriorityQueueTest | Queue a large number of items in a priority queue |
| SetupScenaryDequeue | PriorityQueueTest | Dequeue items from a priority queue |
| SetupScenaryDequeueLimit | PriorityQueueTest | Dequeue many items from a priority queue |
| SetupScenaryRemoveElement | PriorityQueueTest | Remove a specific item from a priority queue |
| SetupScenaryRemoveElementList | PriorityQueueTest | Remove a list of items from a priority queue |
| SetupScenaryIsEmpty | PriorityQueueTest | Check if a priority queue is empty |
| SetupScenaryIsEmptyLimit | PriorityQueueTest | Check for an empty priority queue in an edge case |
| SetupScenaryIsEmptyEnqueueDequeue | PriorityQueueTest | Check if a priority queue is empty after enqueuing and dequeuing items |
| SetupScenaryEnqueue | QueueTest | Glue elements into a glue |
| SetupScenaryEnqueueLimit | QueueTest | Glue a large number of elements in a glue |
| SetupScenaryEnqueueNull | QueueTest | Glue a null element into a glue |
| SetupScenaryDequeue | QueueTest | Dequeue items from a queue |
| SetupScenaryDequeueLimit | QueueTest | Dequeue items from a queue |
| SetupScenaryDequeueEmpty | QueueTest | Trying to dequeue items from an empty queue |
| SetupScenaryFront | QueueTest | Get the first element of a queue |
| SetupScenaryFrontLimit | QueueTest | Get the first element of a queue in an edge case |
| SetupScenaryFrintEmtyQueue | QueueTest | Try to get the first element of an empty queue |
| SetupScenaryIsEmpty | QueueTest | Check if a queue is empty |
| SetupScenaryIsEmptyLimit | QueueTest | Check for an empty queue in an edge case |
| SetupScenaryIsEmptyAfterDequeue | QueueTest | Check if a queue is empty after enqueuing and dequeuing elements |
| SetupScenarySize | QueueTest | Get the size of a queue |
| SetupScenarySize\_Limit | QueueTest | Get the size of a queue in an edge case |
| SetupScenaryAfterEnqueue | QueueTest | Get the size of a queue after enqueuing elements |
| SetupScenaryStress | QueueTest | Test a stressful situation in a queue |
| SetupScenaryConsistency | QueueTest | Verify the consistency of gluing and ungluing elements in a glue |
| SetupScenaryPush | StackTest | Push items into a stack |
| SetupScenaryPushLimit | StackTest | Push a large number of items into a stack |
| SetupScenaryPushEmpty | StackTest | Push an element onto an empty stack |
| SetupScenaryPop | StackTest | Dispatch items from a stack |
| SetupScenaryPopLimit | StackTest | Dispatch many items from a stack |
| SetupScenaryPopEmpty | StackTest | Trying to pop elements from an empty stack |
| SetupScenarySize | StackTest | Get the size of a stack |
| SetupScenarySizeLimit | StackTest | Get the size of a stack |
| SetupScenarySizeInteristing | StackTest | Get the minimum size of a stack |
| SetupScenaryFront | StackTest | Get the top element of a stack |
| SetupScenaryFrontEmpty | StackTest | Trying to get the top element of an empty stack |
| SetupScenaryFrontLimit | StackTest | Getting the top element of a stack in a borderline case |
| SetupScenaryStress | StackTest | Test a stress situation on a stack |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| ControllerTaskTest | setUp | SetupScenaryConstructorController | - | - |
| ControllerTaskTest | testAddTaskWithNullValues | SetupScenaryAddTaskWithNullValues | taskId = "T4",  title = null  description = "Description 4 "  deadline = null  priority = Priority.ALTA | "Error adding task: All fields must be provided and cannot be null." |
| ControllerTaskTest | testAddTask | SetupScenaryAddTask | taskId = "T1"  title = "Test task"  description = "Test description"  deadline = “2023-10-07”  priority = Priority.ALTA | "Task added successfully." |
| ControllerTaskTest | testRemoveTask | SetupScenaryRemoveTask | taskId = "T2"  title = "Test Task"  description = "Test Description"  deadline = “2023-10-07”, priority = Priority.MEDIA | "Task successfully deleted." |
| ControllerTaskTest | testRemoveNonExistentTask | SetupScenaryRemoveNonExistentTask | taskId = "T3" | "The task with ID T3 does not exist." |
| ControllerTaskTest | testModifyTask | SetupScenaryModifyTest | taskId = "T3"  title = "Test task"  description = "Test description"  deadline = “2023-10-07”, priority = Priority.BAJA  -  newTitle = "New title"  newDescription = "New description"  newDeadline = “2023-10-09”  newPriority = Priority.MEDIA | “Task modified successfully” |
| ControllerTaskTest | testModifyTaskWithNullValues | SetupScenaryTaskWithNullValues | taskId = "T3"  title = "Test Task" description = "Test Description"  deadline = “2023-10-07”  priority = Priority.BAJA | "No modifications were made." |
| ControllerTaskTest | testGetNonExistentTask | SetupScenaryGetNonExistentTask | taskId = "T4" | Expected result: null |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| HashTableTest | setUp | SetupScenaryConstructorHash | - | - |
| HashTableTest | testPutAndGet | SetupScenaryPutAndGet | Several objects of type Task with specific values. | Retrieved values ​​must be equal to Task objects. |
| HashTableTest | testPutWithMaxSize | SetupScenaryPutWithMaxSize | - | - |
| HashTableTest | testPutWithCollision | SetupScenaryPutWithCollision | Task objects are created with colliding keys and stored in taskTable. | Retrieved values ​​must be equal to Task objects. |
| HashTableTest | testRemove | SetupScenaryRemove | A Task object is stored and the remove method is called with its key. | The object must not be present in taskTable after deletion. |
| HashTableTest | testRemoveNonExistent | SetupScenaryRemoveNonExistent | The remove method is called with a key that does not exist in the taskTable. | The object must not be present in taskTable after deletion. |
| HashTableTest | testRemoveWithSameHashKey | SetupScenaryRemoveWithSameHash | Task objects are stored with colliding keys and the remove method is called with one of the keys. | The object must not be present in taskTable after deletion. |
| HashTableTest | testContainsKey | SetupScenaryContainsKey | A Task object is stored and a check is made to see if the key exists. | It should return true for the existing key and false for a non-existent key. |
| HashTableTest | testContainsKeyLimitScenario | SetupScenaryContainsKeyLimit | Task objects are stored and the keys are checked to see if they exist. | It should return true for existing keys and false for a non-existent key. |
| HashTableTest | testCollisionHandling | SetupScenaryCollissionHandling | Task objects are stored with colliding keys and retrieved from the taskTable. | Retrieved values ​​must be equal to Task objects. |
| HashTableTest | testGetWithNonExistentKey | SetupScenaryGetWithNonExistentKey | The get method is called with a key that does not exist in a hash table of type HashTable<String, Integer>. | It should return null. |
| HashTableTest | testGetWithSameHashKey | SetupScenaryGetWithSameHash | Task objects are stored with colliding keys and the get method is called with one of the keys. | It must return the corresponding Task object. |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| HeapTest | testInsertScenario | SetupScenaryInsertStandard | Numbers are inserted into the heap and the minimum is extracted. | The minimum value extracted must be equal to 3. |
| HeapTest | testInsertManyElements | SetupScenaryInsertManyElements | 1,000,000 elements are inserted into the heap. | - |
| HeapTest | testInsertEdgeCase | SetupScenaryInsertEdgeCase | A single number is inserted into the heap and the minimum is drawn. | The minimum value extracted must be equal to 5. |
| HeapTest | testInsertMinScenario | SetupScenaryInsertMinCase | Several numbers are inserted into the heap and the minimum is checked. | The minimum value extracted must be equal to 1. |
| HeapTest | testHeapifyScenario | SetupScenatyHeapify | Numbers are inserted into the heap, heapify is applied and the minimum is extracted. | The minimum value extracted must be equal to 2. |
| HeapTest | testHeapifyEdgeCase | SetupScenaryHeapifyEdgeCase | A single number is inserted into the heap, heapify is applied, and the minimum is extracted. | The minimum value extracted must be equal to 5. |
| HeapTest | testHeapifyMinScenario | SetupScenaryHeapifyMinScenary | Numbers are inserted into the heap, heapify is applied several times and the minimum is extracted. | The minimum values ​​extracted must be 1 and 3. |
| HeapTest | testExtractMin | SetupScenaryExtractMin | Numbers are inserted into the heap and the minimum is extracted. | The minimum value extracted must be equal to 1. |
| HeapTest | testExtractMinOrderSucetion | SetupScenaryExtractMinOrder | Numbers are inserted into the heap, extracted, and the order is checked. | Numbers are inserted into the heap, extracted, and the order is checked. |
| HeapTest | testExtractMinFromEmptyHeap | SetupScenaryExtractMinEmptyHeap | Numbers are inserted into the heap, extracted, and the order is checked. | You should throw a NoSuchElementException. |
| HeapTest | testIsEmptyScenario | SetupScenaryEmptyScenary | Checks whether the heap is empty after inserting and removing elements. | Must be true after extracting and false after inserting and before extracting. |
| HeapTest | testIsEmptyEdgeCase | SetupScenaryEmptyEdgeCase | An empty heap is checked to see if the heap is empty. | It must be true. |
| HeapTest | testIsEmptyAfterInsertion | SetupScenaryEmptyAfterInsertion | Checks whether the heap is empty after inserting elements. | Should be false after insert. |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| PriorityQueueTest | testEnqueue | SetupScenaryEnqueue | An item 1 is queued in the priority queue. It checks if the queue is empty. | The priority queue must not be empty after enqueuing an element. |
| PriorityQueueTest | testEnqueueLimit | SetupScenaryEnqueueLimit | 1,000 items are queued in the priority queue. It checks if the queue is empty. | The priority queue must not be empty after 1,000 items are queued. |
| PriorityQueueTest | testDequeue | SetupScenaryDequeue | An item 1 is queued into the priority queue and the item is checked out. | The extracted element must be equal to 1. |
| PriorityQueueTest | testDequeueLimit | SetupScenaryDequeueLimit | 1,000 items are queued into the priority queue and checked out in order. | The elements drawn must be 1, 2, 3,..., 1,000. The queue must be empty at the end. |
| PriorityQueueTest | testRemoveElement | SetupScenaryRemoveElement | It checks if an item 1 is present in the priority queue. | Element 1 must not be present in the queue, so the removeElement function must return false. |
| PriorityQueueTest | testRemoveElementLimit | SetupScenaryRemoveElementList | 1,000 items are queued into the priority queue and removed in order. | All queued elements must be removed correctly, and the queue must be empty at the end. |
| PriorityQueueTest | testIsEmpty | SetupScenaryIsEmpty | Checks whether the priority queue is empty after enqueuing and dequeuing elements. | Must be true after dequeue and false after enqueue and before dequeue. |
| PriorityQueueTest | testIsEmptyLimit | SetupScenaryIsEmptyLimit | 1,000 items are enqueued into the priority queue and dequeued in order. | The priority queue must be empty at the end. |
| PriorityQueueTest | testIsEmptyAfterEnqueueDequeue | SetupScenaryIsEmptyEnqueueDequeue | Checks whether the priority queue is empty after enqueuing and dequeuing elements. | Must be true after dequeue and false after enqueue and before dequeue. |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| QueueTest | testEnqueue | SetupScenaryEnqueue | Elements 1, 2 and 3 are queued into the queue. The size and element in front are verified. | The size of the queue must be 3 and the element in front must be 1. |
| QueueTest | testEnqueueLimit | SetupScenaryEnqueueLimit | 1,000 items are queued in the queue. The size and element in front are verified. | The size of the queue must be 1,000 and the element in front must be 1. |
| QueueTest | testEnqueueNull | SetupScenaryEnqueueNull | An attempt is made to paste null into the queue. It checks if the queue is empty. | The glue must be empty since it is not allowed to glue null. |
| QueueTest | testDequeue | SetupScenaryDequeue | Items 1, 2, and 3 are enqueued into the queue, and dequeued. The dequeued element and size are verified. | The dequeued element must be 1 and the queue size must be 2. |
| QueueTest | testDequeueLimit | SetupScenaryDequeueLimit | 1,000 items are enqueued into the queue and dequeued in order. It checks that the queue is empty at the end. | All enqueued items must be dequeued successfully and the queue must be empty at the end. |
| QueueTest | testDequeueEmptyQueue | SetupScenaryDequeueEmpty | An attempt is made to dequeue from an empty queue. It checks to see if the EmptyQueueException is thrown. | You should throw EmptyQueueException. |
| QueueTest | testFront | SetupScenaryFront | Elements 5, 10, and 15 are queued into the queue, and the element in front is checked. | The element in front must be 5. |
| QueueTest | testFrontLimit | SetupScenaryFrontLimit | 1,000 elements are queued, and the front element is checked. | The element in front must be 1. |
| QueueTest | testFrontEmptyQueue | SetupScenaryFrintEmtyQueue | An attempt is made to access the element at the front of an empty queue. It checks to see if the EmptyQueueException is thrown. | You should throw EmptyQueueException. |
| QueueTest | testIsEmpty | SetupScenaryIsEmpty | It checks if the queue is initially empty. | It must be true at the beginning. |
| QueueTest | testIsEmpty\_limit | SetupScenaryIsEmptyLimit | An element is queued and checked to see if the queue is empty. | Should be false after enqueuing. |
| QueueTest | testIsEmptyAfterDequeue | SetupScenaryIsEmptyAfterDequeue | An element is enqueued, dequeued, and checked to see if the queue is empty. | Should be true after enqueue and dequeue. |
| QueueTest | testSize | SetupScenarySize | The size of an empty queue is checked. | It must be 0 at the beginning. |
| QueueTest | testSize\_limit | SetupScenarySize\_Limit | An element is queued and the queue size is checked. | It should be 1 after gluing. |
| QueueTest | testSizeAfterEnqueue | SetupScenaryAfterEnqueue | Two elements are queued and the size of the queue is checked. | It should be 2 after gluing. |
| QueueTest | testStresss | SetupScenaryStress | 1,000,000 elements are enqueued and dequeued. The queue size and dequeued elements are checked. | The size of the queue must be the given number of iterations and the dequeued elements must range from 0 to the number of iterations. |
| QueueTest | testConsistency | SetupScenaryConsistency | 1,000 items are queued and dequeued in a consistent pattern. It checks that the queue is empty at the end. | All enqueued items must be dequeued in order and the queue must be empty at the end. |

| **Test Objective:** | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input Values** | **Result** |
| StackTest | testPush | SetupScenaryPush | Elements 1, 2 and 3 are stacked on the stack. The size and item are verified at the top. | The stack size must be 3 and the element on top must be 3. |
| StackTest | testPushLimit | SetupScenaryPushLimit | 1,000 items are stacked on the stack. The size and item are verified at the top. | The stack size must be 1,000 and the item on top must be 1,000. |
| StackTest | testPushEmptyStack | SetupScenaryPushEmpty | It checks if the stack is empty initially, stacks an element, and checks the size and the element on top. | The stack must not be empty after stacking an element, and the element on top must be 42. |
| StackTest | testPop | SetupScenaryPop | If elements 1, 2 and 3 are added to the pile, it will be removed. Check the stripped element and its size. | The unstacked element must be 3 and the stack size must be 2. |
| StackTest | testPopLimit | SetupScenaryPopLimit | 1,000 elements are added to the pile and removed in order. It turns out that the dick is empty at the end. | All pilated elements must be removed correctly and the pila must be empty at the end. |
| StackTest | testPopEmptyStack | SetupScenaryPopEmpty | An attempt is made to destack from an empty stack. Checks if the EmptyStackException exception is thrown. | You should throw EmptyStackException. |
| StackTest | testSize | SetupScenarySize | The size of an empty stack is checked. | It must be 0 at the beginning. |
| StackTest | testSize\_limit | SetupScenarySizeLimit | An element is stacked and the stack size is checked. | It should be 1 after stacking. |
| StackTest | testSize\_interesting | SetupScenarySizeInteristing | Two elements are stacked and the size of the stack is checked. | It should be 2 after stacking and 1 after unstacking one. |
| StackTest | testFront | SetupScenaryFront | Elements 5, 10, and 15 are stacked on the stack, and the element on top is checked. | The element at the top must be 15. |
| StackTest | testFrontEmptyStack | SetupScenaryFrontEmpty | An attempt is made to access the element at the top of an empty stack. It checks if the EmptyStackException is thrown. | You should throw EmptyStackException. |
| StackTest | testFrontLimit | SetupScenaryFrontLimit | 1,000 elements are stacked on the stack, and the element on top is checked. | The item at the top should be 1,000. |
| StackTest | testStress | SetupScenaryStress | 1,000,000 elements can be accessed and removed. Check the size of the dick and the removed elements. | The size of the queue must be the given number of iterations and the dequeued elements must range from 0 to the number of iterations. |