**TEST DESIGN**

**Configuration of the Scenarios**

|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenario** |
| **setUpStage1** | **HashTable** | **A HashTable containing nodes with:**  **Key: "key1", Value: 27**  **Key: "key2", Value: 53** |
| **setUpStage2** | **Stack** | **A Stack containing key-value pairs:**  **Key: "key1", Value: 12**  **Key: "key2", Value: 56** |
| **setUpStage3** | **MaxHeap** | **MaxHeap containing TaskReminders:**    **Insert TaskReminder with level importance: 30**  **Insert TaskReminder with level importance: 83** |
| **setUpStage4** | **Queue** | **A Queue containing key-value pairs:**    **Enqueue –> key: "key1", Value: 18**  **Enqueue – >key: "key2", Value: 97** |
| **setUpStage5** | **HashTable** | **Empty HashTable** |
| **setUpStage6** | **Stack** | **Empty Stack** |
| **setUpStage7** | **MaxHeap** | **Empty MaxHeap** |
| **setUpStage8** | **Queue** | **Empty Queue** |
| **setUpStage9** | **MaxHeap** | **Attempt to insert 1001 TaskReminders** |
| **setUpStage10** | **HashTable** | **A HashTable containing nodes with:**  **Key: "key1", Value: 27**  **Key: "key1", Value: 53** |
| **setUpStage11** | **MaxHeap** | **MaxHeap containing TaskReminders:**    **Insert TaskReminder with level importance: 30**  **Insert TaskReminder with level importance: 10** |

***TEST CASES OF HASHTABLE***

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify whether a task has been accurately included in the hashtable.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Hashtable** | **testInsert and search** | **SetupStage1** | **- The size must change the size in +1**  **- The item has been successfully inserted into the hash table.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Check if a task has been properly removed from the hashtable.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Hashtable** | **testDelete** | **SetupStage1** | **After inserting "key1" and "key2" and deleting "key1," "key1" should not be found, and "key2" should still be present with its associated value.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Check if a task has the same key as another, throw an exception, and then insert the task.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Hashtable** | **testInsertDuplicateKey** | **SetupStage10** | **This test expects a DuplicatedObjectException to be thrown when inserting duplicate keys.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: verify the behavior of the isEmpty method in the hashTable class in two scenarios: when it is empty and when it contains at least one element.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Hashtable** | **testIsEmpty** | **A task with:**  **{Key: "key1"**  **Value: 27 }**  **setUpStage5** | **Initially, the hash table should be empty, and after inserting "key1," it should no longer be empty.** |

***TEST CASES OF STACK***

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify the correctness of push and pop operations in the stack data structure.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Stack** | **testPushAndPop** | **SetupStage6**  **SetupStage2** | **Push "key1" and "key2" onto the stack. The should no longer be empty, and the size should increase accordingly. After pushing "key1" and "key2," popping should return the values in the reverse order in which they were pushed.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Ensure the peek method correctly retrieves the top element of the stack without altering it.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Stack** | **testPeek** | **SetupStage2** | **Initially, when the stack is empty, calling peek should return null. After pushing "key1" with a value of 12 and "key2" with a value of 56, peek should return a non-null value. The value returned by peek should match the last pushed value, which is 56.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Validate the behavior of the isEmpty method to check if the stack is empty.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Stack** | **testIsEmpty** | **SetupStage6**  **A task with:**  **{Key: "key1"**  **Value: 12}** | **Initially, the stack should be empty, so calling isEmpty should return true. After pushing "key1" with a value of 12, calling isEmpty should return false.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify that the size method accurately reports the stack's size after push and pop operations.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Stack** | **testSize** | **SetupStage6**  **setUpStage2** | **Initially, the stack should have a size of 0. After pushing "key1" with a value of 12, the stack's size should become 1. When "key2" with a value of 56 is pushed, the size should increase to 2. After popping an element from the stack, the size should decrease to 1.** |

***TEST CASES OF QUEUE***

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify the correctness of enqueue and dequeue operations in the queue data structure.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Queue** | **testEnqueueAndDequeue** | **SetupStage8**  **SetupStage4** | **Initially, the queue should be empty with a size of 0. After enqueuing "key1" with a value of 18, the queue should not be empty, and its size should be 1. Subsequently enqueuing "key2" with a value of 97 should increase the queue size to 2. When dequeuing from the queue, the dequeued value should match the first enqueued value, which is 18, and the queue's size should decrease to 1. After dequeuing the second value, it should be 97, and the queue should become empty.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Ensure the peek method correctly retrieves the front element of the queue without altering it.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Queue** | **testPeek** | **SetupStage4** | **Initially, when the queue is empty, calling peek should return null. After enqueuing "key1" with a value of 18 and "key2" with a value of 97, calling peek should return a non-null node. The value in the peeked node should match the value of the first enqueued element, which is 18.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Validate the delete method's behavior to remove a specified node from the queue.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Queue** | **testDelete** | **SetupStage4** | **After enqueuing "key1" with a value of 18 and "key2" with a value of 97, the node to be deleted should be the first enqueued node, and it should not be null. After deleting the specified node, the new front node should be the one enqueued later, which has a value of 97.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Validate the behavior of the isEmpty method to check if the queue is empty.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Queue** | **testIsEmpty** | **SetupStage8**  **A task with:**  **{Key: "key1"**  **Value: 18}** | **Initially, the queue should be empty, so calling isEmpty should return true. After enqueuing "key1" with a value of 18, calling isEmpty should return false.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify that the size method accurately reports the queue's size after enqueue and dequeue operations.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **Queue** | **testSize** | **SetupStage8**  **setUpStage4** | **Initially, the queue should have a size of 0. After enqueuing "key1" with a value of 18, the queue's size should become 1. When "key2" with a value of 97 is enqueued, the size should increase to 2. After dequeuing an element from the queue, the size should decrease to 1.** |

***TEST CASES OF MAXHEAP***

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify the correctness of inserting tasks into a max-heap and extracting the task with the highest importance level.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **MaxHeap** | **testInsertAndExtractMaximum** | **SetupStage7**  **SetupStage3** | **Initially, the max-heap should be empty with a heap size of 0. After inserting task1 with an importance level of 30, the max-heap should not be empty, and its heap size should be 1. Subsequently inserting task2 with an importance level of 83 should increase the heap size to 2. When extracting the maximum task from the max-heap, the task's importance level should be 83, and the heap size should decrease to 1. Calling the maximum method should return the task with an importance level of 30.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Ensure that the increaseKey method correctly updates a task's importance level in the max-heap.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **MaxHeap** | **testIncreaseKey** | **SetupStage11** | **After inserting task1 with an importance level of 30, using the increaseKey method to update its importance level to 10 should not change the maximum task's importance level. The maximum task's importance level should remain 30.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Validate the behavior of the isEmpty method to check if the max-heap is empty.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **MaxHeap** | **testIsEmpty** | **SetupStage7**  **TaskReminder with level importance: 30** | **Initially, the max-heap should be empty, so calling isEmpty should return true. After inserting task1 with an importance level of 30, calling isEmpty should return false.** |

|  |  |  |  |
| --- | --- | --- | --- |
| **Objective of the Test: Verify that the getHeapSize method accurately reports the size of the max-heap after insertions.** | | | |
| **Class** | **Mehod** | **Scenario** | **Expected result** |
| **MaxHeap** | **testGetHeapSize** | **SetupStage7**  **TaskReminder with level importance: 30** | **Initially, the max-heap should have a heap size of 0. After inserting task1 with an importance level of 30, the heap size should become 1.** |