## hash table

| **Name** | **Class** | **Scenery** |
| --- | --- | --- |
| setup1 | HashTableTest | Hash table is empty |
| setup2 | HashTableTest | The hash table has added values. |
| setup3 | HashTableTest | The hash table has added values and some are collided |

| Test Objective: test if the hastable is capable of detecting when adding is not possible due to a repeated key | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | add | setUp2 | add(0.4, "Alejadro Magno")  add(0.4, "MonitorGod") | "Exception object with the same key that other" |

| Test Objective: test if the hastable is capable of adding an element if the list is empty | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | add | setUp1 | 11.0, "Hola" | the element is added to the hash table |

| Test Objective: test if the hash table is capable of detecting when searching is not possible due to being empty | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | search | setUp1 | search(0,3) | Exception list is void |

| Test Objective: test if the hash table is adding and removing effectively | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | remove | setUp1 | 0.3, 0.4, 2, 2.4, 4.5, 6.5, 8.5. | hash table should be empty |

| Test Objective: test if search method is working properly | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | search | setUp2 | search(8.5) | se encuentra el valor buscado con la llave 0.4, en este caso “Juan” |

| Test Objective: test if search method is working properly when there is a collision | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | add  search | setUp2 | n = 0.39, "Marco Aurelio"  a = 0.37, "Alejandro Magno";  search(0.37) | 0.39, "Alejandro Magno" should be found |

| Test Objective: test if search throws exception when the item doesn’t existe | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | search | setUp2 | search(0.26) | Exception object doesn't exist |

| Test Objective: test if remove method throws exception when the item doesn’t exist | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | remove | setUp2 | remove(0.26) | Exception object doesn’t exist |

| Test Objective: test if remove method works properly | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | remove | setUp2 | remove(0.3) | the element should not exist in the hash table |

| Test Objective: test if clone method makes identical items | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | clone() | setUp2 |  | the elements should be the same in both hash tables |

| Test Objective: Check if the elements are real clones and not pointer to the original items | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | clone() | setUp2 |  | the elements should be the same in both hash tables and changing something in one of the elements doesn’t change the clone of the other hash table |

| Test Objective: test the search method for an element that is collided | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | search | setUp3 | 14.12 | the item should be found and equal to “Aurelio” |

| Test Objective: test if remove works well when it is eliminating a collisioned element | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | remove | setUp3 | 14.12 | the element with key 14.12 should be removed |

| Test Objective: test if add method works well when it would make a collision | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| HashTableTest | add | setUp3 | 0.36, “Marco” | the element with key 14.12 should be removed |

# 

# Heap

| **Name** | **Class** | **Scenery** |
| --- | --- | --- |
| setup1 | HeapTest | An empty Heap. |
| setup2 | HeapTest | The Heap has some elements in it. |
| setup3 | HeapTest | The heap has some negative values in it |

| Test Objective: correctly inserting an element in an empty Heap | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | Insert | setUp1 | a = 1, 10 | Using the method estactMax() we can confirm if the item is at the top of the list, meaning it was inserted. |

| Test Objective: Confirm the extractMax() method is working properly extracting some elements | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | extractMax() | setUp2 |  | Extracting three elements should return: 10, 20, 30 |

| Test Objective: Check if changing the priority of one elements would make it to be on top of the Heap | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | increase key()  extractMax() | setUp2 | IncreaseKey(10, 4) | changing the priority of one element would affect the extractMax()’s result. |

| Test Objective: Check if changing the priority of one elements would make it to be on top of the Heap | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | increase key()  extractMax() | setUp2 | IncreaseKey(10, 4) | changing the priority of one element would affect the extractMax()’s result, in this case it should be equal to 10. |

| Test Objective: Check if extractMax method returns null when the Heap is empty() | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | extractMax() | setUp1 |  | null |

| Test Objective: Check if extractMax works even when the object is null | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | extractMax() | setUp1 |  | null |

ir a linea 79

| Test Objective: Check if extractMax works properly when the priority is negative | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | ExtractMax() | setUp1 |  | null |

ir a linea 86

| Test Objective: Check if extractMax works properly when the priority is negative | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | ExtractMax() | setUp1 |  | null |

| Test Objective: Inserting and extracting elements with high priority | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | insert | setUp1 | Integer.MaxValue, 10  Integer.Min, 20  Integer.MaxValue, 30 | when extracting elements, it should be in order |

| Test Objective: Inserting and extracting | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | insert | setUp1 | 5, 10 | 10 |

| Test Objective: Inserting and extracting them in order | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | insert | setUp1 | 3, 10  2, 20  1, 30  4, 40 | Extracting them should return  40  10  20  30 |

| Test Objective: Inserting elements with the same priority and extracting them | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | insert | setUp1 | 1, 10  1, 20  1, 30  1, 40  1, 50 | Extracting them should give all of these elements.  10  50  40  30  20 |

| Test Objective: Test if cloning makes an identical Heap | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| Heap | insert | setUp1 | 1, 10  2, 20  3, 30 | Each of the elements should be the same in both Heaps. |

## Queue

| **Name** | **Class** | **Scenery** |
| --- | --- | --- |
| setup1 | QueueTest | Queue is empty |
| setup2 | QueueTest | The queue has some String values |

| Test Objective: Test if adding in an empty queue works | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | offer | setUp1 | “Hola” | The element should be added to the queue |

| Test Objective: Test if adding in an non empty queue works | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | offer | setUp2 | “Hola” | The element should be added to the queue |

| Test Objective: Test if removing an element in an empty queue throws exception | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | poll | setUp1 |  | it should throw an exception of type : exceptionThisDataStructureIsVoid |

| Test Objective: Test if removing an element in an non empty queue works | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | poll | setUp2 |  | it should return “Hola” |

| Test Objective: Test if looking at the element in the front in an empty queue throws an exception | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | front | setUp1 |  | it should throw an exception of type : exceptionThisDataStructureIsVoid |

| Test Objective: Test if looking at the element in the front in an non empty queue throws an exception | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | front | setUp2 |  | it should return “Hola” |

| Test Objective: Test if eliminating every element in the queue makes it empty | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | empty() | setUp2 | 4\*poll() | queue.isEmpty() = true |

| Test Objective: Test if the size() method return the quantity of elements that are in the queue | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | size() | setUp2 |  | queue.size() == 4 |

| Test Objective: Test if the clone() methods clones properly each of the elements of the queue | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | size() | setUp2 |  | queue.size() == newQueue.size()  for every element  queue.poll() == newQueue.poll()  or equals if it is an object |

| Test Objective: Check if cloning doesn’t make a reference to the previous element, but a new object that is identical. Changes an element of the first queue and the same element in the cloned queue shouldn’t be identical. | | | | |
| --- | --- | --- | --- | --- |
| **Class** | **Method** | **Scenery** | **Input value** | **Return** |
| QueueTest | size() | setUp2 |  | The items shouldn’t be identical |