|  |  |  |
| --- | --- | --- |
| **Name** | **Class** | **Scenery** |
| setUpStage1 | ReminderTest | Initializes controller class and hash, priority queue, queue,  stack structures. |
| setUpStage2 | ReminderTest | Initialize hash, create reminder and add it to hash table. |
| setUpStage3 | ReminderTest | Initializes queue, creates three reminder objects, adds them to the queue. |
| setUpStage4 | ReminderTest | Initialize queue and priority queue creates two reminder objects, one with priority and one without priority, and adds the objects to their respective structure. |
| setUpStage6 | ReminderTest | Initialize priority queue creates two reminder objects, both with priority, and saves them in the respective structure. |
| setUpStage7 | ReminderTest | Initialize stack, create two reminder objects, create Action.type variables and create Action objects, then add them to the stack. |
| setUpStage8 | ReminderTest | Initialize the controller class with its methods, hash class and register two reminders with the registerRemiders method of controller. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1HashInsert()** | | | | |
| **Objective of test: Add a reminder and search for it to verify it was saved.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashInsert() | setUpStage1 | Integer key, Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2HashInsert()** | | | | |
| **Objective of test: Save two reminders with different keys in the hash table and lookup them to verify that they are not the same.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashInsert() | setUpStage1 | Reminder not null  Reminder not null | Reminder not null  Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1HashSearch()** | | | | |
| **Objective of test: A reminder is searched with the save key hoping that what is found is not null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashSearch() | setUpStage2 | Integer key | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2HashSearch()** | | | | |
| **Objective of test: An existing reminder is searched in the hash table but with a different key than the reminder itself, with this it is expected that the search will be null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashSearch() | setUpStage2 | Integer key | Reminder Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1HashDelete()** | | | | |
| **Objective of test: Delete an existing reminder in the hashtable with a reminder key and compare whether the number of objects was reduced from 1 to 0.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashDelete() | setUpStage2 | Integer key | size 0 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2HashDelete()** | | | | |
| **Objective of test: Reminder is deleted with its respective key, and it is verified that the reminder has been deleted by searching for it and verifying that it is null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashDelete() | setUpStage2 | Integer key | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3HashDelete()** | | | | |
| **Objective of test: Try to delete a reminder with a key that is not its, so the reminder is not deleted.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| ChainigHashTable | hashDelete() | setUpStage2 | Integer key | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1QueueEnQueue()** | | | | |
| **Objective of test:  A reminder object is created, saved in the queue and then checked to see if the list is empty.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | enQueue() | setUpStage1 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2QueueEnQueue()** | | | | |
| **Objective of test: A reminder object is created, added to the queue, then the object is deleted, and then check that the first element of the structure is empty** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | enQueue() | setUpStage1 | Reminder not null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3QueueEnQueue()** | | | | |
| **Objective of test: A reminder object is created, added to the queue and then checked to see if the first element of the structure is the same as the one added.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | enQueue() | setUpStage1 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1QueueDeQueue()** | | | | |
| **Objective of test: A reminder object is created, added to the queue, the first object is removed from the structure and the structure is checked to see if it is empty.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | deQueue() | setUpStage1 | Reminder not null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2QueueDeQueue()** | | | | |
| **Objective of test: The first object is removed from the structure and it is checked to see if the structure is empty, knowing that there are more elements added.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | deQueue() | setUpStage1 | none | False |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3QueueDeQueue()** | | | | |
| **Objective of test: The result of removing the first object from the structure is saved in a variable and compared with the new first object.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | deQueue() | setUpStage3 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1QueueFront()** | | | | |
| **Objective of test: The first object of the structure is requested and it is verified that it is not null knowing that there are objects in the structure** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | front() | setUpStage3 | Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2QueueFront()** | | | | |
| **Objective of test: The first object of the structure is requested and it is verified that it is null knowing that there are no objects in the structure** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | front() | setUpStage1 | Reminder null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3QueueFront()** | | | | |
| **Objective of test: A reminder object is created with the assignment of the first element of the structure, and it is compared with the new first element** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Queue | front() | setUpStage3 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1StackPush()** | | | | |
| **Objective of test: A reminder object and an action Type object are created to create an action object, then add it to the stack and finally check if the structure is empty** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | push() | setUpStage1 | Reminder not null | Action action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2StackPush()** | | | | |
| **Objective of test: A reminder object is created, an action Type object to create an action object, and added to the stack, then the first element is taken out of the structure and it is checked if the object is the same** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | push() | setUpStage7 | Reminder not null | Action action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3StackPush()** | | | | |
| **Objective of test: An action type object that is null is created and added to the stack to then check if there are objects in the structure** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | push() | setUpStage1 | Action null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1StackPop()** | | | | |
| **Objective of test: A null action object is created and added to the stack to then check if the first object in the structure is null, knowing that other objects had already been added.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | pop() | setUpStage7 | Action null | Action action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2StackPop()** | | | | |
| **Objective of test: An action type object that is null is created and added to the stack to then check if there is a first element in the structure** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | pop() | setUpStage1 | Action null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3StackPop()** | | | | |
| **Objective of test: Two action type objects are created, then one is assigned the first element of the list and the other is assigned the output of the first element, and then compare if they are the same** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | pop() | setUpStage7 | 2 Action null | Action action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1StackTop()** | | | | |
| **Objective of test: An action type object that is null is created and added to the stack to then check if there is a first element in the structure** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | top() | setUpStage1 | Action null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2StackTop()** | | | | |
| **Objective of test: An action type object that is null is created and added to the stack to then check if there is a first element in the structure, knowing which element was added previously.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | top() | setUpStage7 | Action null | Action action |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3StackTop()** | | | | |
| **Objective of test: A null action type object is created to assign the first element of the structure, then another action type object is created with parameters other than null and it is verified when the new object is added, these are not the same.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Stack | top() | setUpStage7 | Action action  Action action | Action action |

Revisar

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1PriorityQueueMaximun()** | | | | |
| **Objective of test:  An object of type priority queue is created and added to the priority queue structure, then it is checked if the first object in the list is different from null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage5 | Reminder not null  Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2PriorityQueueMaximun()** | | | | |
| **Objective of test: A null reminder is added to the priority queue class and the first element of the structure is searched** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage1 | Reminder null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3PriorityQueueMaximun()** | | | | |
| **Objective of test: Queue and priority queue objects are created and added to their structures, then it is verified that the elements are not the same** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage4 | 2 Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1PriorityQueueExtract()** | | | | |
| **Objective of test: A variable type object is created and the first object of the priority queue structure is assigned to it, knowing that no more objects have been added** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | extractMax() | setUpStage1 | Reminder  null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2PriorityQueueExtract()** | | | | |
| **Objective of test:  A variable type object is created and the first object of the priority queue structure is assigned to it, knowing that more objects have been added** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | extractMax() | setUpStage6 | Reminder not null | Reminder reminder |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1PriorityQueueInsert()** | | | | |
| **Objective of test: A reminder is created and added to the priority queue to then check if the number of objects in the structure has increased** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage1 | Reminder not null | 1 |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2PriorityQueueInsert()** | | | | |
| **Objective of test: A null reminder is created and added to the priority queue, then the first element of the structure is assigned to the null reminder knowing that no elements have been added** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage1 | Reminder null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3PriorityQueueInsert()** | | | | |
| **Objective of test: A reminder type is created with a priority of 5 and added to the structure knowing that more reminders have been added with lower priorities and it is checked if the object that is with the highest priority is the same object that was added previously** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | insert() | setUpStage6 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1PriorityQueueIncreaseKey()** | | | | |
| **Objective of test: Two null reminders are assigned to the element with the highest priority in the priority queue with the difference that the priority of the second one was increased before being assigned. Finally, the elements are compared, checking that they are not equal.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | increaseKey() | setUpStage6 | 2 Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2PriorityQueueIncreaseKey()** | | | | |
| **Objective of test: A reminder object is created with a priority of 1 and added to the priority queue, to then verify that the only object found in the table has the same priority** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | increaseKey() | setUpStage1 | Reminder not null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3PriorityQueueIncreaseKey()** | | | | |
| **Objective of test: A null reminder object is created to which the first element of the priority queue is assigned, then the priority is increased to the maximum to another element with a lower priority and it is verified that the maximum is no longer the same as it was before.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| PriorityQueue | increaseKey() | setUpStage6 | 2 Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1RegisterReminder()** | | | | |
| **Objective of test: Two null reminder elements are created, then a reminder is registered in the method, the object sought in the hash and the priority queue are assigned to the null reminders respectively and it is verified that they are equal.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | registerReminders() | setUpStage1 | 2 Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2RegisterReminder()** | | | | |
| **Objective of test: A priority reminder is registered in the controller method, a null reminder is created to which the first queue element is assigned, and the queue object is verified to be null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | registerReminders() | setUpStage1 | Reminder null | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3RegisterReminder()** | | | | |
| **Objective of test: A reminder is registered in the controller method and then the first element of the queue and priority queue structures is searched, hoping to verify that they are not equal.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | registerReminders() | setUpStage1 | none |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1DeleteReminder()** | | | | |
| **Objective of test: A reminder is registered in the controller method, then an attempt is made to delete it with the controller delete method, and then search for the object and verify if it exists.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | deleteReminders() | setUpStage1 | Reminder null  Integer key | Null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2DeleteReminder()** | | | | |
| **Objective of test: A priority reminder object is eliminated with a key and the object with the highest priority in the structure is searched, knowing that more objects have been added and it is verified if the first one in the structure is not null.** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | deleteReminders() | setUpStage8 | Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3DeleteReminder()** | | | | |
| **Objective of test: An object with priority is deleted with a key and assigned to a null object resider the first of the queue structure, it is expected to verify that the object that was deleted in priority queue has not been deleted in queue** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | deleteReminders() | setUpStage8 | Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test1ModifyReminder()** | | | | |
| **Objective of test: Two null reminder objects are created to which an existing object in the hash table is assigned with the difference that the second object has had data changed and it is verified that the two objects have the same designation** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | modifyReminders() | setUpStage8 | 2 Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test2ModifyReminder()** | | | | |
| **Objective of test: A null reminder object is created and assigned an object searched in the hash table, whose priority has been changed to 5, it is verified if it is correct that the priority of the object has changed** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | modifyReminders() | setUpStage8 | Reminder null | Reminder not null |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Name of test: test3ModifyReminder()** | | | | |
| **Objective of test: An object is deleted from the hash table with its respective key, then an attempt is made to change the data of the deleted object, finally it is verified if the object was actually deleted** | | | | |
| **Class** | **Method** | **Scenery** | **Input** | **Output** |
| Controller | modifyReminders() | setUpStage8 | Reminder null | Null |