**Formato de escenarios y casos de uso**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setup1 | ControllerTest | Mades a controller id for the methods to use |
| ConfirmAssistancePassengerNotFoundTest | ControllerTest | Uses the setup and verifies if an inexistent passenger id can be confirmed, and it should not be found |
| GenerateEntryListNotConfirmedPassengersTest | ControllerTest | Uses the setup and verifies if the passengers have confirmed creating an entry list, and it should not find any confirmed passengers |
| AddToExistListNotConfirmedPassengersTest | ControllerTest | Uses the setup and verifies if any passenger confirmed can be added in a list, and it should not be found any confirmed passenger |
| ImportDataFromCSVLoadedTest | ControllerTest | Uses the setup and verifies if the Import can be done, and later it tries to import again and it should say that is already loaded |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| PassengerCreatedSuccesfullyTest | PassengerTest | Creates a new Passenger and confirms it his information of the registration equals with the created one |
| PassengerCreatedSuccessBTest | PassengerTest | Creates a new Passenger and confirms it his information of the registration equals with the created one, having on count that the numeric values are alright with the invalid number exception |
| ConfirmAssistanceTest | PassengerTest | Creates a new Passenger and asserts if the passenger is confirmed, it should not, and later confirms the passenger and verify it. |
| CompareToPassengerPriorityTest | PassengerTest | Creates three Passengers and compares between their priority, being lower or higher among them |

…

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | StackTest | Makes two stacks |
| isEmptyTest1 | StackTest | Asserts that the stack is empty |
| isEmptyTest2 | StackTest | Asserts that the stack is not empty because an element was added |
| isEmptyTest3 | StackTest | Asserts that the stack is empty adding an element and poping it later |
| isTopTest1 | StackTest | Tries to obtain the element on an empty stack |
| isTopTest2 | StackTest | Adds two elements and the top one should be the second added |
| isPopTest1 | StackTest | Tries to pop an empty stack |
| isPopTest2 | StackTest | Adds an element and later pops it asserting that the stack is empty now |
| isPopTest3 | StackTest | Adds two elements and pops one, so the second one should be popped |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| inserElementTest | HeapTest | Add some elements and asserts if were added |
| heapMaximumTest | HeapTest | Add some elements and asserts if the max element added is de maximum of the heap |
| heapExtractMaxTes | HeapTest | Add some elements and extract the max added, asserting if was the one added and if was extracted of the heap |
| heapIncreaseKeyTest | HeapTest | Add some elements to a heap and increase the key value of an element, asserting the change |
| MaxHeapInsertTest | HeapTest | Add some elements and asserts the max heap array |
| HeapSortTest | HeapTest | Add some elements and sorts the heap, asserting the correct order |
| buildMaxHeapTest | HeapTest | Add some elements and sorts the heap, asserting the correct max order |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | HashMapTest | Makes a hashmap empty with 10 slots |
| Setup2 | HashMapTest | Makes a hashmap with 4 elements and 5 slots |
| isEmptyTest | HashMapTest | Asserts that the hash map is empty |
| isEmptyTest2 | HashMapTest | Adds an element in the empty hash and asserts that the hash is not empty |
| insertTest1 | HashMapTest | Adds an element in the hash map and asserts if was added |
| insertTest2 | HashMapTest | Adds an element in the hash map and asserts if was added with the size |
| insertTest3 | HashMapTest | Tries to add an element in the hash map and assert that cannot because the size is full |
| hashSearchTest1 | HashMapTest | Add an element in a hashmap and searchs it in the hash |
| hashSearchTest2 | HashMapTest | Tries to search an element in a hash that does not exist |
| hashSearchTest3 | HashMapTest | Tries to search an element with an empty key |
| deleteTest1 | HashMapTest | Deletes an element and asserts with the hash size |
| deleteTest2 | HashMapTest | Deletes an element and tries to search it in the hash |
| deleteTest3 | HashMapTest | Deletes an element and adds another element to increase the size |

\* El nombre de los escenarios puede ser setupStage1, setupStage2, etc.

\* La clase es la clase de testing correspondiente al modelo donde acontece el escenario. Por ejemplo si usted está probando User, clase será UserTest.

\* El escenario es la descripción de las condiciones iniciales del escenario.

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
|  |  |  |  |  |
|  |  |  |  |  |
|  |  |  |  |  |

…

\* Una prueba se compone de un conjunto de casos de prueba.

\* Cada fila representa un ***caso de prueba*** difente

\* En el objetivo de la prueba debe escribir una descripción sobre qué es lo que específicamente está probando del modelo del programa.

\* La clase es la clase del modelo que está siendo puesto a prueba.

\* El método es específicamente el método de la clase que está siendo puesto a prueba.

\* El escenario se refiere al nombre del escenario que usted definió. Todos los casos de prueba corresponden a escenarios.

\* Los valores de entrada son valores que entran al método puesto a prueba.

\* El resultado esperado es lo que se espera que suceda luego de ejecutar el método.