**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 |

…

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | StackTest | Makes two empty stacks |

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| Setup1 | HashMapTest | Makes a hashmap empty with 10 slots |
| Setup2 | HashMapTest | p1 = Mary sue", "456", 20, "BusinessClass", 30, "10", "A"  p2 = Mary on a cross", "111", 17, "BusinessClass", 20, "11", "A"  p4 = Oscar", "432", 18, "EconomicClass", 20, "22", "C"  p5 = “Makoto", "555", 17, "FirstClass", 40, "15", "B"  hashmap with 5 slots |

\* 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:** Verify the methods of the controller class | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| ControllerTest | confirmAssistance | ConfirmAssistancePassengerNotFoundTest | String id = “123”, ex = “Passenger ID has not been found”, a = confirmAssitance(id) | Uses the setup and verifies if an inexistent passenger id can be confirmed. It should not be found |
| ControllerTest | generateEntryList | GenerateEntryListNotConfirmedPassengersTest | String ex = “There are not confirmed passengers yet”  List = generateEntryList() | Uses the setup and verifies if the passengers have confirmed creating an entry list. It should not find any confirmed passengers |
| ControllerTest | addtoExistList | AddToExistListNotConfirmedPassengersTest | Ex = “There are not confirmed passengers!”  List = addToExistList() | Uses the setup and verifies if any passenger confirmed can be added in a list. It should not be found any confirmed passenger |
| ControllerTest | importDataFromCSV | ImportDataFromCSVLoadedTest | String ex = “Succesful import!”  Ex = “The data base has already Loaded”  A = importDataFromCSV() | Uses the setup and verifies if the Import can be done and later it tries to import again It should say that is already loaded |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verifies the methods of Passenger class | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PassengerTest | Passenger | PassengerCreatedSuccesfullyTest | “John” “123” 27 “TouristClass” 0 “1” “A” | Creates a new Passenger and confirms it his information of the registration equals with the created one |
| PassengerTest | Passenger | PassengerCreatedSuccessBTest | “Jane” “456” 22 “TouristClass” 10 “1” “B” | 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 |
| PassengerTest | isConfirmed | ConfirmAssistanceTest | “Mary” “111” 30 “BusinessClass” 20 “3” | Creates a new Passenger and asserts if the passenger is confirmed, it should not, and later confirms the passenger and verify it. |
| PassengerTest | compareTo | CompareToPassengerPriorityTest | "Mary sue", "456", 20, "BusinessClass", 30, "10", "A"  "Gary Stu", "123", 21, "TouristClass", 50, "20", "B"  "John Shepard", "777", 29, "FirstClass", 70, "7", "N" | Creates three Passengers and compares between their priority, being lower or higher among them |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** Verifies the methods of Stack class | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| StackTest | IsEmpty | isEmptyTest1 | Setup1() | Asserts that the stack is empty |
| StackTest | IsEmpty | isEmptyTest2 | Setup1()  p1 = “Mary sue” "456", 20, "BusinessClass", 30, "10", "A" | Asserts that the stack is not empty because an element was added |
| StackTest | isEmpty | isEmptyTest3 | Setup1()  p1 = “Mary sue” "456", 20, "BusinessClass", 30, "10", "A" | Asserts that the stack is empty adding an element and poping it later |
| StackTest | getTop | isTopTest1 | Setup1() | Tries to obtain the element on an empty stack |
| StackTest | getTop | isTopTest2 | Setup1()  p1 = “Mary sue” "456", 20, "BusinessClass", 30, "10", "A"  p2 = "Jane", "456", 22, "TouristClass", 10, "1", "B" | Adds two elements and the top one should be the second added |
| StackTest | pop | isPopTest1 | Setup1() | Tries to pop an empty stack |
| StackTest | pop | isPopTest2 | Setup1()  p1 = “Mary sue” "456", 20, "BusinessClass", 30, "10", "A" | Adds an element and later pops it asserting that the stack is empty now |
| StackTest | pop | isPopTest3 | Setup1()  p1 = “Mary sue” "456", 20, "BusinessClass", 30, "10", "A"  p2 = "Jane", "456", 22, "TouristClass", 10, "1", "B" | Adds two elements and pops one, so the second one should be popped |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| HeapTest | insertElement | inserElementTest | Heap H1 = 1,2,3 | Add some elements and asserts if were added |
| HeapTest | heapMaximum | heapMaximumTest | Heap h1= 6,5,4 | Add some elements and asserts if the max element added is de maximum of the heap |
| HeapTest | heapExtractMax | heapExtractMaxTest | Heap h2 = 4,3,2 | Add some elements and extract the max added, asserting if was the one added and if was extracted of the heap |
| HeapTest | heapIncreaseKey | heapIncreaseKeyTest | Heap h1 = 5,7,1 | Add some elements to a heap and increase the key value of an element, asserting the change |
| HeapTest | maxHeapInsert | MaxHeapInsertTest | Heap h1 = 5, 7, 1 | Add some elements and asserts the max heap array |
| HeapTest | heapSort | HeapSortTest | Heap h2 = 5, 7, 1 | Add some elements and sorts the heap, asserting the correct order |
| HeapTest | buildMaxHeap | buildMaxHeapTest | Heap h1 = 5, 7, 1 | Add some elements and sorts the heap, asserting the correct max order |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Objetivo de la Prueba:** | | | | |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| HashMapTest | isEmpty | isEmptyTest | Setup1 | Asserts that the hash map is empty |
| HashMapTest | isEmpty | isEmptyTest2 | Setup1 | Adds an element in the empty hash and asserts that the hash is not empty |
| HashMapTest | hashInsert | insertTest1 | Setup1  P1 = "Gary stu", "112", 20, "BusinessClass", 10, "10", "G" | Adds an element in the hash map and asserts if was added |
| HashMapTest | hashInsert | insertTest2 | Setup2  P6 = "John Shepard", "777", 29, "FirstClass", 70, "7", "N" | Adds an element in the hash map and asserts if was added with the size |
| HashMapTest | hashInsert | insertTest3 | Setup1  P6 = "John Shepard", "777", 29, "FirstClass", 70, "7", "N" | Tries to add an element in the hash map and assert that cannot because the size is full |
| HashMapTest | hashSearch | hashSearchTest1 | Setup2  "John", "123", 27, "TouristClass", 0, "1", "A" | Add an element in a hashmap and searchs it in the hash |
| HashMapTest | hashSearch | hashSearchTest2 | Setup2  P6 = "John Shepard", "777", 29, "FirstClass", 70, "7", "N" | Tries to search an element in a hash that does not exist |
| HashMapTest | hashSearch | hashSearchTest3 | Setup2 | Tries to search an element with an empty key |
| HashMapTest | delete | deleteTest1 | Setup2  “Mary sue” | Deletes an element and asserts with the hash size |
| HashMapTest | delete | deleteTest2 | Setup2  “Mary sue” | Deletes an element and tries to search it in the hash |
| HashMapTest | delete | deleteTest3 | Setup2  “Oscar”  P6 = "John Shepard", "777", 29, "FirstClass", 70, "7", "N"  P7 = "Jane Shepard", "777", 29, "FirstClass", 70, "7", "N" | Deletes an element and adds another element to increase the size |

…

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

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

\* 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.