**Formato de escenarios y casos de uso (pruebas unitarias)**

**HashTable**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp | HashTableTests | Inicializa una tabla hash que luego será usada en los demás test planteados. |
|  |  |  |
|  |  |  |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| HashTableTest | testPutNuevoElemento | Agregar un nuevo elemento | Clave: "clave1", Valor: 1 | El tamaño debería ser 1 después de agregar un elemento |
| HashTableTest | testPutSobrescribirElemento | Sobrescribir un elemento existente | Clave: "clave1", Valor inicial: 1, Nuevo valor: 2 | El valor asociado con 'clave1' debería ser 2 después de sobrescribir |
| HashTableTest | testPutElementosColision | Agregar elementos con colisión | Claves: "clave1", "clave2", Valores: 1, 2 | El tamaño debería ser 2 incluso si hay colisión |
| HashTableTest | testGetExistente | Obtener un elemento existente | Clave: "clave1" Valor : 1 | Debería devolver el valor 1 asociado con "clave1" |
| HashTableTest | testGetNoExistente | Obtener un elemento no existente | Clave: "clave1" | Debería devolver null para una clave que no existe |
| HashTableTest | testGetDespuesDeColision | Obtener un elemento después de una colisión | Clave: "clave2" | Debería devolver el valor 2 incluso después de una colisión |
| HashTableTest | testRemoveExistente | Eliminar un elemento existente | Clave: "clave1" | Después de eliminar, debería devolver null para "clave1 |
| HashTableTest | testRemoveNoExistente | Eliminar un elemento no existente | Clave: "clave1" | El tamaño debería seguir siendo 0 después de intentar eliminar una clave no existente |
| HashTableTest | testRemoveEnColision | Eliminar un elemento con colisión | Clave: "clave1" | Después de eliminar, "clave2" debería seguir existiendo |
| HashTableTest | testSizeVacio | Comprobar el tamaño de una tabla vacía | Ninguno | El tamaño debería ser 0 para una tabla vacía |
| HashTableTest | testSizeConElementos | Comprobar el tamaño después de agregar elementos | Claves: "clave1", "clave2", Valores: 1, 2 | El tamaño debería ser 2 después de agregar dos elementos |
| HashTableTest | testSizeDespuesDeRemove | Comprobar el tamaño después de eliminar un elemento | Claves: "clave1", "clave2", Valor eliminado: "clave1" | El tamaño debería ser 1 después de eliminar un elemento |

**Queue**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp | QueueTest | Inicializa una queue que luego será usada en los demás test planteados. |
|  |  |  |
|  |  |  |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| QueueTest | testEnqueueElemento | Agregar un elemento | Elemento: 1 | El frente de la cola debería ser el primer elemento añadido |
| QueueTest | testEnqueueVariosElementos | Agregar varios elementos | Elementos: 1, 2, 3 | El tamaño de la cola debería ser 3 después de añadir tres elementos |
| QueueTest | testEnqueueNull | Agregar un elemento nulo | Elemento: null | El frente de la cola debería ser null después de agregar un elemento nulo |
| QueueTest | testDequeueVacio | Eliminar de una cola vacía | Ninguno | Debería devolver null si la cola está vacía |
| QueueTest | testDequeueElemento | Eliminar un elemento de la cola | Elemento: 1 | Debería devolver el primer elemento si la cola tiene elementos |
| QueueTest | testDequeueDespuesDeVariosEnqueue | Eliminar después de varios enqueue | Elementos: 1, 2, 3 | El nuevo frente debería ser 2 después de eliminar un elemento |
| QueueTest | testFrontVacio | Obtener el frente de una cola vacía | Elemento: 1 | Debería devolver el primer elemento si la cola tiene uno |
| QueueTest | testFrontNoModificaTamaño | Obtener el frente sin modificar el tamaño | Elemento: 1 | El tamaño de la cola no debería cambiar al obtener el frente |
| QueueTest | testIsEmptyVerdadero | Verificar si una cola está vacía | Ninguno | Debería devolver verdadero si la cola está vacía |
| QueueTest | testIsEmptyFalso | Verificar si una cola no está vacía | Elemento: 1 | Debería devolver falso si la cola tiene elementos |
| QueueTest | testIsEmptyDespuesDeDequeue | Verificar si la cola está vacía después de eliminar elementos | Elementos: 1 | Debería devolver verdadero si la cola se vació después de eliminar elementos |
| QueueTest | testSizeVacio | Obtener el tamaño de una cola vacía | Ninguno | Debería devolver 0 si la cola está vacía |
| QueueTest | testSizeUnElemento | Obtener el tamaño de una cola con un elemento | Elemento: 1 | Debería devolver 1 si la cola tiene un elemento |
| QueueTest | testSizeDespuesDeDequeue | Obtener el tamaño después de eliminar un elemento | Elementos: 1, 2 | Debería devolver 1 después de eliminar un elemento en una cola con dos elementos |
|  |  |  |  |  |
|  |  |  |  |  |

**Stack**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp | StackTests | Inicializa un stack que luego será usada en los demás test planteados. |
|  |  |  |
|  |  |  |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| StackTest | testPushElemento | Agregar un elemento | Elemento: 1 | La cima de la pila debería ser el último elemento añadido |
| StackTest | testPushVariosElementos | Agregar varios elementos | Elementos: 1, 2, 3 | El tamaño de la pila debería ser 3 después de añadir tres elementos |
| StackTest | testPushNull | Agregar un elemento nulo | Elemento: null | La cima de la pila debería ser null después de agregar un elemento nulo |
| StackTest | testPopVacio | Eliminar de una pila vacía | Ninguno | Debería devolver null si la pila está vacía |
| StackTest | testPopElemento | Eliminar un elemento de la pila | Elemento: 1 | Debería devolver el primer elemento si la pila tiene elementos |
| StackTest | testPopDespuesDeVariosPush | Eliminar después de varios push | Elementos: 1, 2, 3 | La nueva cima debería ser 2 después de eliminar un elemento |
| StackTest | testPeekVacio | Obtener la cima de una pila vacía | ninguno | Debería devolver null si la pila está vacía |
| StackTest | testPeekElemento | Obtener la cima de una pila con un elemento | Elemento: 1 | Debería devolver el primer elemento si la pila tiene uno |
| StackTest | testPeekNoModificaTamaño | Obtener la cima sin modificar el tamaño | Elemento: 1 | El tamaño de la pila no debería cambiar al obtener la cima |
| StackTest | testIsEmptyVerdadero | Verificar si una pila está vacía | Ninguno | Debería devolver verdadero si la pila está vacía |
| StackTest | testIsEmptyFalso | Verificar si una pila no está vacía | Elementos: 1 | Debería devolver falso si la pila tiene elementos |
| StackTest | testIsEmptyDespuesDePop | Verificar si la pila está vacía después de eliminar elementos | Elementos: 1 | Debería devolver verdadero si la pila se vació después de eliminar elementos |
| StackTest | testSizeVacio | Obtener el tamaño de una pila vacía | Ninguno | Debería devolver 0 si la pila está vacía |
| StackTest | testSizeUnElemento | Obtener el tamaño de una pila con un elemento | Elemento: 1 | Debería devolver 1 si la pila tiene un elemento |
| StackTest | testSizeDespuesDePop | Obtener el tamaño después de eliminar un elemento | Elementos: 1, 2 | Debería devolver 1 después de eliminar un elemento en una pila con dos elementos |
|  |  |  |  |  |

**Priority Queue**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp | PriorityQueueTest | Inicializan dos priority queue que luego serán usadas en los demás test planteados. |
|  |  |  |
|  |  |  |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| PriorityQueueHTest | testEnqueueAndDequeue | Agregar y eliminar nodos | Nodos con datos y prioridades | Debería agregar y eliminar nodos correctamente |
| PriorityQueueHTest | testPeek | Obtener el nodo principal con mayor prioridad | Nodos con datos y prioridades | Debería devolver el nodo principal sin eliminarlo |
| PriorityQueueHTest | testRemove | Eliminar un nodo específico | Nodos con datos y prioridades | Debería eliminar el nodo especificado correctamente |
| PriorityQueueHTest | testAssignPriority | Asignar nueva prioridad a los nodos | Nodos con datos y prioridades | Debería reorganizar la cola según las nuevas prioridades |

**Controller**

**Configuración de los Escenarios**

|  |  |  |
| --- | --- | --- |
| **Nombre** | **Clase** | **Escenario** |
| setUp | ControllerTest | Se inicializa un nuevo Controller y se reparten cartas a cuatro jugadores usando el método propio de Controller |
|  |  |  |
|  |  |  |

**Diseño de Casos de Prueba**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Clase** | **Método** | **Escenario** | **Valores de Entrada** | **Resultado esperado** |
| ControllerTest | topCardProof | Verificar la carta superior | Ninguno | Debería devolver la carta superior de tipo NORMAL |
| ControllerTest | testPlayCard | Probar jugar una carta | Ninguno | Debería jugar una carta si es válida y actualizar el estado del juego correctamente |
| ControllerTest | testDrawCard | Probar robar una carta | Ninguno | Debería robar una carta del mazo y actualizar el estado del juego correctamente |
| ControllerTest | testCreatePlayersAndDealCards | Probar crear jugadores y repartir cartas | Número de jugadores | Debería crear la cantidad correcta de jugadores y repartirles la cantidad correcta de cartas |
| ControllerTest | testNextPlayer | Probar pasar al siguiente jugador | Ninguno | Debería avanzar al siguiente jugador correctamente |