DISEÑO DE PRUEBAS UNITARIAS

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 1** | **Objetivo**: Probar el método de búsqueda del árbol rojinegro. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + search(K): V | Buscar una acción que se encuentra en el mercado. | Nombre de la acción. | El método devolvió la acción que se buscaba. |
| RBTree | + search(K): V | Buscar una acción que no se encuentre en el mercado. | Nombre de la acción. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 2** | **Objetivo**: Probar el método que indica si un elemento está en el árbol. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + isInTree(K): boolean | Comprobar si existe una acción que en verdad está en un mercado. | Nombre de la acción. | El método devolvió true. |
| RBTree | + isInTree(K): boolean | Comprobar si existe una acción que no está en un mercado. | Nombre de la acción. | El método devolvió false. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 3** | **Objetivo**: Probar el método que devuelve el valor mínimo. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + getMin(): K | Buscar la acción con el precio más bajo en un mercado con más de una acción. | Ninguna. | El método devolvió la acción con el precio más bajo. |
| RBTree | + getMin(): K | Buscar la acción con el precio más bajo en un mercado con solo una acción. | Ninguna. | El método devolvió esa misma acción que es la única en ese mercado. |
| RBTree | + getMin(): K | Buscar la acción con el precio más bajo en un mercado vacío (sin acciones en él). | Ninguna. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 4** | **Objetivo**: Probar el método que devuelve el valor máximo. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + getMax(): K | Buscar la acción con el precio más alto en un mercado con más de una acción. | Ninguna. | El método devolvió la acción con el precio más alto. |
| RBTree | + getMax(): K | Buscar la acción con el precio más alto en un mercado con solo una acción. | Ninguna. | El método devolvió esa misma acción que es la única en ese mercado. |
| RBTree | + getMax(): K | Buscar la acción con el precio más alto en un mercado vacío (sin acciones en él). | Ninguna. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 5** | **Objetivo**: Probar el método que devuelve el predecesor de un nodo. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + getPredecessor(K): K | Buscar el predecesor de la raíz de un árbol con un peso mayor a 1. | Clave de la raíz del árbol. | El método devolvió el nodo mayor del subárbol izquierdo de la raíz. |
| RBTree | + getPredecessor(K): K | Buscar el predecesor de la raíz de un árbol con peso 1. | Clave de la raíz del árbol. | El método devolvió el mismo nodo. |
| RBTree | + getPredecessor(K): K | Buscar el predecesor de la raíz del árbol vacío. | Clave de la raíz del árbol. | El método devolvió null. |
| RBTree | + getPredecessor(K): K | Buscar el predecesor de un nodo que no está en el árbol. | Clave de la raíz del árbol. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 6** | **Objetivo**: Probar el método que devuelve el sucesor de un nodo. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + getSuccessor(K): K | Buscar el sucesor de la raíz de un árbol con un peso mayor a 1. | Clave de la raíz del árbol. | El método devolvió el nodo menor del subárbol derecho de la raíz. |
| RBTree | + getSuccessor(K): K | Buscar el sucesor de la raíz de un árbol con peso 1. | Clave de la raíz del árbol. | El método devolvió el mismo nodo. |
| RBTree | + getSuccessor(K): K | Buscar el sucesor de la raíz del árbol vacío. | Clave de la raíz del árbol. | El método devolvió null. |
| RBTree | + getSuccessor(K): K | Buscar el sucesor de un nodo que no está en el árbol. | Clave de la raíz del árbol. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 7** | **Objetivo**: Probar el método de insertar. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + insert(K, V): void | Insertar una acción en un árbol no vacío. | Nombre de la acción y su precio. | Se agregó la acción al mercado. |
| RBTree | + insert(K, V): void | Insertar una acción en un árbol vacío. | Nombre de la acción y su precio. | Se agregó la acción al mercado. |
| RBTree | + insert(K, V): void |  |  |  |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 8** | **Objetivo**: Probar el método de eliminar. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| RBTree | + delete(K): V | Eliminar la raíz en un árbol de peso mayor a 1 | Nombre de la acción a eliminar. | El método eliminó el nodo del árbol y lo devolvió. |
| RBTree | + delete(K): V | Eliminar la raíz de un árbol de peso 1. | Nombre de la acción a eliminar. | El método eliminó el nodo del árbol y lo devolvió. |
| RBTree | + delete(K): V | Eliminar la raíz de un árbol vacío. | Nombre de la acción a eliminar. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 7** | **Objetivo**: Probar el método offer de la PriorityQueue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| PriorityQueue | + offer(V): void | Agregar un nodo nuevo. | Nodo. | El método agregó  correctamente el nuevo nodo. |
| PriorityQueue | + offer(V): void | Agregar un nodo que ya existe. | Nodo que ya está en la cola. | El método agregó el nuevo nodo y lo acomodó para cumplir la propiedad de la cola  de prioridad. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 8** | **Objetivo**: Probar el método peek de la PriorityQueue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| PriorityQueue | + peek(): V | Solicitar el primer nodo de una cola no vacía con más de  un elemento. | Ninguna. | El método devolvió el primer nodo. |
| PriorityQueue | + poll(): V | Solicitar el primer nodo de  una cola de tamaño 1. | Ninguna. | El método devolvió el único  nodo que tiene la cola. |
| PriorityQueue | + poll(): V | Solicitar el primer nodo de  una cola vacía. | Ninguna. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 9** | **Objetivo**: Probar el método poll de la PriorityQueue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| PriorityQueue | + poll(): V | Pedir el primer nodo de una cola no vacía con más de un elemento. | Ninguna. | El método devolvió el primer nodo y lo eliminó de la cola, dejando de primero al que tiene el valor más grande  después del que se eliminó. |
| PriorityQueue | + poll(): V | Pedir el primer nodo de una cola de tamaño 1. | Ninguna. | El método devolvió el único nodo que tiene la cola y lo eliminó de esta, dejando a la  cola vacía. |
| PriorityQueue | + poll(): V | Pedir el primer nodo de una  cola vacía. | Ninguna. | El método devolvió null y no  eliminó nada. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 7** | **Objetivo**: Probar el método offer de la Queue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Queue | + offer(V): void | Agregar un nodo nuevo. | Nodo. | El método agregó  correctamente el nuevo nodo. |
| Queue | + offer(V): void | Agregar un nodo que ya existe. | Nodo que ya está en la cola. | El método agregó el nuevo  nodo y lo acomodó a después del ya existía en la cola. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 8** | **Objetivo**: Probar el método peek de la Queue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Queue | + peek(): V | Solicitar el primer nodo de  una cola no vacía con más de un elemento. | Ninguna. | El método devolvió el primer nodo. |
| Queue | + poll(): V | Solicitar el primer nodo de  una cola de tamaño 1. | Ninguna. | El método devolvió el único  nodo que tiene la cola. |
| Queue | + poll(): V | Solicitar el primer nodo de  una cola vacía. | Ninguna. | El método devolvió null. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 9** | **Objetivo**: Probar el método poll de la Queue. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Queue | + poll(): V | Pedir el primer nodo de una cola no vacía con más de un elemento. | Ninguna. | El método devolvió el primer nodo y lo eliminó de la cola, dejando de primero al que estaba después del que se  eliminó. |
| Queue | + poll(): V | Pedir el primer nodo de una cola de tamaño 1. | Ninguna. | El método devolvió el único nodo que tiene la cola y lo eliminó de esta, dejando a la  cola vacía. |
| Queue | + poll(): V | Pedir el primer nodo de una  cola vacía. | Ninguna. | El método devolvió null y no  eliminó nada. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 10** | **Objetivo**: Probar el método push de la Stack. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Stack | + push(E): V | Insertar un nuevo nodo en una pila no vacía con varios elementos. | Nuevo nodo. | El método agregó correctamente el nuevo nodo en el tope de la pila. |
| Stack | + push(E): V | Insertar un nuevo nodo en una pila vacía. | Nuevo nodo. | El método agregó  correctamente el nuevo nodo en el tope de la pila. |
| Stack | + push(E): V | Insertar un nodo que ya está en una pila no vacía con  varios elementos. | Nodo repetido. | El método agregó correctamente el nuevo nodo  en el tope de la pila. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 11** | **Objetivo**: Probar el método pop de la Stack. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Stack | + pop(): V | Pedir el primer nodo de una pila no vacía con más de un elemento. | Ninguna. | El método devolvió el primer nodo y lo eliminó de la pila, dejando en el tope al que estaba detrás del que se  eliminó. |
| Stack | + pop(): V | Pedir el primer nodo de una pila de tamaño 1. | Ninguna. | El método devolvió el único nodo que tiene la pila y lo eliminó de esta, dejando a la  pila vacía. |
| Stack | + pop(): V | Pedir el primer nodo de una  pila vacía. | Ninguna. | El método devolvió null y no  eliminó nada. |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Prueba N° 12** | **Objetivo**: Probar el método top de la Stack. | | | |
| **Clase** | **Método** | **Escenario** | **Entradas** | **Resultado** |
| Stack | + top(): V | Pedir el primer nodo de una pila no vacía con más de un elemento. | Ninguna. | El método devolvió el nodo que está en el tope. |
| Stack | + top(): V | Pedir el primer nodo de una  cola de tamaño 1. | Ninguna. | El método devolvió el nodo  que está en el tope. |
| Stack | + top(): V | Pedir el primer nodo de una  cola vacía. | Ninguna. | El método devolvió null. |