DISEÑO DE PRUEBAS UNITARIAS

Árbol Rojinegro

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.

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° 9	**Objetivo**: Probar el método leftRotate.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
RBTree	+ leftRotate(RBNode): void	Rotar una hoja.	Nodo.	El método rotó la hoja hacia la izquierda.
RBTree	+ leftRotate(RBNode): void	Rotar la raíz.	Nodo raíz.	El método rotó la raíz hacia la izquierda.
RBTree	+ leftRotate(RBNode): void	Rotar un nodo que no existe.	Nodo inexistente en el árbol.	El método no hace nada.

Prueba N° 10	**Objetivo**: Probar el método rightRotate.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
RBTree	+ rightRotate(RBNode): void	Rotar una hoja.	Nodo.	El método rotó la hoja hacia la derecha.
RBTree	+ rightRotate(RBNode): void	Rotar la raíz.	Nodo raíz.	El método rotó la raíz hacia la derecha.
RBTree	+ rightRotate(RBNode): void	Rotar un nodo que no existe.	Nodo inexistente en el árbol.	El método no hace nada.

Árbol AVL

Prueba N° 11	**Objetivo**: Probar el método que devuelve el valor mínimo.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**

AVLTree	+ 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.
AVLTree	+ 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.
AVLTree	+ 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° 12	**Objetivo**: Probar el método que devuelve el valor máximo.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
AVLTree	+ 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.
AVLTree	+ 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.
AVLTree	+ 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**
AVLTree	+ 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.
AVLTree	+ 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.

AVLTree	+ getPredecessor(K): K	Buscar el predecesor de la raíz del árbol vacío.	Clave de la raíz del árbol.	El método devolvió null.
AVLTree	+ 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° 13	**Objetivo**: Probar el método que devuelve el sucesor de un nodo.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
AVLTree	+ 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.
AVLTree	+ 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.
AVLTree	+ getSuccessor(K): K	Buscar el sucesor de la raíz del árbol vacío.	Clave de la raíz del árbol.	El método devolvió null.
AVLTree	+ 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° 14	**Objetivo**: Probar el método de búsqueda del árbol AVL.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**

AVLTree	+ search(K): V	Buscar una acción que se encuentra en el mercado.	Nombre de la acción.	El método devolvió la acción que se buscaba.
AVLTree	+ search(K): V	Buscar una acción que no se encuentre en el mercado.	Nombre de la acción.	El método devolvió null.

Prueba N° 15	**Objetivo**: Probar el método que indica si un elemento está en el árbol.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
AVLTree	+ isInTree(K): boolean	Comprobar si existe una acción que en verdad está en un mercado.	Nombre de la acción.	El método devolvió true.
AVLTree	+ isInTree(K): boolean	Comprobar si existe una acción que no está en un mercado.	Nombre de la acción.	El método devolvió false.

Prueba N° 16	**Objetivo**: Probar el método de insertar.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
AVLTree	+ 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.
AVLTree	+ insert(K, V): void	Insertar una acción en un árbol vacío.	Nombre de la acción y su precio.	Se agregó la acción al mercado.

Prueba N° 17	**Objetivo**: Probar el método de eliminar.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
AVLTree	+ 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.
AVLTree	+ 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.
AVLTree	+ delete(K): V	Eliminar la raíz de un árbol vacío.	Nombre de la acción a eliminar.	El método devolvió null.