DISEÑO DE PRUEBAS UNITARIAS

Prueba N° 1	**Objetivo**: Probar el método de búsqueda de la tabla hash.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
HashMap	+ get(K): V	Buscar un libro con ejemplares disponibles en la estantería.	Libro con ejemplares	El método devolvió el libro que se buscaba.
HashMap	+ get(K): V	Buscar un libro que no tenga ejemplares.	Libro agotado.	El método devolvió null.

Prueba N° 2	**Objetivo**: Probar el método que cambia el valor de una clave de la tabla hash.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
HashMap	+ changeValue(K, V): void	Cambiar un libro que esté en la tabla.	Libro con ejemplares	El método cambió el libro que estaba en la tabla hash con el que se pasó por parámetro en el método.
HashMap	+ changeValue(K, V): void	Cambiar un libro que no esté en la tabla.	Libro agotado.	El método agregó ese nuevo libro en lugar de reemplazarlo.

Prueba N° 3	**Objetivo**: Probar el método que indica si un libro está en la tabla hash.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
HashMap	+ contains(K): boolean	Ver si un libro está en la tabla hash.	Clave que está asociada a un libro en la tabla	El método devolvió el valor true.
HashMap	+ changeValue(K, V): void	Ver si un libro está en la tabla hash.	Clave que no está asociada a ningún libro de la tabla.	El método devolvió el valor false.

Prueba N° 4	**Objetivo**: Probar que en verdad se elimina un elemento de la tabla hash.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
HashMap	+ delete(K): boolean	Eliminar un libro que se encuentra en la tabla hash.	Libro.	El método devolvió true.
	+ get(K): V	Buscar el libro que se acaba de eliminar.	Libro eliminado.	El método devolvió null.
HashMap	+ delete(K): boolean	Eliminar un libro que no se encuentra en la tabla hash.	Libro.	El método devolvió false.

Prueba N° 5	**Objetivo**: Probar que el programa muestra un mensaje cuando no haya cajeros disponibles.			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
Reception	+ Reception(String)	El número de cajeros disponibles es cero.	Texto plano con la segunda línea en 0.	Mensaje que indica que no hay cajeros disponibles.

Prueba N° 6	**Objetivo**: Probar que se inserta correctamente un elemento en la tabla hash			
Clase	**Método**	**Escenario**	**Entradas**	**Resultado**
HashTable	+ put(K, V): void	Insertar un libro con diferente ISBN.	Clave que no esté asociada con algún libro ya agregado en la tabla.	El método agregó correctamente el nuevo libro.
HashTable	+ put(K, V): void	Insertar un libro con el mismo ISBN que otro.	Clave que ya esté asociada con algún libro agregado en la tabla.	El método reemplazó el libro antiguo con el nuevo que se le acaba de agregar.

HashTable	+ put(K, V): void	Insertar el mismo libro con diferente ISBN.	Clave diferente pero con un libro que ya esté en la tabla hash.	El método agregó el mismo libro pero con ISBN diferente.

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.