K-Lista

Una KList es una colección de elementos que soporta las operaciones: Add, Remove, MoveFirstTo, Rotate, pudiendo deshacerlas y rehacerlas.

Programe una biblioteca de clases EjercicioKList.dll donde defina una clase genérica Examen.KList<T> que implemente la interfaz Utils.IKList<T> suministrada en la biblioteca Utils.dll.

A continuación se describen los miembros de la interfaz y la lógica esperada para cada uno de ellos.
Usted deberá garantizar que su lista tenga un constructor público que no reciba parámetros.

Miembro	**Descripción**
`void Add(T element)`	Agrega `element` al final de la colección.
`bool Remove(T element)`	Elimina la primera ocurrencia de `element` en la lista. Devolverá `true` si puede eliminar este elemento, `false` en caso contrario.
`void MoveFirstTo(int k)`	Mueve el primer elemento a la posición k. Ejemplo: **Lista** : **1** 2 3 4 5 **MoveFirstTo(2):** 2 3 **1** 4 5. De no haber k elementos lo mueve al final. De estar vacía la lista lanza una excepción: `InvalidOperationException`
`void Rotate(int k)`	Desplaza los k primeros elementos hacia el final de la colección. Si k es mayor que la cantidad de elementos la lista permanece igual (note en este caso, que si en el paso siguiente se hace un Undo la lista también permanece igual). Ejemplo: **Lista** : **1 2 3** 4 5 **Rotate(3)** : 4 5 **1 2 3**. De estar vacía la lista lanza una excepción: `InvalidOperationException`

Miembro	Descripción
`void Undo()`	Deshace la última operación realizada. Es equivalente a realizar la acción opuesta a la última operación sobre la lista. Ejemplo: **Lista:** 1 4 2 4 **Add (5):** 1 4 2 4 5 **Add (7):** 1 4 2 4 5 7 **Remove (4):** 1 2 4 5 7 **Undo:** 1 4 2 4 5 7 (Recupera el 4 eliminado) **Undo:** 1 4 2 4 5 (Elimina el 7 adicionado) **Undo:** 1 4 2 4 (Elimina el 5 adicionado) Se podrá ejecutar mientras queden operaciones por deshacer. De lo contrario lanzará la excepción `InvalidOperationException`

Miembro	**Descripción**
`void Redo()`	Revierte el efecto del último `Undo`, el cual tiene que haberse realizado justo en el paso anterior (Ejemplo 1). Si en los últimos k pasos se hicieron `Undo`, entonces se pueden hacer hasta k `Redo` consecutivos (Ejemplo 2). Un `Redo` también puede ser deshecho por un `Undo` (Ejemplo 3). En general se puede hacer `Redo`, si hasta ese momento la cantidad de operaciones `Undo` es mayor que la cantidad de operaciones `Redo` y la última operación no es `Add`, `Remove`, `Rotate` o `MoveFirstTo` De no poder aplicarse debe lanzar la excepción `InvalidOperationException` Ejemplos: 1. **Lista:** 3, 5, 7, 9, 11 **Remove(5):** 3, 7, 9, 11 **MoveFirstTo(1):** 7, 3, 9, 11 **Undo:** 3, 7, 9, 11 **Add(8):** 3, 7, 9, 11, 8 **Redo:** `InvalidOperationException` 2. **Lista:** 3, 5, 7, 9, 11 **Remove(5):** 3, 7, 9, 11 **MoveFirstTo(1):** 7, 3, 9, 11 **Undo:** 3, 7, 9, 11 **Undo:** 3, 5, 7, 9, 11 **Redo:** 3, 7, 9, 11 **Redo:** 7, 3, 9, 11 **Redo:** `InvalidOperationException` 3. **Lista:** 3, 5, 7, 9, 11 **Remove(5):** 3, 7, 9, 11 **MoveFirstTo(1):** 7, 3, 9, 11 **Undo:** 3, 7, 9, 11 **Redo:** 7, 3, 9, 11 **Undo:** 3, 7, 9, 11
`int Count { get; }`	Cantidad de elementos.
`T this[int index]`	Devuelve el valor que está en la posición `index` de la lista. Si `index` no es una posición válida usted deberá lanzar `IndexOutOfRangeException`

Miembro	**Descripción**
`IEnumerator<T> GetEnumerator()`	En la iteración se deben obtener los elementos en el mismo orden en que están posicionados en ese instante en la colección.
`IEnumerable<T> HistoricalElements()`	Devuelve todos los elementos presentes en la colección en el orden en que se encuentran y además, aquellos que se pueden obtener como resultado de hacer Undo una o más veces en el orden en que van reapareciendo. Ejemplo: **Lista:** 3, 5, 7, 9, 11 **Remove(5):** 3, 7, 9, 11 **MoveFirstTo(1):** 7, 3, 9, 11 **Undo:** 3, 7, 9, 11 **Add(8):** 3, 7, 9, 11, 8 **Add(9):** 3, 7, 9, 11, 8, 9 **Remove(9):** 3, 7, 11, 8, 9 **HistoricalElements():** 3, 7, 11, 8, 9, 9, 5

Nota: Asuma para todos los enumeradores que implemente, que no se le modificará la colección mientras está siendo iterada.