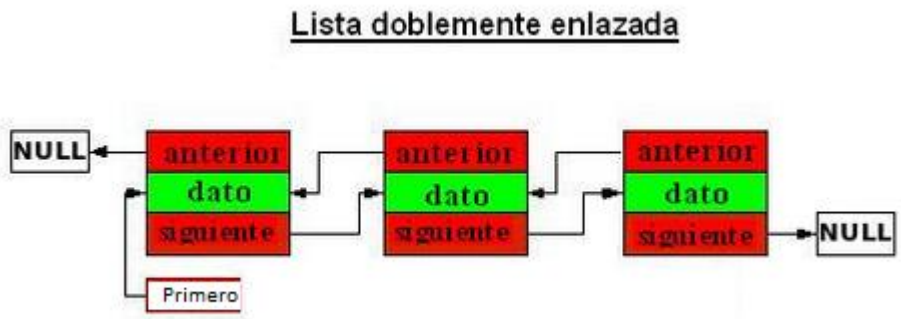
Tarea Programación Genérica:



Las listas doblemente enlazadas son semejantes a las listas enlazadas simples, pero algo más complejas. En las listas doblemente enlazadas el enlace entre los elementos se hace gracias a "**dos punteros"** (uno que apunta hacia el elemento anterior y otro que apunta hacia el elemento siguiente):

El puntero **anterior** del primer elemento debe apuntar hacia **NULL** (el inicio de la lista).

El puntero **siguiente** del último elemento debe apuntar hacia **NULL** (el fin de la lista).

Para acceder a un elemento, la lista puede ser recorrida en ambos sentidos: comenzando por el inicio, el puntero **siguiente** permite el desplazamiento hacia el próximo elemento; comenzando por el final, el puntero **anterior** permite el desplazamiento hacia el elemento anterior.

Resumiendo: el desplazamiento se hace en **ambas direcciones**, del primer al último elemento y/o del último al primer elemento.

Tarea:

Usando programación genérica implementa las clases necesarias para una lista doblemente enlazada.

Los atributos de la clase Nodo serán:

- Dato de tipo genérico (privado).
- **Anterior** de tipo Nodo<T> (público)
- **Siguiente** de tipo Nodo<T> (público)

Cuando se crea un nuevo Nodo, aún sin enlazar en la lista, Anterior y Siguiente apuntan a NULL.

Los métodos de la clase nodo serán:

- Constructor que reciba un dato (anterior y siguiente a null)
- Métodos get y set de dato.

Los atributos de la clase ListaDobleEnlazada serán:

- Primero de tipo Nodo<T> (privado).

- Size de tipo entero (privado).

Los métodos que se deben implementar para la lista doblemente enlazada son los siguientes:

- Constructor que cree una lista vacía.
- Sobreescribir método toString para mostrar todos los elementos de la lista.
- Todos los métodos indicados a continuación:

```
/**
 * Insertar un nodo al final de la lista
 * @param v el valor del nodo a insertar
 * @return el nuevo tamaño de la lista
 */
int push(T v);
/**
 * Insertar un nodo al inicio de la lista
 * @param v el valor del nodo a insertar
 * @return el nuevo tamaño de la lista
 */
int unshift(T v);
```

```
/**
 * Devuelve el valor del último nodo de la lista y lo elimina (extrae)
 * @return el valor del último nodo o null si la lista está vacía
 */
T pop();
/**
 * Devuelve el valor del primer nodo de la lista y lo elimina (extrae)
 * @return el valor del primer nodo o null si la lista está vacía
 */
T shift();
```

```
/**
 * Busca el valor pasado como parámetro y devuelve la posición del primer nodo que lo contenga
 * @param v valor a buscar
 * @return posición del nodo que lo contiene o -1 si no existe o la lista está vacía
 */
int contains(T v);
```

```
/**
 * Devuelve el valor del nodo que se encuentre en la posición indicada
 * @param pos posición del nodo
 * @return su valor o null si no es una posición correcto o la lista está vacía
 */
T get(int pos);
```

```java
/**
 * Insertar un nodo con el valor pasado en la posición indicada
 * @param value el valor del nuevo nodo a insertar
 * @param pos la posición donde se desea insertar
 * @return el nuevo tamaño de lista
 */
int put(T value, int pos);

/**
 * Elimina el nodo que se encuentre en la posición indicada
 * @param pos posición del nodo a eliminar
 * @return el tamaño de la lista
 */
int remove(int pos);
/**
 * Eliminar el nodo que contenga el valor indicado
 * @param v valor del nodo a eliminar
 * @return el tamaño de la lista
 */
int removeElement(T v);
```