

# Lección Listas

**Fecha de entrega** 24 de oct de 2023 en 9:40 **Puntos** 100 **Preguntas** 2  
**Disponible** 24 de oct de 2023 en 9:10 - 24 de oct de 2023 en 9:40 30 minutos  
**Límite de tiempo** 30 minutos **Intentos permitidos** Ilimitados

## Instrucciones

En esta lección, usted implementará el método solicitado abajo tanto en la clase **ArrayList** como en la **LinkedList**. Usted es responsable de definir el prototipo de dicho método. Es decir, usted debe decidir cuántos argumentos se requieren (y el tipo de cada uno) así como el tipo de dato que el método debe retornar.

Asuma que usted cuenta con una implementación funcional de las clases **ArrayList** y **LinkedList** que, además de los constructores correspondientes, incluye los siguientes métodos:

```
public int size();  
public boolean isEmpty();  
public void clear();  
public boolean addFirst(E element); // inserta el elemento element al inici  
public boolean addLast(E element); // inserta el elemento element al final  
public E removeFirst(); // remueve el elemento al inicio de la lista  
public E removeLast(); // remueve el elemento al final de la lista  
public boolean add(int index, E element); // inserta element en la posición  
public E remove(int index); // remueve y retorna el elemento en la posición  
public E get(int index); // retorna el elemento ubicado en la posición inde  
public E set(int index, E element); // setea el element en la posición inde  
@Override public String toString();
```

Cuando considere pertinente, incluya comentarios en sus respuestas para proveer detalles adicionales sobre sus decisiones de diseño (p. ej., *// esta implementación asume que...*).



Este examen fue bloqueado en 24 de oct de 2023 en 9:40.

## Historial de intentos



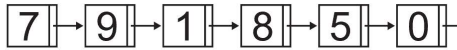
	Intento	Hora	Puntaje
MÁS RECIENTE	<a href="#">Intento 1</a>	30 minutos	50 de 100

Puntaje para este intento: **50** de 100


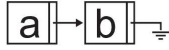

Entregado el 24 de oct de 2023 en 9:40


Este intento tuvo una duración de 30 minutos.

Implemente el método **addAll**, el cual recibe una lista como parámetro. El método añade los elementos de la lista suministrada a la lista que lo invoca. A continuación, se ilustran ejemplos del uso del método **addAll** al ser invocado por dos listas distintas:

Lista que va a invocar a <b>addAll</b> :	
Lista recibida como argumento:	
Lista original después de invocar a <b>addAll</b> :	

Lista que invoca <b>addAll</b> antes de la llamada:	
Lista recibida como argumento:	
Lista original después de invocar a <b>addAll</b> :	



Pregunta 1

30 / 50 pts

Implemente **addAll** en la clase **ArrayList**

Su respuesta:

```
public void addAll(ArrayList<E> arreglo){
    if(!isFull()){
        addCapacity();
    }
    for(int i=0;i<=arreglo.size();i++){
        if(arreglo.get(i)!=null)
            this.addLast(arreglo.get(i));
    }
}

public boolean addLast(E element) {
    if (element == null) {
        return false;
    }
}
```

```
if (isFull()) {  
    addCapacity();  
}  
  
elements[effectiveSize++] = element;  
return true;  
}
```

Mal manejo de operadores de comparacion

## Pregunta 2

20 / 50 pts

Implemente addAll en la clase **LinkedList**

Su respuesta:

```
public void addAll(LinkedList<E> lista){  
  
    last.nextSet(list.header);  
  
}
```

Nunca actualiza la referencia a last de la lista original

Puntaje del examen: **50** de 100