Questions of Advanced-Programing course at Shahid-Beheshti-University

LinkedList Implementation

by

Farbod Fooladi

LinkedList

A list is a data structure used to store elements. One of the types of lists is the famous LinkedList, which you are probably more or less familiar with and will learn more about in the course.

In this exercise, we ask you to implement something like a LinkedList that holds an integer. You can read the documentation for the LinkedList class here.

You must implement the LinkedList class yourself according to the following description.

Your program should include all of the following methods.

```
/***
* adds the specified element to the end of the list
***/
public boolean add(Integer element) ;

/***
* adds the specified element
* at the specified index of your list
***/
public void add(int index , Integer element);
```

```
/***
* append all of the elements
* in the specified linkedlist to the end of this list
***/
public boolean addAll(LinkedList linkedlist);


/**appends all of the elements
* in the specified linkedlist starting at the specified index
***/
public boolean addAll(int index, LinkedList linkedlist);

/***
* inserts a specified element at the beginning of this list
***/
public void addFirst(Integer element);

/***
* appends the specified element at the end of this list
***/
public void addLast(Integer element);

/***
* Removes all of the elements from this list. The list will be empty after this call returns.
***/
public void clear();

/***
```

```
* Returns `true` if this list contains the specified element.
***/
public boolean contains(Integer i);

/***
* Returns the element at the specified position in this list.
***/
public Integer get(int index);

/***
* Returns the index of the first occurrence of the specified element in this list, or -1 if this
list does not contain the element.
***/
public int indexOf(Integer i);

/***
* Retrieves and removes the head (first element) of this list.
***/
public Integer remove();

/***
* Removes the element at the specified position in this list.
* Shifts any subsequent elements to the left (subtracts one from their indices).
*  Returns the element that was removed from the list.
***/
public Integer remove(int index);

/***
* Returns the number of elements in this list.
```

```
***/
public int size();

/***
* Returns an array containing all of the elements in this list in proper sequence.
***/
public Integer[] toArray();

/***
* Returns true if this list contains no elements.
***/
public boolean isEmpty();
```