LinkedList

Overview

1. The underlying Data Structure is a Doubly LinkedList.
2. Insertion order is preserved.
3. Duplicate objects are allowed.
4. Heterogeneous elements are allowed.
5. Null insertion is possible.
6. Implement Serializable and Cloneable Interface. (do not implement RandomAccess Interface).

1. It extends AbstractSequentialList class
2. implements the List and Deque interfaces.

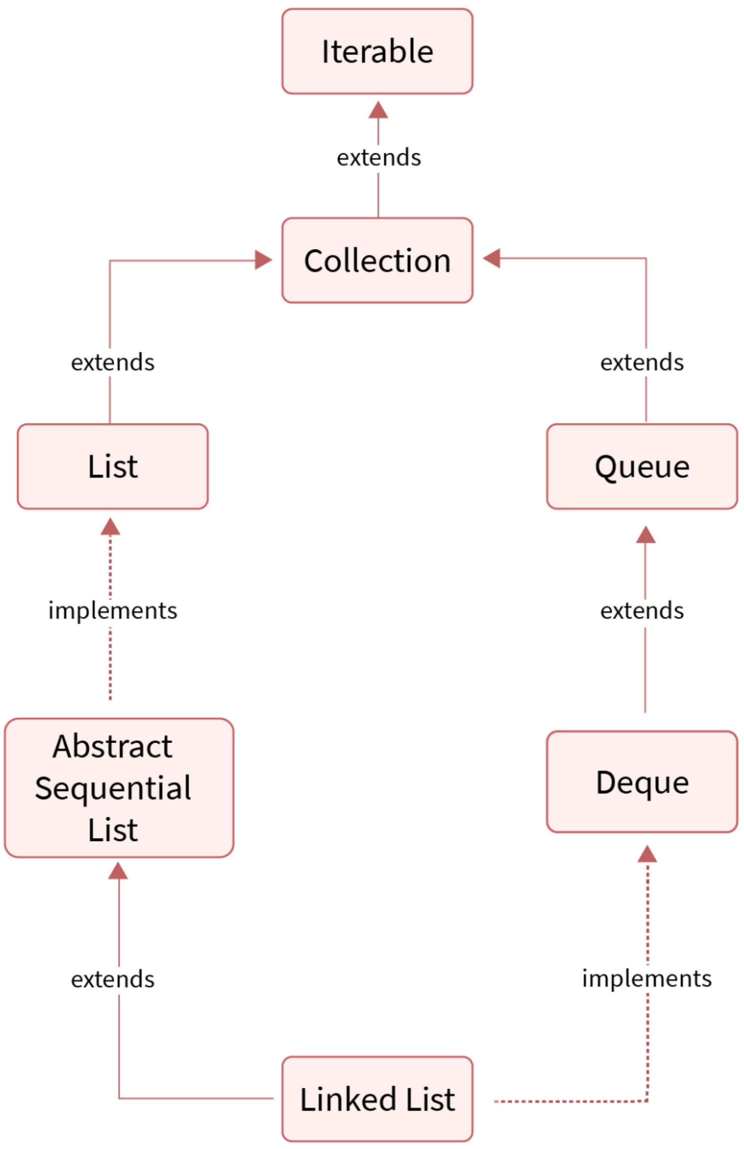
   ```
   public class LinkedList<E>

   extends AbstractSequentialList<E>

   implements List<E>, Deque<E>, Cloneable, java.io.Serializable
   ------------------------------------------------------------
   Deque

   public class Deque<E> implements Queue<E>
   ```

3. Unlike ArrayList, we don't have to specify the size of the list.
4. LinkedList is a dynamic data structure.
5. It automatically changes size when an element is added or removed.
6. Nodes of the linked list are not stored in a contiguous memory location, they are linked through each other with the help of next and previous pointer.

Constructors in the Linked List

1. LinkedList()
   a. It creates empty LinkedList
   b. eg.,
      LinkedList ll = new LinkedList();
2. LinkedList(Collection c)
   a. It is a parameterized constructor.
   b. used to create an ordered list containing elements of the specified collection.
   c. eg.,
      Collection c = new ArrayList<>(Arrays.asList(new String[]{"foo", "bar"}));

Hierarchy of LinkedList



Hierarchy of LinkedList Class in Java

Methods for Linked List in Java

Sr. No.	Method	Description
1	add(int index, E element)	This method is used to **insert the element in a specified position** in the list.
2	add(E element)	This method **appends** the **element to the end** of the list.
3	addAll(int index, Collection c)	This method is used to **insert all the elements** of the **Collection** "c" starting **from** the **specified index**.
4	addAll(Collection c)	This method **appends** all the **elements of** the **Collection** "c" **to the end** of the list.
5	addFirst(E element)	This method **inserts** the desired **element in the beginning** of the list.
6	addLast(E element)	This method **appends** the desired **element at the end** of the list.
7	clear()	This method is used to **clear all the elements** from the list.
8	clone()	This method **returns a shallow copy** of the list.
9	contains(Object x)	This method will **return true if** the specified **element is present** in the list.
10	descendingIterator()	This method **returns an iterator** over the elements **in the reversed sequential** order in deque.
11	element()	This method is used to **retrieve the first element** or head of the list.
12	get(int index)	This method is used to **return** the **element** present **at the specified position**.
13	getFirst()	This method is used to **return the first element** of the list.
14	getLast()	This method is used to **return the last element** of the list.
15	indexOf(Object x)	This method is used to **return the index of** the **first occurrence of** the **specified element**,

		if the element is not present in the list it will return -1.
16	lastIndexOf(Object x)	This method is used to **return the index of** the **last occurrence of** the **specified element**, if the element is not present in the list it will return -1.
17	listIterator(int index)	This method is used to **return the list-iterator of the elements starting from the specified position,** in proper sequence.
18	offer(E element)	This method is used to **add** the specified element **as the tail** of the list.
19	offerFirst(E element)	This method is used to **add** the specified element **in the front** of the list.
20	offerLast(E element)	This method is used to **insert** the specified element **at the end** of the list.
21	peek()	This method **fetches the head** of the list, but doesn't remove it.
22	peekFirst()	This method **fetches the head** of the list or returns NULL if the list is empty.
23	peekLast()	This method **fetches the tail** of the list or returns NULL if the list is empty.
24	poll()	This method is used to **fetch and remove the first element** of the list.
25	pollFirst()	This method **fetches and removes the first element** of the list **or returns null if the list is empty.**
26	pollLast()	This method **fetches and removes the last element** of the list **or returns null if the list is empty.**
27	pop()	sed to **pop an element from the stack** which is repre
28	push(E element)	sed to **push an element onto the stack** which is repr
29	remove()	This method **fetches and removes** the first element of the list.
30	remove(int index)	This method is used to **remove an element at the specified position.**

31	remove(Object o)	This method is used to **remove the first occurrence of the element** if it is present in the list.
32	removeFirst()	This method **returns and removes the first element** of the list.
33	removeFirstOccurrence(Object o)	This method is used to **remove the first occurrence** of the specified element when traversing the list from head to tail.
34	removeLast()	This method **returns and removes the last element** from the list.
35	removeLastOccurrence(Object o)	This method is used to **remove the last occurrence** of the specified element when traversing the list from head to tail.
36	set(int index, E element)	This method is used to **replace the element present at the specified position** with the provided element.
37	size()	This method is used to **return the number of elements** in the list.
38	spliterator()	This method is used to **create a late-binding and fail-fast Spliterator** over the elements in the list.
39	toArray()	This method **returns an array** having all the elements of the list, following the order of head to tail.
40	toArray(T[] a)	This method **returns an array** having all the elements of the list, following the order of head to tail. The **runtime type of the returned array is decided by the typename provided**.
41	toString()	This method is used to **return a string having all the elements of the list**, in the order of head to tail. Each element is separated by a comma and the string is enclosed in square brackets.

Iterating over Elements

1. We can iterate over linked list using
    a. get(int index)
    b. Foreach loop
2. eg., of get(int index)
```
for(int i=0;i<ll.size();i++)
{
        System.out.print(ll.get(i) + " ");
}
```
3. e.g., of foreach loop
```
for(String s:ll)
{
        System.out.print(s+" ");
}
```

LinkedList as Queue

1. Linked List **implements Queue interface.**
2. Since the **queue** follows the first in first out (**FIFO**) **order**, hence we will be accessing the first element of the list and will be inserting a new element at the end of the list.
3. **LinkedList class contains** the **methods** for the **queue interface**.
4. Methods of Queue interfaces are:
    - boolean add(E e) - **Inserts** the specified **element E** into this queue.

    - boolean offer(E e) - **Inserts** the specified **element E** into this queue.
      When using a capacity-restricted queue, the offer(E e) method is generally preferable.

    - E remove() - **Retrieves and removes the head of** this **queue**.
      remove() method throws an exception if this queue is empty.

    - E poll() - Retrieves and removes the head of this queue.
      poll() method returns null if the queue is empty.

    - E element() - **Retrieve** the **head** of the Queue **but do not remove** the **head**.
      If Queue is empty, it throws an Exception “NoSuchElementException”.

    - E peek() - **Retrieve** the **head** of the Queue **but do not remove** the **head**.
      Return null if the Queue is empty.

LinkedList as Deque

1. **LinkedList class implements Deque interface**.
2. Hence **LinkedList** class **contains methods of Deque**.
3. In Double Ended Queue, we can **insert and access elements from the both ends** of the list i.e., start and end.
4. **Methods from the Deque interface** implemented in LinkedList class
    a. void addFirst(E e)
    b. void addLast(E e)
    c. boolean offerFirst(E e)
    d. boolean offerLast(E e)
    e. E removeFirst()
    f. E removeLast()
    g. E pollFirst()
    h. E pollLast()
    i. E getFirst()
    j. E getLast()
    k. E peekFirst()
    l. E peekLast()
    m. boolean removeFirstOccurrence(Object o)
    n. boolean removeLastOccurrence(Object o)
    o. boolean add(E e)
    p. boolean offer(E e)
    q. E remove()
    r. E poll()
    s. E element()
    t. E peek()
    u. boolean addAll(Collection<? extends E> c)
    v. void push(E e)
    w. E pop()
    x. boolean remove(Object o)
    y. boolean contains(Object o)
    z. int size()
    aa. Iterator<E> iterator()
    bb. Iterator<E> descendingIterator()