LinkedList and its methods:

- Java LinkedList class uses a doubly linked list to store the elements.
- It provides a linked-list data structure.
- It implements List and Deque interfaces.

The important points about Java LinkedList are:

- Java LinkedList class can contain duplicate elements.
- Java LinkedList class maintains insertion order.
- Java LinkedList class is non synchronized.
- In Java LinkedList class, manipulation is fast because no shifting needs to occur.
- Java LinkedList class can be used as a list, stack or queue.
- In the case of a doubly linked list, we can add or remove elements from both sides.

```
import java.util.Iterator;
import java.util.LinkedList;
public class A4
{
   public static void main(String[] args)
   {
      //LinkedList constructor (1)
      // It is used to construct an empty list.
      LinkedList<Double> L1 = new LinkedList<>();

      // 1. boolean add(E e)
      //It is used to append the specified element to the end of a list.

      // L1.add(10); // C.E.: Element must be Double
      L1.add(10.0);
      L1.add(20.0);
      L1.add(30.0);
      L1.add(40.0);
      L1.add(50.0);
      System.out.println("LinkedList-L1::"+L1);
      //LinkedList-L1::[10.0, 20.0, 30.0, 40.0, 50.0]

      //LinkedList constructor (2)
```

```
// It is used to construct a list containing the elements of the
// specified collection, in the order, they are returned by the
// collection's iterator.

LinkedList<Double> L2 = new LinkedList<>(L1);
System.out.println("LinkedList L2 is (Copy of L1): "+L2);
//LinkedList L2 is (Copy of L1): [10.0, 20.0, 30.0, 40, 50.0]

//LinkedList<Double> L3 = new LinkedList<>(10);
//C.E:Cannot infer type arguments for LinkedList<>
// C.E.: No concept of initial capacity

// 2. void add(int index, E element)
//It is used to insert the specified element at the specified
//position index in a list.

LinkedList<Double> L3 = new LinkedList<>();
//L3.add(2,16.0);
//RE:IndexOutOfBoundsException: Index: 2, Size: 1
//Always start from index 0

System.out.println("LinkedList-L1::"+L1);
//LinkedList-L1::[10.0, 20.0, 30.0, 40.0, 50.0]
L1.add(2,15.0);
// L1.add(7,15.0); // R.E.: IndexOutOfBoundsException
L1.add(6,12.0);
// You can add up to +1 position of given Linked List
System.out.println("LinkedList-L1::"+L1);
//LinkedList-L1::[10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0]

//3. boolean addAll(Collection<? extends E> c)
//It is used to append all of the elements in the specified
//collection to the end of this list,in the order that they
//are returned by the specified collection's iterator.

System.out.println("LinkedList-L3::"+L3);//LinkedList-L3::[]
System.out.println(L3.addAll(L1));//true
System.out.println("LinkedList-L3::"+L3);
//LinkedList-L3::[10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0]

//4. boolean addAll(int index, Collection<? extends E> c)
//It is used to append all the elements in the specified
//collection, starting at the specified position of the list.
```

```
System.out.println("LinkedList L2 is: "+L2);
//LinkedList L2 is: [10.0, 20.0, 30.0, 40.0, 50.0]
System.out.println("LinkedList L1 is: "+L1);
//LinkedList L1 is: [10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0]
System.out.println(L2.addAll(2,L1));//true
System.out.println("LinkedList L2 is: "+L2);
//LinkedList L2 is: [10.0, 20.0, 10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0, 30.0,
40.0, 50.0]

// 5. void addFirst(E e)
  //It is used to insert the given element at the beginning of a list.

L1.addFirst(121.0);

// 6. void addLast(E e)
  //It is used to append the given element to the end of a list.

L1.addLast(225.0);
System.out.println("LinkedList L1 is: "+L1);
//LinkedList L1 is: [121.0, 10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0, 225.0]

// 7. void clear()
//It is used to remove all the elements from a list.

L2.clear();
System.out.println("LinkedList L2 is: "+L2);//LinkedList L2 is: []

// 8. boolean contains(Object o)
//It is used to return true if a list contains a specified element.

System.out.println("L1 contains 12.0 ? "+L1.contains(12.0));
//L1 contains 12.0 ? true
System.out.println("L1 contains 331.0 ? "+L1.contains(331.0));
//L1 contains 331.0 ? false

// 9. E get(int index)
//It is used to return the element at the specified position in a list.

System.out.println("LinkedList L1 is: "+L1);
//LinkedList L1 is: [121.0, 10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0, 225.0]
System.out.println("Element at Index 6 is: "+L1.get(6));
//Element at Index 6 is: 50.0
```

```
        //System.out.println("Element at Index 11 is: "+L1.get(11));
        //R.E:IndexOutOfBoundsException

        // 10. E getFirst()
        //It is used to return the first element in a list.

        System.out.println("First element of L1 is: "+L1.getFirst());
        //First element of L1 is: 121.0

        // 11. E getLast()
        //It is used to return the last element in a list.

        System.out.println("Last element of L1 is: "+L1.getLast());
        //Last element of L1 is: 225.0

        // 12. E removeFirst()
        //It is used to retrieve and remove the first element of a list.
        System.out.println("LinkedList L1 is: "+L1);
        //LinkedList L1 is: [121.0, 10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0,
        225.0]
        System.out.println(L1.removeFirst());//121.0

        //13. E removeLast()
        //It is used to retrieve and remove the Last element of a list.

        System.out.println(L1.removeLast());//225.0
        System.out.println("LinkedList L1 is: "+L1);
        //LinkedList L1 is: [10.0, 20.0, 15.0, 30.0, 40.0, 50.0, 12.0]

        //Also explain remove() and remove(int Index)

        // 14. Use of iterator to print elements of List

        Iterator itr = L1.iterator();
        while(itr.hasNext())
        {
           System.out.println(itr.next());
        }
    }
}
```