Java List

List in Java provides the facility to maintain the *ordered collection*. It contains the index-based methods to insert, update, delete and search the elements. It can have the duplicate elements also. We can also store the null elements in the list.

The List interface is found in the java.util package and inherits the Collection interface. It is a factory of ListIterator interface. Through the ListIterator, we can iterate the list in forward and backward directions. The implementation classes of List interface are ArrayList, LinkedList, Stack and Vector. The ArrayList and LinkedList are widely used in Java programming. The Vector class is deprecated since Java 5.

List Interface declaration

```
public interface List<E> extends Collection<E>
```

Java List Methods

Method	Description
void add(int index, E element)	It is used to insert the specified element at the specified position in a list.
boolean add(E e)	It is used to append the specified element at the end of a list.
boolean addAll(Collection<? extends E> c)	It is used to append all of the elements in

	the specified collection to the end of a list.
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.
void clear()	It is used to remove all of the elements from this list.
boolean equals(Object o)	It is used to compare the specified object with the elements of a list.
int hashcode()	It is used to return the hash code value for a list.
E get(int index)	It is used to fetch the element from the particular position of the list.
boolean isEmpty()	It returns true if the list is empty, otherwise false.
int lastIndexOf(Object o)	It is used to return the index in this list of the last occurrence of the specified element, or -1 if the list does not contain this element.
Object[] toArray()	It is used to return

	an array containing all of the elements in this list in the correct order.
<T> T[] toArray(T[] a)	It is used to return an array containing all of the elements in this list in the correct order.
boolean contains(Object o)	It returns true if the list contains the specified element
boolean containsAll(Collection<?> c)	It returns true if the list contains all the specified element
int indexOf(Object o)	It is used to return the index in this list of the first occurrence of the specified element, or -1 if the List does not contain this element.
E remove(int index)	It is used to remove the element present at the specified position in the list.
boolean remove(Object o)	It is used to remove the first occurrence of the specified element.
boolean removeAll(Collection<?> c)	It is used to remove all the elements from the list.

void replaceAll(UnaryOperator<E> operator)	It is used to replace all the elements from the list with the specified element.	
void retainAll(Collection<?> c)	It is used to retain all the elements in the list that are present in the specified collection.	
E set(int index, E element)	It is used to replace the specified element in the list, present at the specified position.	
void sort(Comparator<? super E> c)	It is used to sort the elements of the list on the basis of specified comparator.	
Spliterator<E> spliterator()	It is used to create spliterator over the elements in a list.	
List<E> subList(int fromIndex, int toIndex)	It is used to fetch all the elements lies within the given range.	
int size()	It is used to return the number of elements present in the list.	

Java List vs ArrayList

List is an interface whereas ArrayList is the implementation class of List.

How to create List

The ArrayList and LinkedList classes provide the implementation of List interface. Let's see the examples to create the List:

```
//Creating a List of type String using ArrayList
List<String> list=new ArrayList<String>();

//Creating a List of type Integer using ArrayList
List<Integer> list=new ArrayList<Integer>();

//Creating a List of type Book using ArrayList
List<Book> list=new ArrayList<Book>();

//Creating a List of type String using LinkedList
List<String> list=new LinkedList<String>();
```

In short, you can create the List of any type. The ArrayList<T> and LinkedList<T> classes are used to specify the type. Here, T denotes the type.

Java List Example

Let's see a simple example of List where we are using the ArrayList class as the implementation.

```
import java.util.*;
public class ListExample1{
public static void main(String args[]){
```

```
//Creating a List
List<String> list=new ArrayList<String>();
//Adding elements in the List
list.add("Mango");
list.add("Apple");
list.add("Banana");
list.add("Grapes");
//Iterating the List element using for-each loop
for(String fruit:list)
 System.out.println(fruit);

}
}
```

Test it Now

Output:

```
Mango
Apple
Banana
Grapes
```

How to convert Array to List

We can convert the Array to List by traversing the array and adding the element in list one by one using list.add() method. Let's see a simple example to convert array elements into List.

```
import java.util.*;
public class ArrayToListExample{
public static void main(String args[]){
//Creating Array
String[] array={"Java","Python","PHP","C++"};
System.out.println("Printing Array: "+Arrays.toString(array));
//Converting Array to List
List<String> list=new ArrayList<String>();
for(String lang:array){
list.add(lang);
}
System.out.println("Printing List: "+list);


}
}
```

Test it Now

Output:

```
Printing Array: [Java, Python, PHP, C++]
Printing List: [Java, Python, PHP, C++]
```

How to convert List to Array

We can convert the List to Array by calling the list.toArray() method. Let's see a simple example to convert list elements into array.

```
import java.util.*;
public class ListToArrayExample{
public static void main(String args[]){
 List<String> fruitList = new ArrayList<>();
 fruitList.add("Mango");
 fruitList.add("Banana");
 fruitList.add("Apple");
 fruitList.add("Strawberry");
 //Converting ArrayList to Array
 String[] array = fruitList.toArray(new String[fruitList.size()]);
 System.out.println("Printing Array: "+Arrays.toString(array))
 System.out.println("Printing List: "+fruitList);
}
}
```

Test it Now

Output:

```
Printing Array: [Mango, Banana, Apple, Strawberry]
Printing List: [Mango, Banana, Apple, Strawberry]
```

Get and Set Element in List

The *get() method* returns the element at the given index, whereas the *set() method* changes or replaces the element.

```
import java.util.*;
public class ListExample2{
 public static void main(String args[]){
 //Creating a List
 List<String> list=new ArrayList<String>();
 //Adding elements in the List
 list.add("Mango");
 list.add("Apple");
 list.add("Banana");
 list.add("Grapes");
 //accessing the element
 System.out.println("Returning element: "+list.get(1));//it will return the 2nd element, because index
 //changing the element
 list.set(1,"Dates");
 //Iterating the List element using for-each loop
 for(String fruit:list)
  System.out.println(fruit);


 }
}
```

Test it Now

Output:

```
Returning element: Apple
Mango
Dates
Banana
Grapes
```

How to Sort List

There are various ways to sort the List, here we are going to use Collections.sort() method to sort the list element. The *java.util* package provides a utility class **Collections** which has the static method sort(). Using the **Collections.sort()** method, we can easily sort any List.

```
import java.util.*;
class SortArrayList{
 public static void main(String args[]){
  //Creating a list of fruits
  List<String> list1=new ArrayList<String>();
  list1.add("Mango");
  list1.add("Apple");
  list1.add("Banana");
  list1.add("Grapes");
  //Sorting the list
  Collections.sort(list1);
   //Traversing list through the for-each loop
  for(String fruit:list1)
    System.out.println(fruit);

 System.out.println("Sorting numbers...");
  //Creating a list of numbers
  List<Integer> list2=new ArrayList<Integer>();
  list2.add(21);
  list2.add(11);
  list2.add(51);
```

```
    list2.add(1);
    //Sorting the list
    Collections.sort(list2);
     //Traversing list through the for-each loop
    for(Integer number:list2)
      System.out.println(number);
  }

 }
```

Output:

```
Apple
Banana
Grapes
Mango
Sorting numbers...
1
11
21
51
```

Java ListIterator Interface

ListIterator Interface is used to traverse the element in a backward and forward direction.

ListIterator Interface declaration

```
public interface ListIterator<E> extends Iterator<E>
```

Methods of Java ListIterator Interface:

Method	Description
void add(E e)	This method inserts the specified element into the list.
boolean	This method returns true if the list

hasNext()	iterator has more elements while traversing the list in the forward direction.
E next()	This method returns the next element in the list and advances the cursor position.
int nextIndex()	This method returns the index of the element that would be returned by a subsequent call to next()
boolean hasPrevious()	This method returns true if this list iterator has more elements while traversing the list in the reverse direction.
E previous()	This method returns the previous element in the list and moves the cursor position backward.
E previousIndex()	This method returns the index of the element that would be returned by a subsequent call to previous().
void remove()	This method removes the last element from the list that was returned by next() or previous() methods
void set(E e)	This method replaces the last element returned by next() or previous() methods with the specified element.

Example of ListIterator Interface

```
import java.util.*;
public class ListIteratorExample1{
public static void main(String args[]){
List<String> al=new ArrayList<String>();
        al.add("Amit");
        al.add("Vijay");
        al.add("Kumar");
        al.add(1,"Sachin");
        ListIterator<String> itr=al.listIterator();
        System.out.println("Traversing elements in forward direction");
        while(itr.hasNext()){

        System.out.println("index:"+itr.nextIndex()+" value:"+it

        }
        System.out.println("Traversing elements in backward d

        while(itr.hasPrevious()){

        System.out.println("index:"+itr.previousIndex()+" value:"+itr.previous());
        }
}
}
```

Output:

Narendra
PLEAS
For Suffolk Commonwealth's Attorney
Fair.
Responsive.
Unwavering.
PAID FOR AND AUTHORIZED BY: PLEAS FOR SUFFOLK VA

```
Traversing elements in forward direction
index:0 value:Amit
index:1 value:Sachin
index:2 value:Vijay
index:3 value:Kumar
Traversing elements in backward direction
index:3 value:Kumar
index:2 value:Vijay
```

```
index:1 value:Sachin
index:0 value:Amit
```

Example of List: Book

Let's see an example of List where we are adding the Books.

```
import java.util.*;
class Book {
int id;
String name,author,publisher;
int quantity;
public Book(int id, String name, String author, String publisher, int quantity) {
    this.id = id;
    this.name = name;
    this.author = author;
    this.publisher = publisher;
    this.quantity = quantity;
}
}
public class ListExample5 {
public static void main(String[] args) {
    //Creating list of Books
    List<Book> list=new ArrayList<Book>();
    //Creating Books
    Book b1=new Book(101,"Let us C","Yashwant Kanetkar",'
    Book b2=new Book(102,"Data Communications and Networking","Forouzan","Mc Graw Hill",4);
    Book b3=new Book(103,"Operating System","Galvin","W
    //Adding Books to list
    list.add(b1);
    list.add(b2);
    list.add(b3);
    //Traversing list
    for(Book b:list){
```

```
    System.out.println(b.id+" "+b.name+" "+b.author+" "+b.publisher+" "+b.quantity);
    }
}
}
```

Test it Now

Output:

```
101 Let us C Yashwant Kanetkar BPB 8
102 Data Communications and Networking Forouzan Mc Graw Hill 4
103 Operating System Galvin Wiley 6
```

Related Topics

ArrayList in Java

LinkedList in Java

Difference between ArrayList and LinkedList

Difference between Array and ArrayList

When to use ArrayList and LinkedList in Java

Difference between ArrayList and Vector

How to Compare Two ArrayList in Java

How to reverse ArrayList in Java

When to use ArrayList and LinkedList in Java

How to make ArrayList Read Only

Difference between length of array and size() of ArrayList in Java

How to Synchronize ArrayList in Java

How to convert ArrayList to Array and Array to ArrayList in java

Array vs ArrayList in Java

How to Sort Java ArrayList in Descending Order

How to remove duplicates from ArrayList in Java

← Prev

Next →

Splunk	SPSS	Swagger	Transact-SQL
Tumblr tutorial Tumblr	React tutorial ReactJS	Regex tutorial Regex	Reinforcement learning tutorial Reinforcement Learning
R Programming tutorial R Programming	RxJS tutorial RxJS	React Native tutorial React Native	Python Design Patterns Python Design Patterns
Python Pillow tutorial Python Pillow	Python Turtle tutorial Python Turtle	Keras tutorial Keras	

TIM ANDERSON'S AGENDA:

Overturn Roe v. Wade

End Safe & Legal Abortions

Gut Reproductive Rights

STOP HIM.

PAID FOR BY GUY FOR DELEGATE.

Preparation

Aptitude Aptitude	Logical Reasoning Reasoning	Verbal Ability Verbal Ability	Interview Questions Interview Questions
Company Interview Questions			

Company Questions

Trending Technologies

Artificial Intelligence Tutorial

Artificial Intelligence

AWS Tutorial

AWS

Selenium tutorial

Selenium

Cloud Computing tutorial

Cloud Computing

Hadoop tutorial

Hadoop

ReactJS Tutorial

ReactJS

Data Science Tutorial

Data Science

Angular 7 Tutorial

Angular 7

Blockchain Tutorial

Blockchain

Git Tutorial

Git

Machine Learning Tutorial

Machine Learning

DevOps Tutorial

DevOps

TIM ANDERSON'S AGENDA:

☒ Overturn Roe v. Wade

☒ End Safe & Legal Abortions

☒ Gut Reproductive Rights

STOP HIM.

PAID FOR BY GUY FOR DELEGATE.

B.Tech / MCA

DBMS tutorial

DBMS

Data Structures tutorial

Data Structures

DAA tutorial

DAA

Operating System tutorial

Operating System

Computer Network tutorial

Computer Network

Compiler Design tutorial

Compiler Design

Computer Organization and Architecture

Computer Organization

Discrete Mathematics Tutorial

Discrete Mathematics

Ethical Hacking Tutorial

Ethical Hacking

Computer Graphics Tutorial

Computer Graphics

Software Engineering Tutorial

Software Engineering

html tutorial

Web Technology

Cyber Security tutorial

Cyber Security

Automata Tutorial

Automata

C Language tutorial

C Programming

C++ tutorial

C++

Java tutorial

Java

.Net Framework tutorial

.Net

Python tutorial

Python

List of Programs

Programs

Control Systems tutorial

Control System

Data Mining Tutorial

Data Mining

Data Warehouse Tutorial

Data Warehouse