**Java Iterator Examples**

All of the collection classes provides iterator() method to iterate through the collection. The iterator() method returns the Iterator object through which you can access the collection elements in an order.

Enumeration also does the same purpose. The difference between Iterator and Enumerations is: Iterators allow the caller to remove elements from the underlying collection during the iteration with well-defined semantics.

# How to iterate through collection objects?

You can iterate through any collection object by using Iterator object. It provides two methods to iterate. The hasNext() method returns true if the iteration has more elements. The next() method returns the next element in the iteration. Below example shows how to iterate through an ArrayList.

**package com.java2novice.iterator;**

**import java.util.ArrayList;**

**import java.util.Iterator;**

**import java.util.List;**

**public class MyCollectionIterator {**

**public static void main(String a[]){**

**List<String> myList = new ArrayList<String>();**

**myList.add("Java");**

**myList.add("Unix");**

**myList.add("Oracle");**

**myList.add("C++");**

**myList.add("Perl");**

**Iterator<String> itr = myList.iterator();**

**while(itr.hasNext()){**

**System.out.println(itr.next());**

**}}}**

# How to remove an element from collection using Iterator object?

 To remove an element from collection object using Iterator object. The remove() method removes from the underlying collection the last element returned by the iterator.

**package com.java2novice.iterator;**

**import java.util.ArrayList;**

**import java.util.Iterator;**

**import java.util.List;**

**public class MyItrRemoveElement {**

**public static void main(String a[]){**

**String removeElem = "Perl";**

**List<String> myList = new ArrayList<String>();**

**myList.add("Java");**

**myList.add("Unix");**

**myList.add("Oracle");**

**myList.add("C++");**

**myList.add("Perl");**

**System.out.println("Before remove:");**

**System.out.println(myList);**

**Iterator<String> itr = myList.iterator();**

**while(itr.hasNext()){**

**if(removeElem.equals(itr.next())){**

**itr.remove();**

**}**

**}**

**System.out.println("After remove:");**

**System.out.println(myList);**

**}}**

Before remove:

[Java, Unix, Oracle, C++, Perl]

After remove:

[Java, Unix, Oracle, C++]

# Java ListIterator Sample Code

Using ListIterator, we can iterate all elements of a list in either direction. You can access next element by calling next() method, and also you can access previous element by calling previous() method on the list.

**package com.myjava.listiterator;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.ListIterator;**

**public class MyListIterator {**

**public static void main(String a[]){**

**List<Integer> li = new ArrayList<Integer>();**

**ListIterator<Integer> litr = null;**

**li.add(23);**

**li.add(98);**

**li.add(29);**

**li.add(71);**

**li.add(5);**

**litr=li.listIterator();**

**System.out.println("Elements in forward directiton");**

**while(litr.hasNext()){**

**System.out.println(litr.next());**

**}**

**System.out.println("Elements in backward directiton");**

**while(litr.hasPrevious()){**

**System.out.println(litr.previous());**

**}}}**

Elements in forward directiton

23

98

29

71

5

Elements in backward directiton

5

71

29

98

23

# Java Enumeration Sample Code

A class that implements Enumeration interface provides access to series of elements one at a time. You need to call nextElement method to get next element in the series. Also hasMoreElements() method gives you status about the availability of next element in the series.

**package com.myjava.Enumeration;**

**import java.util.Enumeration;**

**import java.util.Vector;**

**public class MyEnumeration {**

**public static void main(String a[]){**

**Vector<String> lang = new Vector<String>();**

**Enumeration<String> en = null;**

**lang.add("JAVA");**

**lang.add("JSP");**

**lang.add("SERVLET");**

**lang.add("EJB");**

**lang.add("PHP");**

**lang.add("PERL");**

**en = lang.elements();**

**while(en.hasMoreElements()){**

**System.out.println(en.nextElement());**

**}}}**

JAVA

JSP

SERVLET

EJB

PHP

PERL

# Java Vector Examples

The Vector class implements a growable array of objects. Like an array, it contains components that can be accessed using an integer index. However, the size of a Vector can grow or shrink as needed to accommodate adding and removing items after the Vector has been created.

# Basic Vector Operations.

Below example shows how to create vector object, adding elements to it, getting elements by specifying index, getting elements index, getting first element, getting last element, and is vector is empty or not.

**package com.java2novice.vector;**

**import java.util.Vector;**

**public class BasicVectorOperations {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**System.out.println(vct);**

**//adding element at specified index**

**vct.add(2,"Random");**

**System.out.println(vct);**

**//getting elements by index**

**System.out.println("Element at index 3 is: "+vct.get(3));**

**//getting first element**

**System.out.println("The first element of this vector is: "+vct.firstElement());**

**//getting last element**

**System.out.println("The last element of this vector is: "+vct.lastElement());**

**//how to check vector is empty or not**

**System.out.println("Is this vector empty? "+vct.isEmpty());**

**}**

**}**

[First, Second, Third]

[First, Second, Random, Third]

Element at index 3 is: Third

The first element of this vector is: First

The last element of this vector is: Third

Is this vector empty? false

---==============

# How to read all elements in vector by using iterator?

To iterate through vector using iterator object. You can get iterator object by calling iterator() method.

**package com.java2novice.vector;**

**import java.util.Iterator;**

**import java.util.Vector;**

**public class VectorIterator {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**Iterator<String> itr = vct.iterator();**

**while(itr.hasNext()){**

**System.out.println(itr.next());**

        }}}

First

Second

Third

Random

# How to read all elements in vector by using Enumeration?

# To iterate through vector using Enumeration object. You can get Enumeration object by calling elements() method.

**package com.java2novice.vector;**

**import java.util.Enumeration;**

**import java.util.Vector;**

**public class VectorEnnumaratio {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**Enumeration<String> enm = vct.elements();**

**while(enm.hasMoreElements()){**

**System.out.println(enm.nextElement());**

**}**

**}**

**}**

First

Second

Third

Random

# How to copy or clone a vector?

To create exactly same object as available vector object. You can use clone() method to create similar copy of vector.

**package com.java2novice.vector;**

**import java.util.Vector;**

**public class MyVectorClone {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**System.out.println("Actual vector:"+vct);**

**Vector<String> copy = (Vector<String>) vct.clone();**

**System.out.println("Cloned vector:"+copy);**

**}**

**}**

Actual vector:[First, Second, Third, Random]

Cloned vector:[First, Second, Third, Random]

# How to add all elements of a list to vector?

# To copy or create a vector with another collection object. In the code we have created an ArrayList and by using addAll() method, we can copy another collection object.

**package com.java2novice.vector;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Vector;**

**public class MyVectorNewCollection {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**System.out.println("Actual vector:"+vct);**

**List<String> list = new ArrayList<String>();**

**list.add("one");**

**list.add("two");**

**vct.addAll(list);**

**System.out.println("After Copy: "+vct);**

**}**

**}**

Actual vector:[First, Second, Third, Random]

After Copy: [First, Second, Third, Random, one, two]

# How to delete all elements from my vector?

To remove all elements from vector at once. You can call clear() method to delete all elements of a vector at one call.

package com.java2novice.vector;

import java.util.Vector;

**public class ClearMyVector {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**//adding elements to the end**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**System.out.println("Actual vector:"+vct);**

**vct.clear();**

**System.out.println("After clear vector:"+vct);**

**}**

**}**

Actual vector:[First, Second, Third, Random]

After clear vector:[]

# How to find does vector contains all list elements or not?

# To find whether a vector contains list of objects or not. The method containsAll() helps us to find the match.

**package com.java2novice.vector;**

**import java.util.ArrayList;**

**import java.util.List;**

**import java.util.Vector;**

**public class MyElementCheck {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**List<String> list = new ArrayList<String>();**

**list.add("Second");**

**list.add("Random");**

**System.out.println("Does vector contains all list elements?: "+vct.containsAll(list));**

**list.add("one");**

**System.out.println("Does vector contains all list elements?: "+vct.containsAll(list));**

**}**

**}**

Does vector contains all list elements?: true

Does vector contains all list elements?: false

# How to copy vector to array?

You can copy all elements of a vector object to an array. By passing an array object to copyInto() method, you can copy content of a vector object to an array.

**package com.java2novice.vector;**

**import java.util.Vector;**

**public class MyVectorArrayCopy {**

**public static void main(String a[]){**

**Vector<String> vct = new Vector<String>();**

**vct.add("First");**

**vct.add("Second");**

**vct.add("Third");**

**vct.add("Random");**

**System.out.println("Actual vector:"+vct);**

**String[] copyArr = new String[vct.size()];**

**vct.copyInto(copyArr);**

**System.out.println("Copied Array content:");**

**for(String str:copyArr){**

**System.out.println(str);**

**}**

**}**

**}**

Actual vector:[First, Second, Third, Random]

Copied Array content:

First

Second

Third

Random

# ==

# How to get sub list from vector?

# You can copy a range of vector content. The method subList() helps us to get sub range from the given vector object.

package com.java2novice.vector;

import java.util.List;

import java.util.Vector;

public class MyVectorSubRange {

    public static void main(String a[]){

        Vector<String> vct = new Vector<String>();

        //adding elements to the end

        vct.add("First");

        vct.add("Second");

        vct.add("Third");

        vct.add("Random");

        vct.add("Click");

        System.out.println("Actual vector:"+vct);

        List<String> list = vct.subList(2, 4);

        System.out.println("Sub List: "+list);

    }

}

Actual vector:[First, Second, Third, Random, Click]

Sub List: [Third, Random]

# Java ArrayList Examples

# ArrayList is a resizable-array implementation of the List interface. Implements all optional list operations, and permits all elements, including null. In addition to implementing the List interface, this class provides methods to manipulate the size of the array that is used internally to store the list. (This class is roughly equivalent to Vector, except that it is unsynchronized.)

# Basic ArrayList Operations.

# Here we can see example for basic ArrayList operations like creating object for ArrayList, adding objects into ArrayList, accessing objects based on index, searching an object in ArrayList whether it is listed under this instance or not, adding elements at specific index, checking whether the ArrayList is empty or not, getting object index, and finally size of the ArrayList.

**package com.java2novice.arraylist;**

**import java.util.ArrayList;**

**public class MyBasicArrayList {**

**public static void main(String[] a){**

**ArrayList<String> al = new ArrayList<String>();**

**//add elements to the ArrayList**

**al.add("JAVA");**

**al.add("C++");**

**al.add("PERL");**

**al.add("PHP");**

**System.out.println(al);**

**//get elements by index**

**System.out.println("Element at index 1: "+al.get(1));**

**System.out.println("Does list contains JAVA? "+al.contains("JAVA"));**

**//add elements at a specific index**

**al.add(2,"PLAY");**

**System.out.println(al);**

**System.out.println("Is arraylist empty? "+al.isEmpty());**

**System.out.println("Index of PERL is "+al.indexOf("PERL"));**

**System.out.println("Size of the arraylist is: "+al.size());**

**}**

**}**

[JAVA, C++, PERL, PHP]

Element at index 1: C++

Does list contains JAVA? true

[JAVA, C++, PLAY, PERL, PHP]

Is arraylist empty? false

Index of PERL is 3

Size of the arraylist is: 5

# How to read all elements in ArrayList by using iterator?

Example for reading all elements from ArrayList by using Iterator. Also you can iterate through the ArrayList based on index too.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.Iterator;

public class ArrayListIterator {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        Iterator<String> itr = arrl.iterator();

        while(itr.hasNext()){

            System.out.println(itr.next());

        }

    }

}

First

Second

Third

Random

# How to copy or clone a ArrayList?

For creating duplicate object of an ArrayList instance. we can done this by using clone() function.

**package com.java2novice.arraylist;**

**import java.util.ArrayList;**

**public class MyArrayListClone {**

**public static void main(String a[]){**

**ArrayList<String> arrl = new ArrayList<String>();**

**//adding elements to the end**

**arrl.add("First");**

**arrl.add("Second");**

**arrl.add("Third");**

**arrl.add("Random");**

**System.out.println("Actual ArrayList:"+arrl);**

**ArrayList<String> copy = (ArrayList<String>) arrl.clone();**

**System.out.println("Cloned ArrayList:"+copy);**

**}**

**}**

Actual ArrayList:[First, Second, Third, Random]

Cloned ArrayList:[First, Second, Third, Random]

# How to add all elements of a list to ArrayList?

# Example for copying another collection instance objects to existing ArrayList.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.List;

public class MyArrayListNewCollection {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual ArrayList:"+arrl);

        List<String> list = new ArrayList<String>();

        list.add("one");

        list.add("two");

        arrl.addAll(list);

        System.out.println("After Copy: "+arrl);

    }

}

|  |
| --- |
| **Output:** |
| Actual ArrayList:[First, Second, Third, Random]  After Copy: [First, Second, Third, Random, one, two] |

# How to delete all elements from my ArrayList?

# for deleting all objects from ArrayList at one method call. We can do this by calling clear() method on ArrayList, it will delete all objects.

package com.java2novice.arraylist;

import java.util.ArrayList;

public class ClearMyArrayList {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual ArrayList:"+arrl);

        arrl.clear();

        System.out.println("After clear ArrayList:"+arrl);

    }

}

Actual ArrayList:[First, Second, Third, Random]

After clear ArrayList:[]

# How to find does ArrayList contains all list elements or not?

# For finding whether the instance of an ArrayList contains all objects of another Collection instance. Here we are checking with another List instance.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.List;

public class MyElementCheck {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        List<String> list = new ArrayList<String>();

        list.add("Second");

        list.add("Random");

        System.out.println("Does ArrayList contains all list elements?: "

                    +arrl.containsAll(list));

        list.add("one");

        System.out.println("Does ArrayList contains all list elements?: "

                    +arrl.containsAll(list));

    }

}

Does ArrayList contains all list elements?: true

Does ArrayList contains all list elements?: false

# How to copy ArrayList to array?

# For copying all content of ArrayList to an array. You can get this done by calling toArray() method.

package com.java2novice.arraylist;

import java.util.ArrayList;

public class MyArrayListArray {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual ArrayList:"+arrl);

        String[] strArr = new String[arrl.size()];

        arrl.toArray(strArr);

        System.out.println("Created Array content:");

        for(String str:strArr){

            System.out.println(str);

        }

    }}

Actual ArrayList:[First, Second, Third, Random]

Created Array content:

First

Second

Third

Random

# How to get sub list from ArrayList?

For getting ArrayList content based on range of index. By specifing the start and end index of the range, you can get the sub-list.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.List;

public class MyArrayListSubRange {

    public static void main(String a[]){

        ArrayList<String> arrl = new ArrayList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        arrl.add("Click");

        System.out.println("Actual ArrayList:"+arrl);

        List<String> list = arrl.subList(2, 4);

        System.out.println("Sub List: "+list);

    }

}

Actual ArrayList:[First, Second, Third, Random, Click]

Sub List: [Third, Random]

# How to sort ArrayList using Comparator?

To sort an ArrayList using Comparator. The ArrayList contains user defined objects. By using Collections.sort() method you can sort the ArrayList. You have to pass Comparator object which contains your sort logic. The example sorts the Empl objects based on highest salary.

ackage com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.Collections;

import java.util.Comparator;

import java.util.List;

public class MyArrayListSort {

    public static void main(String a[]){

        List<Empl> list = new ArrayList<Empl>();

        list.add(new Empl("Ram",3000));

        list.add(new Empl("John",6000));

        list.add(new Empl("Crish",2000));

        list.add(new Empl("Tom",2400));

        Collections.sort(list,new MySalaryComp());

        System.out.println("Sorted list entries: ");

        for(Empl e:list){

            System.out.println(e);

        }

    }

}

class MySalaryComp implements Comparator<Empl>{

    @Override

    public int compare(Empl e1, Empl e2) {

        if(e1.getSalary() < e2.getSalary()){

            return 1;

        } else {

            return -1;

        }

    }

}

class Empl{

    private String name;

    private int salary;

    public Empl(String n, int s){

        this.name = n;

        this.salary = s;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getSalary() {

        return salary;

    }

    public void setSalary(int salary) {

        this.salary = salary;

    }

    public String toString(){

        return "Name: "+this.name+"-- Salary: "+this.salary;

    }

}

Sorted list entries:

Name: John-- Salary: 6000

Name: Ram-- Salary: 3000

Name: Tom-- Salary: 2400

Name: Crish-- Salary: 2000

# How to reverse ArrayList content?

 to reverse ArrayList content. You can reverse the content by calling Collections.reverse() method. You have to pass ArrayList instance to this method, which reverses the content.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.Collections;

public class MyArrayListReverse {

    public static void main(String a[]){

        ArrayList<String> list = new ArrayList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        Collections.reverse(list);

        System.out.println("Results after reverse operation:");

        for(String str: list){

            System.out.println(str);

        }

    }

}

Results after reverse operation:

Glass

Watch

Play

Cric

Java

# How to shuffle elements in ArrayList?

# to shuffle elements in the ArrayList. By calling Collections.shuffle() method you can shuffle the content of the ArrayList. Everytime you call shuffle() method, it generates different order of output.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.Collections;

public class MyArrayListShuffle {

    public static void main(String a[]){

        ArrayList<String> list = new ArrayList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        list.add("Movie");

        list.add("Girl");

        Collections.shuffle(list);

        System.out.println("Results after shuffle operation:");

        for(String str: list){

            System.out.println(str);

        }

        Collections.shuffle(list);

        System.out.println("Results after shuffle operation:");

        for(String str: list){

            System.out.println(str);

        }

    }

}

Results after shuffle operation:

Movie

Girl

Watch

Glass

Java

Cric

Play

Results after shuffle operation:

Glass

Watch

Play

Girl

Cric

Movie

Java

# How to swap two elements in a ArrayList?

# To swap two elements in the ArrayList. By calling Collections.swap() method you can swap two elements of the ArrayList. You have to pass the indexes which you need to swap.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.Collections;

public class MyArrayListSwap {

    public static void main(String a[]){

        ArrayList<String> list = new ArrayList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        list.add("Movie");

        list.add("Girl");

        Collections.swap(list, 2, 5);

        System.out.println("Results after swap operation:");

        for(String str: list){

            System.out.println(str);

        }

    }

}

Results after swap operation:

Java

Cric

Movie

Watch

Glass

Play

Girl

# How to convert list to csv string format?

# To convert given list of strings to comma seperated values (csv) format.

package com.java2novice.arraylist;

import java.util.ArrayList;

import java.util.List;

public class MyListToCsvString {

    public String getListAsCsvString(List<String> list){

        StringBuilder sb = new StringBuilder();

        for(String str:list){

            if(sb.length() != 0){

                sb.append(",");

            }

            sb.append(str);

        }

        return sb.toString();

    }

    public static void main(String a[]){

        List<String> li1 = new ArrayList<String>(){

            {

                this.add("animal");

                this.add("nuts");

                this.add("java");

            }

        };

        MyListToCsvString mtc = new MyListToCsvString();

        System.out.println(mtc.getListAsCsvString(li1));

        List<String> li2 = new ArrayList<String>(){

            {

                this.add("java");

                this.add("unix");

                this.add("c++");

            }

        };

        System.out.println(mtc.getListAsCsvString(li2));

    }

}

animal,nuts,java

java,unix,c++

# Java LinkedList Examples

# LinkedList is a linked list implementation of the List interface. Implements all optional list operations, and permits all elements (including null). In addition to implementing the List interface, the LinkedList class provides uniformly named methods to get, remove and insert an element at the beginning and end of the list. These operations allow linked lists to be used as a stack, queue, or double-ended queue.

# Basic LinkedList Operations.

# For basic operations like creating object for LinkedList, adding objects into LinkedList, searching an object in LinkedList, whether it is listed under this LinkedList instance or not, checking whether the LinkedList is empty or not, and finally size of the LinkedList.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyBasicOperations {

    public static void main(String a[]){

        LinkedList<String> ll = new LinkedList<String>();

        ll.add("Orange");

        ll.add("Apple");

        ll.add("Grape");

        ll.add("Banana");

        System.out.println(ll);

        System.out.println("Size of the linked list: "+ll.size());

        System.out.println("Is LinkedList empty? "+ll.isEmpty());

        System.out.println("Does LinkedList contains 'Grape'? "+ll.contains("Grape"));

    }

}

[Orange, Apple, Grape, Banana]

Size of the linked list: 4

Is LinkedList empty? false

Does LinkedList contains 'Grape'? true

# How to read all elements in LinkedList by using iterator?

# for reading all elements from LinkedList by using Iterator. Also you can iterate through the LinkedList based on index too.

package com.java2novice.linkedlist;

import java.util.Iterator;

import java.util.LinkedList;

public class MyLinkedListIterate {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        Iterator<String> itr = arrl.iterator();

        while(itr.hasNext()){

            System.out.println(itr.next());

        }

    }

}

First

Second

Third

Random

# How to copy or clone a LinkedList?

# for creating duplicate object of an LinkedList instance. we can done this by using clone() function.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyLinkedListClone {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual LinkedList:"+arrl);

        LinkedList<String> copy = (LinkedList<String>) arrl.clone();

        System.out.println("Cloned LinkedList:"+copy);

    }

}

Actual LinkedList:[First, Second, Third, Random]

Cloned LinkedList:[First, Second, Third, Random]

# How to add all elements of a list to LinkedList?

# for copying another collection instance objects to existing LinkedList.

package com.java2novice.linkedlist;

import java.util.ArrayList;

import java.util.LinkedList;

import java.util.List;

public class MyLinkedListNewCollection {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual LinkedList:"+arrl);

        List<String> list = new ArrayList<String>();

        list.add("one");

        list.add("two");

        arrl.addAll(list);

        System.out.println("After Copy: "+arrl);

    }

}

Actual LinkedList:[First, Second, Third, Random]

After Copy: [First, Second, Third, Random, one, two]

# How to delete all elements from my LinkedList?

# for deleting all objects from LinkedList at one method call. We can do this by calling clear() method on LinkedList, it will delete all objects.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class ClearMyLinkedList {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual LinkedList:"+arrl);

        arrl.clear();

        System.out.println("After clear LinkedList:"+arrl);

    }

}

Actual LinkedList:[First, Second, Third, Random]

After clear LinkedList:[]

# How to find does LinkedList contains all list elements or not?

# for finding whether the instance of an LinkedList contains all objects of another Collection instance. Here we are checking with another List instance.

package com.java2novice.linkedlist;

import java.util.LinkedList;

import java.util.List;

public class MyElementCheck {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        List<String> list = new LinkedList<String>();

        list.add("Second");

        list.add("Random");

        System.out.println("Does LinkedList contains all list elements?: "

                                +arrl.containsAll(list));

        list.add("one");

        System.out.println("Does LinkedList contains all list elements?: "

                                +arrl.containsAll(list));

    }}

Does LinkedList contains all list elements?: true

Does LinkedList contains all list elements?: false

# How to copy LinkedList to array?

for copying all content of LinkedList to an array. You can get this done by calling toArray() method.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyLinkedListArray {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Actual LinkedList:"+arrl);

        String[] strArr = new String[arrl.size()];

        arrl.toArray(strArr);

        System.out.println("Created Array content:");

        for(String str:strArr){

            System.out.println(str);

        }

    }

}

Actual LinkedList:[First, Second, Third, Random]

Created Array content:

First

Second

Third

Random

# How to get sub list from LinkedList?

 for getting LinkedList content based on range of index. By specifing the start and end index of the range, you can get the sub-list.

package com.java2novice.linkedlist;

import java.util.LinkedList;

import java.util.List;

public class MyLinkedListSubRange {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        //adding elements to the end

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        arrl.add("Click");

        System.out.println("Actual LinkedList:"+arrl);

        List<String> list = arrl.subList(2, 4);

        System.out.println("Sub List: "+list);

    }

}

Actual LinkedList:[First, Second, Third, Random, Click]

Sub List: [Third, Random]

# How to sort LinkedList using Comparator?

to sort an LinkedList using Comparator. The LinkedList contains user defined objects. By using Collections.sort() method you can sort the LinkedList. You have to pass Comparator object which contains your sort logic. The example sorts the Empl objects based on highest salary.

package com.java2novice.linkedlist;

import java.util.Collections;

import java.util.Comparator;

import java.util.LinkedList;

public class MyLinkedListSort {

    public static void main(String a[]){

        LinkedList<Empl> list = new LinkedList<Empl>();

        list.add(new Empl("Ram",3000));

        list.add(new Empl("John",6000));

        list.add(new Empl("Crish",2000));

        list.add(new Empl("Tom",2400));

        Collections.sort(list,new MySalaryComp());

        System.out.println("Sorted list entries: ");

        for(Empl e:list){

            System.out.println(e);

        }

    }

}

class MySalaryComp implements Comparator<Empl>{

    @Override

    public int compare(Empl e1, Empl e2) {

        if(e1.getSalary() < e2.getSalary()){

            return 1;

        } else {

            return -1;

        }

    }

}

class Empl{

    private String name;

    private int salary;

    public Empl(String n, int s){

        this.name = n;

        this.salary = s;

    }

    public String getName() {

        return name;

    }

    public void setName(String name) {

        this.name = name;

    }

    public int getSalary() {

        return salary;

    }

    public void setSalary(int salary) {

        this.salary = salary;

    }

    public String toString(){

        return "Name: "+this.name+"-- Salary: "+this.salary;

    }

}

Sorted list entries:

Name: John-- Salary: 6000

Name: Ram-- Salary: 3000

Name: Tom-- Salary: 2400

Name: Crish-- Salary: 2000

# How to reverse LinkedList content?

 to reverse LinkedList content. You can reverse the content by calling Collections.reverse() method. You have to pass LinkedList instance to this method, which reverses the content.

package com.java2novice.linkedlist;

import java.util.Collections;

import java.util.LinkedList;

public class MyLinkedListReverse {

    public static void main(String a[]){

        LinkedList<String> list = new LinkedList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        Collections.reverse(list);

        System.out.println("Results after reverse operation:");

        for(String str: list){

            System.out.println(str);

        }}}

Results after reverse operation:

Glass

Watch

Play

Cric

Java

# How to shuffle elements in LinkedList?

to shuffle elements in the LinkedList. By calling Collections.shuffle() method you can shuffle the content of the LinkedList. Everytime you call shuffle() method, it generates different order of output.

package com.java2novice.linkedlist;

import java.util.Collections;

import java.util.LinkedList;

public class MyLinkedListShuffle {

    public static void main(String a[]){

        LinkedList<String> list = new LinkedList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        list.add("Movie");

        list.add("Girl");

        Collections.shuffle(list);

        System.out.println("Results after shuffle operation:");

        for(String str: list){

            System.out.println(str);

        }

        Collections.shuffle(list);

        System.out.println("Results after shuffle operation:");

        for(String str: list){

            System.out.println(str);

        }

    }

}

|  |
| --- |
| **Output:** |
| Results after shuffle operation:  Movie  Girl  Watch  Glass  Java  Cric  Play  Results after shuffle operation:  Glass  Watch  Play  Girl  Cric  Movie  Java |

# How to swap two elements in a LinkedList?

to swap two elements in the LinkedList. By calling Collections.swap() method you can swap two elements of the LinkedList. You have to pass the indexes which you need to swap.

package com.java2novice.linkedlist;

import java.util.Collections;

import java.util.LinkedList;

public class MyLinkedListSwap {

    public static void main(String a[]){

        LinkedList<String> list = new LinkedList<String>();

        list.add("Java");

        list.add("Cric");

        list.add("Play");

        list.add("Watch");

        list.add("Glass");

        list.add("Movie");

        list.add("Girl");

        Collections.swap(list, 2, 5);

        System.out.println("Results after swap operation:");

        for(String str: list){

            System.out.println(str);

        }

    }

}

Results after swap operation:

Java

Cric

Movie

Watch

Glass

Play

Girl

# How to add element at first position in LinkedList?

 to add element at first position in LinkedList. LinkedList provides few methods to add element at first position, those methods are:  
  
**addFirst():**Inserts the specified element at the beginning of this list.  
**offerFirst():**Inserts the specified element at the front of this list.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyAddFirst {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println(arrl);

        System.out.println("Adding element at first position...");

        arrl.addFirst("I am first");

        System.out.println(arrl);

        System.out.println("Adding element at first position...");

        arrl.offerFirst("I am first - 2");

        System.out.println(arrl);

    }

}

[First, Second, Third, Random]

Adding element at first position...

[I am first, First, Second, Third, Random]

Adding element at first position...

[I am first - 2, I am first, First, Second, Third, Random]

# How to add element at last position in LinkedList?

to add element at last position in LinkedList. LinkedList provides few methods to add element at last position, those methods are:  
  
**addLast():**Appends the specified element to the end of this list.  
**offerLast():**Inserts the specified element at the end of this list.  
**offer():**Adds the specified element as the tail (last element) of this list.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyAddLast {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println(arrl);

        System.out.println("Adding element at last position...");

        arrl.addLast("I am last");

        System.out.println(arrl);

        System.out.println("Adding element at last position...");

        arrl.offerLast("I am last - 1");

        System.out.println(arrl);

        System.out.println("Adding element at last position...");

        arrl.offer("I am last - 2");

        System.out.println(arrl);

    }

}

[First, Second, Third, Random]

Adding element at last position...

[First, Second, Third, Random, I am last]

Adding element at last position...

[First, Second, Third, Random, I am last, I am last - 1]

Adding element at last position...

[First, Second, Third, Random, I am last, I am last - 1, I am last - 2]

# How to read first element from LinkedList?

 to read first element from LinkedList. LinkedList provides few methods to read first element, those methods are:  
  
**element():**Retrieves, but does not remove, the head (first element) of this list.  
**getFirst():**Returns the first element in this list.  
**peek():**Retrieves, but does not remove, the head (first element) of this list.  
**peekFirst():**Retrieves, but does not remove, the first element of this list, or returns null if this list is empty.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyFirstElement {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("First Element: "+arrl.element());

        System.out.println("First Element: "+arrl.getFirst());

        System.out.println("First Element: "+arrl.peek());

        System.out.println("First Element: "+arrl.peekFirst());

    }

}

First Element: First

First Element: First

First Element: First

First Element: First

# How to read last element from LinkedList?

 to read last element from LinkedList. LinkedList provides few methods to read last element, those methods are:  
  
**getLast():**Returns the last element in this list.  
**peekLast():**Retrieves, but does not remove, the last element of this list, or returns null if this list is empty.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyLastElement {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println("Last Element: "+arrl.getLast());

        System.out.println("Last Element: "+arrl.peekLast());

    }

}

Last Element: First

Last Element: Random

Last Element: Random

# How to iterate through LinkedList in reverse order?

to iterate through LinkedList in reverse order. The method descendingIterator() returns an Iterator object with reverse order. By iterating through it, you can get the elements in the reverse order.

package com.java2novice.linkedlist;

import java.util.Iterator;

import java.util.LinkedList;

public class MyReverseIter {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        Iterator<String> itr = arrl.descendingIterator();

        while(itr.hasNext()){

            System.out.println(itr.next());

        }

    }

}

Random

Third

Second

First

# LinkedList push(), pop() operations examples.

 to call push() and pop() methods on LinkedList objects.  
  
**push():**Pushes an element onto the stack represented by this list.  
**pop():**Pops an element from the stack represented by this list.

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyPushPopOpr {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        System.out.println(arrl);

        arrl.push("push element");

        System.out.println("After push operation:");

        System.out.println(arrl);

        arrl.pop();

        System.out.println("After pop operation:");

        System.out.println(arrl);

    }

}

[First, Second, Third, Random]

After push operation:

[push element, First, Second, Third, Random]

After pop operation:

[First, Second, Third, Random]

# How to remove elements from LinkedList?

 to remove or delete an element from LinkedList. LinkedList provides few methods to remove elements, those methods are:  
  
**remove():**Retrieves and removes the head (first element) of this list.  
**remove(index):**Removes the element at the specified position in this list.  
**remove(object):**Removes the first occurrence of the specified element from this list, if it is present.  
**removeFirst():**Removes and returns the first element from this list.  
**removeFirstOccurrence(object):**Removes the first occurrence of the specified element in this list (when traversing the list from head to tail).  
**removeLast():**Removes and returns the last element from this list.  
**removeLastOccurrence(object):**Removes the last occurrence of the specified element in this list (when traversing the list from head to tail).

package com.java2novice.linkedlist;

import java.util.LinkedList;

public class MyAllRemoveOprs {

    public static void main(String a[]){

        LinkedList<String> arrl = new LinkedList<String>();

        arrl.add("First");

        arrl.add("Second");

        arrl.add("Third");

        arrl.add("Random");

        arrl.add("four");

        arrl.add("five");

        arrl.add("six");

        arrl.add("seven");

        arrl.add("eight");

        arrl.add("nine");

        System.out.println(arrl);

        System.out.println("Remov() method:"+arrl.remove());

        System.out.println("After remove() method call:");

        System.out.println(arrl);

        System.out.println("remove(index) method:"+arrl.remove(2));

        System.out.println("After remove(index) method call:");

        System.out.println(arrl);

        System.out.println("Remov(object) method:"+arrl.remove("six"));

        System.out.println("After remove(object) method call:");

        System.out.println(arrl);

        System.out.println("removeFirst() method:"+arrl.removeFirst());

        System.out.println("After removeFirst() method call:");

        System.out.println(arrl);

        System.out.println("removeFirstOccurrence() method:"

                            +arrl.removeFirstOccurrence("eight"));

        System.out.println("After removeFirstOccurrence() method call:");

        System.out.println(arrl);

        System.out.println("removeLast() method:"+arrl.removeLast());

        System.out.println("After removeLast() method call:");

        System.out.println(arrl);

        System.out.println("removeLastOccurrence() method:"

                            +arrl.removeLastOccurrence("five"));

        System.out.println("After removeLastOccurrence() method call:");

        System.out.println(arrl);

    }

}

[First, Second, Third, Random, four, five, six, seven, eight, nine]

Remov() method:First

After remove() method call:

[Second, Third, Random, four, five, six, seven, eight, nine]

remove(index) method:Random

After remove(index) method call:

[Second, Third, four, five, six, seven, eight, nine]

Remov(object) method:true

After remove(object) method call:

[Second, Third, four, five, seven, eight, nine]

removeFirst() method:Second

After removeFirst() method call:

[Third, four, five, seven, eight, nine]

removeFirstOccurrence() method:true

After removeFirstOccurrence() method call:

[Third, four, five, seven, nine]

removeLast() method:nine

After removeLast() method call:

[Third, four, five, seven]

removeLastOccurrence() method:true

After removeLastOccurrence() method call:

[Third, four, seven]