Iterator Interface

**Iterator interface provides the facility of iterating the elements in a forward direction only.**

**Methods of Iterator Interface**

**There are only three methods in the Iterator interface. They are :**

|  |  |  |
| --- | --- | --- |
| **No** | **Method** | **Description** |
| **1** | public boolean hasNext() | It returns true if the iterator has more elements otherwise it returns false |
| **2** | public Object next() | It returns the element and moves the cursor pointer to the next element |
| **3** | public void remove() | It removes the last elements returned by the iterator. It is less used. |

**Iterable Interface**

The Iterable Interface is the root interface for all the Collection classes . The Collection interface extends the Iterable Interface and therefore all the subclasses of Collection interface also implement the Iterable Interface.

It contains only one abstract method. i.e,

1.Iterator <T> iterator ()

It returns the Iterator over the elements of type T.

**Collection Interface**

**The Collection interface is the Interface which is implemented by all the classes in the collection framework. It declares the methods that every collection will have. In other words , we can say that the collection interface builds the foundations on which the Collection framework depends.**

**Some of the methods of Collections interface are boolean add (Object obj) ,boolean addAll( Collection c), void clear() , etc.. which are implemented by all the subclasses of Collection interface.**

**Collection Frame work [ Zahangir Alam]**

The Collection in java is a framework that provides an architecture to store and manipulate the group of objects. it has child which includes some interface and some class.

All the operations that you perform on a data such as searching, sorting , insertion, manipulation, deletion , etc. can be achieved by java Collections.

**Java collection** means a single unit of objects. Java Collection framework provides many interfaces (List, Queue -Deque, Set- SortedSet, etc) and classes ()

What is collection in Java ?

A collection represents a single unit of objects, i.e, a group.

What is framework in java?

* It provides readymade architecture.
* It represent a set of classes and interfaces
* It is optional

What is collection framework?

The collection framework represents a unified architecture for sorting and manipulating a group of objects. It has :

1.Interfaces and its implementations, i.e. classes

2. Algorithm

**Interface includes** –Iterable, Collection,List, Queue -Deque, Set- SortedSet.

**Class Includes** – ArrayList, LinkedList, Vector , Stack, Priority Queue, ArrayDeque, HashSet, LinkedHashSet, TreeSet.

**Implements** – class implements interface

**Extend**—One Interface extends another interface

**Map** –To keep data using a key value pair

In Map ---

**Interface includes** – Map, SortedMap.

**Class Includes- HashMap, LinkedHashMap. TreeMap**

Iterable [Green = Class and Yellow = Interface]

Collection

List Queue Set

ArrayList HashSet

priorityQueue

LinkedHashSet

Deque

LinkedList

SortedSet

Vector ArrayDeque TreeSet

Stack

Green = Class and Yellow = Interface

Map

SortedMap HashMap

LinkedHashMap

TreeMap

**List Interface**

List Interface is the child Interface of Collection interface. It inhibits a list type data structure in which we can store the ordered collection of objects. It can have duplicate values.

List interface is implemented by the classes ArrayList, LinkedList, Vector and Stack.

To initialize the List Interface, we must use :

List<data-type> list 1 = new ArrayList();

List<data-type> list 2 = new LinkedList();

List<data-type> list 3 = new Vector();

List<data-type> list 4 = new Stack();

There are various methods in List interface that can be used to insert, delete, and access the elements from the list.

The classes that implements the List interface are given below.

**ArrayList**

The ArrayList class implements the List interface . It uses a dynamic array to store the duplicate elements of different data types. The ArrayList class maintains the Insertion order and is non-synchronized. The elements stored in the ArrayList class can be randomly accessed. Consider the following example –

Import java.util.\*;

class TestJAvaCollection1{

public static void main (String [] args){

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

list.add(“Ruma”); //Adding object in arraylist

list.add(“keya”);

list.add(“Ruma”);

list.add(“Afsheen”);

// Traversing list through Iterator

Iterator itr = list.iterator();

While (itr.hasNext()){

Syso(itr.next());

}}

Output :

Ruma

Keya

Ruma

Afsheen

**LinkedList**

LinkedList implements the Collection interface. It uses a doubly linked list internally to store the elements. It can store the duplicate elements. It maintains the insertion order and is not synchronized. In LinkedList, the manipulation is fast because no shifting is required.

Consider the following example :

Import java.util.\*;

class TestJAvaCollection2{

public static void main (String [] args){

LinkedList <String> link = new LinkedList <String>(); // creating LinkedList

link .add(“Ruma”); //Adding object in linkedist

link.add(“keya”);

link.add(“Ruma”);

link.add(“Afsheen”);

// Traversing list through Iterator

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

While (itr.hasNext()){

Syso(itr.next());

}}}

Output :

Ruma

Keya

Ruma

Afsheen

**Vector**

Vector uses a dynamic array to store the data elements. it is similar to ArrayList. However, it is synchronize and contains many methods that are not the of collection framework.

Consider the following example –

Import java.util.\*;

class TestJAvaCollection3{

public static void main (String [] args){

Vector <String> Vk = new Vector<String>(); // creating Vector

Vk .add(“Ayush”); //Adding object in linkedist

Vk.add(“Amit”);

Vk.add(“Ashish”);

Vk.add(“Garima”);

// Traversing list through Iterator

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

While (itr.hasNext()){

Syso(itr.next());

}}}

Output :

Ayush

Amit

Ashish

Garima

**Stack**

The stack is the subclass of Vector. It implements the last-in-first-out data structure, i.e, Stack. The stack contains all of the methods of Vector class and also provides its methods like boolean push(), boolean peek(), boolean push(object.o), which defines its properties.

Consider the following example :

Import java.util.\*;

class TestJAvaCollection4{

public static void main (String [] args){

Stack <String> stack = new Stack<String>(); // creating Stack

Stack.push(“Ayush”); //Adding object in

stack.push(“Gravit”);

stack.push (“Amit”);

stack.push (“Ashish”);

stack.push(Garima);

stack.pop();

// Traversing list through Iterator

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

While (itr.hasNext()){

Syso(itr.next());

}}}

Output :

Ayush

Amit

Ashish

…………….

**Queue Interface**

Queue interface maintains the first-in-first-out order. It can be defined as an ordered list that is used to hold the elements which are about to be processed . There are Various classes like Priority Queue, Deque and ArrayDeque which implements the Queue interface .

Queue interface can be instantiated as :

Queue<String> q1 = new PriorityQueue();

Queue<String> q2 = new ArrayDeque();

There are various classes that implements the Queue interface , some of them are given below—

**PriorityQueue**

The priorityQueue class implements the Queue interface. It holds the elements or objects which are to be processed by their priorities. PriorityQueue doesn’t allow null values to be stored in the queue.

Consider the following example ---

Import java.util.\*;

class TestJAvaCollection5{

public static void main (String [] args){

PriorityQueue <String> priority = new PriorityQueue <String>();

priority.add(“Amit sharma”); //Adding object in

priority.add(“Gravit”);

priority.add (“Vijay Raj”);

priority.add (“JaiShankar”);

priority.push(Raj);

Syso(“head : + priority.element());

Syso(“head : + priority.peek());

Syso(“iterating the queue elements: “);

Iterator itr = priority. Iterator ();

While (itr.hasNext()){

Syso(itr.next());

}

priority.remove();

priority.poll();

syso(“after removing two elements : ”);

Iterator <String> itr2 = priority .iterator();

While (itr2 .hasNext()){

Syso(itr2.next());

….

Output :

Head: Amit sharma

Head: Amit sharma

Iterating the queue elements:

Amit Sharma

Raj

Jaishankar

Vijay Raj

After removing two elements- Raj , Vijay Raj

**Deque Interface**

Deque interface extends the Queue interface. In Deque, we can remove and add the elements from both the side. Deque stands for a double-ended queue which enables us to perform the operations at both the ends.

Deque can be instantiated as :

Deque d = new ArayDeque();

**ArrayDeque**

ArrayDeque class implements the deque interface . It facilitates us to use the Deque. Unlike queue, we can add or delete the elements from both the ends.

ArayDeque is faster than ArrayList and Stack and has no capacity restrictions.

Consider the following example ---

Import java.util.\*;

class TestJAvaCollection6{

public static void main (String [] args){

//Creating Deque and Adding elements

Deque <String> deque = new ArrayDeque <String>();

deque.add(“Gautam”);

deque.add(“Karan”);

deque.add(“Ajay”);

//Traversing elements

for(String str : deque){

syso (str);

}}}

Output :

Gautam

Karan

Ajay

**Set Interface**

Set Interface in java is present in java.util package. It extends the Collection interface. It represents the unordered set of elements which doesn’t allow us to store the duplicate items. We can store at most one null value in Set. Set is implemented by HashSet, LinkedHashSet and TreeSet .

Set can be instantiated as :---

Set<data-type> s1 = new HashSet<data-type>();

Set<data-type> s2 = new LinkedHashSet<data-type>();

Set<data-type> s1 = new TreeSet<data-type>();

**HashSet**

HashSet class implements Set interface. It represents the collection that uses a hash table for storage. Hashing is used to store the elements in the HashSet. It contains unique items .

Consider the following example –

Import java.util.\*;

class TestJAvaCollection7{

public static void main (String [] args){

//Creating HashSet and Adding elements

HashSet <String> Hset = new HashSet <String>();

hset.add(“Ravi”);

hset.add(“Vijay”);

hset.add(“Ravi”);

hset.add(“Ajay”);

//Traversing elements

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

While (itr.hasNext())

Syso(itr.next());

}}}

Output:

Vijay

Ravi

Ajay

**LinkedHashSet**

LinkedHashSet class represents the LinkedList implementation of Set interface. It Extends the HashSet class and implements Set interface. Like HashSet. It also contains unique elements. It maintains the insertion order and permits null elements.

Consider the following example –

Import java.util.\*;

class TestJAvaCollection8{

public static void main (String [] args){

//Creating Linked HashSet and Adding elements

LinkedHashSet <String> Lset = new LinkedHashSet <String>();

Lset.add(“Ravi”);

Lset.add(“Vijay”);

Lset.add(“Ravi”);

hset.add(“Ajay”);

//Traversing elements

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

While (itr.hasNext())

Syso(itr.next());

}}}

Output:

Ravi

Vijay

Ajay

**SortedSet Interface**

SortedSet is the alternative of Set Interface that provides a total ordering on its elements. The elements of the SortedSet are arranged in the increasing (ascending ) order . The sortedSet provides the additional methods that inhibit the natural ordering of the elements.

The SortedSet can be instantiated as ----

SortedSet<data-type > sorest = new TreeSet();

**TreeSet**

Java TreeSet class implements the Set Interface that uses a tree for storage. Like HashSet. TreeSet also Contains unique elements. However, the access and retrieval time of TreeSet is quite fast. The elements in TreeSet stored in ascending order.

Consider the following Example ---

Import java.util.\*;

class TestJAvaCollection9{

public static void main (String [] args){

//Creating Linked HashSet and Adding elements

TreeSet <String> Tset = new TreeSet <String>();

Tset .add(“Ravi”);

Tset.add(“Vijay”);

Tset.add(“Ravi”);

Tset.add(“Ajay”);

//Traversing elements

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

While (itr.hasNext())

Syso(itr.next());

}}}

Output:

Ajay

Ravi

Vijay

**TalentSprint coding Prep**

Collections

Framework for storing and manipulating a group of objects. It is derived from java.util.package.

Q How to navigate through the list or how to move through all the objects in the list.

Collections provide you a list of methods , Out of which the very basic and highly used ones are

add()

remove()

size()

clear()

contains()

iterator ()

isEmpty

**add() – it is simply used to add elements to your collections.**

**remove() --- it is used to remove elements from your collection**

**size()--------- used to give you the size of the collection.**

**clear()--- it will delete all elements from the collection**

**contains() –is going to tell you whether your collections contains a particular elements or not.**

**iterator () – most important in java collection. It helps you navigate through the elements of collection.**

**isEmpty------ it Returns Boolean value depending on whether the collection has any value or not.**

**Q Diff between Iterator and Iterable ?**

**Iterable :**

A class that can be iterated over. It contains only one method iterator()

It just moving forward and backward .

Iterable is the functionality and Iterator is what actually executes the functionality and holds the state at any point of tim.

It will just tell you that this particular collection framework that is extending iterable will available for iteration.

You can go backward and forward , that is the sole purpose of

Iterator

It manage iteration over an iterable. It keeps track of where we are in the current location and knows what the next element is and how to get it .

It contains three method hasNext(). Next(), remove().

**Like this is Array-- 5,6,9** and Iterator is pointing to 5 and you do has next, it will return true. Since we have a next element that is six. And it will give you the output 6. And remove () it will remove this current element from the List.

hasNext()--- Like this is an Array – and Iterator is pointing 9 is the current position and If I do dot hasnext(). It will return either true or false this. Function returns boolean value true or false. Depending on whether this list element has next value or not.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 8 | 9 | 10 | 7 | 9 |

If Let’s see we have array pointer is at the last , So there are no more elements after this and if we do hasNext at that point , it will return false.

So It is generally used to travels. Like Move to the Array from starting till hasnext.

Example –Array.hasNext(), or whatever collection you are using.

dothasNext repeat the loop until you have travels through the whole list

The Current state has next value or not. And Next will give you the value.

So iterator will hold the current state and Iterable just let you know that you can move through this list or Map or whatever and you can traverse through it.

Example – like this is an Array {5 ,2,1,8,7,5,}

If you are at this particular position at any point of time. Let’s see Iterator is pointing to 1. Iterator is what is responsible to tell you which element at any point of time you are referring to, What is the next element that would be 8 , what is the previous element that would be 2 and also if at all you want to manipulate this current element.

So Iterator keeps the current state in itself. It knows where is the picture it is pointing to

So iterator will hold the current state and Iterable just let you know that you can move through this list or Map or whatever and you can traverse through it but

next () it will give the value at this position [ 10 ] or in this case

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 8 | 9 | 10 | 7 | 9 |

hasNext()----Let’s There is this Array--- and these are the indexes . So this is an Array of size 6 and length of index 5 and it has elements inside the Array {5,8,9,10,7,9}

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 5 | 8 | 9 | 10 | 7 | 9 |

0 1 2 3 4 5—indexes

So here we are iterating through the Array and we are pointing to index number 3 and that has value 10. So at the point of time if we check whether the **Array.hasNext() , this function will return true because the Array has this Next element which has value 7.**

Now you have travels through the Array Completely and you are finally at the last position that is index number 5 which has value 9 . In that case if we run this function again it will return you null value or it will return you basically false, because after 5 index we have no more values.

So has next simply check whether you have a next value or not.

hasNest returns false only when you are at the last location in which there is no value to operate on.

**Iterator VS ListIterator**

**Iterator -**

These are the three methods / functions of Iterator--

a.hasNext()

b.next ()

c. remove()

Iterator is just checking whether it has next element or not. The first method is hasNext that will help you check whether there is a next element od not. But we can notice that there is no method provided to check for the previous element that have passed. So here there is no has previous .

SO Itrator do just forward reversal , since they ae talking about the next element and removing the current element .

Iterator – Forward Traversal

ListIterator however do it backwards as well . they are bi-directional .

**Example -** if you have a situation in your program wherein you have to traverse forward as well as backwoard. You will be using ListIterator to do that functionality for you.

ListIterator

– Extends Iterator – because it has to operate forward and additionally it operates backward as well so it is bi-directional traversal

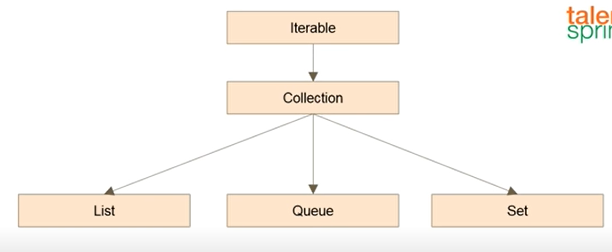
-BiDirectional traversal

Q – can we use loops for all these collections?

Yes . but purpose of collection is to make our task easy.

The purpose of creating collections is to make our task easy, so that we don’t have to write big bulky lengthy for loops. We can just use collections to do the task for us like pre-existing functions to do the task. And then we can actually focus on the logic or the programming that we have to do. So that is the reason why all these different type of collections have been created depending on program requirement and the data that we are working on.

**Collection Hierarchy**



Collections all of these are interfaces , none of this is a class. Iterable ,Collection, List, Queue and Set are the interfaces .

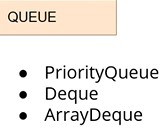
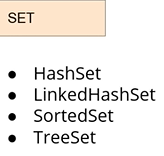
So Collection is extending Iterable. And List , Queue and Set Extend collection.

Whatever the functions and Functionalities of Iterable , they are inherited by collections and then further down to list and Queue and Set.

Java Considers Map or Mapping as not a part of collection, because they are separate from collection.

Collections hold element . so they just talk about elements in element traversals Mut Maps hold key value pairs. So there are two elements in one single node in a Map . Other than that in linkedList or tree etc there is just one element in a node but In a Map there are two elements in a node, because of which it creates a mapping between the Key and the Value. So for a particular key there is a value.

That is the reason why Maps are not in the collection hierarchy chats.

**Keep in Mind—**

1. Which particular of these collections frameworks will return your values in a shorter point of time that is one of the pointer.
2. Which of these collection framework will help you access the data faster , so if at all the requirement of the program is just accessing the data and there is no manipulation involved, which one will you prefer that is the another thing
3. Third is accessing the first and the last elements. so there are chances that your program would want you to just access first and last elements and Ignore the middle elements .

**So which one would you prefer in those conditions.**

* **Difference between ArrayList and Vector**

In ArrayList Size is increased by 50 % of the Array Size while In Vector size is doubled if you encounter the limit in size.

* Difference between ArrayList and LinkedList

ArrayList is a dynamic Array while LinkedList is doubly likedList.

* Difference between HashMap and HashTable?

HashMap is not Synchronized but HashTable is Synchronized

* HashSet ?
* Whenever we are talking about Hash codes like it be HashMap , it be HashTable , it be HashSet , whatever it is. We are talking about values that give us the pointer to the actual values where our elements are stored .

Example – let’s see HashSet . So we have this Set of values . one thing is that we traverse through them through the Indexes.

0 1 2 3 4 5 6

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 10 | 8 | 12 | 15 | 6 | 18 |

When we use a for loop , So for(int i=0; i<=size; ++1) {

And move through the indexes one by one and you do your operation.

………………….

……………………..

and

}

Now Like you have hasNext again you simply do while loop, and your what ever set ot any or array. So While we have a next element , do these operations on every element

While Set .hasNext{

}

Similarly HashSet would work on the hash values of these individual element---

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| 7 | 10 | 8 | 12 | 15 | 6 | 18 |

……………………..

**Q – Which of these methods Sets every element of a list into a specified Objects?**

**A. set()**

**B. complete()**

**C. fill()**

**D. add()**

**Here fill() is the right answer. So like you have a list with different values like 7,10,8,12,15,6 and you want to remove all these values and just store Seven in all the palaces . so you simply say –list.fill(7) a or that you want to replace all these values with and once you run this all these values will be replace by 7.**

**So you have a List and Let’s say you want to change all the elements of the List to a Single value, ie you had 5,6,7,8,9 but now you want 1,1,1,1,1 in all those four places. So Which of this functions would be used in this case?**

**Add is a simple addition. If you want to add anything to the list . you simply say --List.add and the value. So this will not change all the values in the list.**

**Q- Which methods is used to randomize all elements in a List ?**

**A.rand()**

**B.shuffle()**

**C. randomize()**

**D.ambiguous()**

**Like you have a List with certain elements, now you want a list with random values but the values will remain the same that you have given , they will be just in a random model . what is the function that you would use . B. shuffle.**

**Q .Which of these objects can convert an object into a List?**

**A.setList()**

**B.convertList()**

**C.singletonList()**

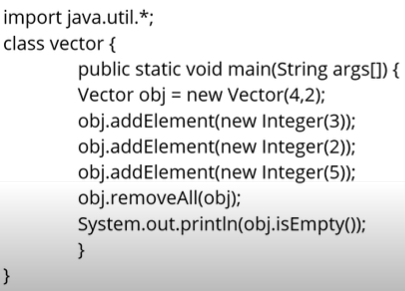
**D.copyList()**

**Ans - C.singletonList() is the right answer.**

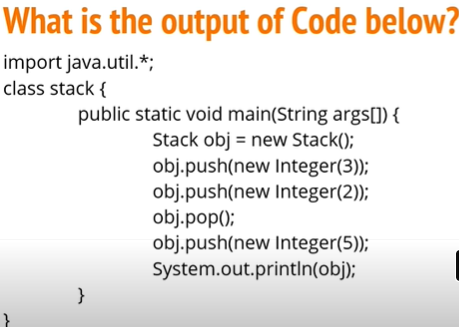
**Example –** like open your computer system 🡪 open Eclipse or whatever IDE that you are using to practice java and > create a List > in this type that list dot. So you get to see all the variations or the functions that are available to manipulate the list and When you click on those functions , you might have seen that every time you click on a functions there is a little help box that pops up that tells you what actually the would be , what output would be , what parameters that the function can take . So I think this IDE’s inherently provide you with all the options to you lean on your own.

So this all you need to do is take out some time every day, let’s say half an hour or one hour or whatever is feasible.

**Q. What will be the output of this Code?**



**First created Vector object (4,2) and Finally add a new element that is 3, so the first element that we add is three then we add 2. And then 5. And then we do remove all obj. remove(obj); here. So will remove all these values from your object ,so that your object doesn’t return have any at all. And when you do this isEmpty () function on the object , it will return true. Since you have already removed all the values . that why isEmpty () gives you true.**



When you create a stack , Let’s say there is a bucket . A stack in real-life reference , you can just call it a bucket and like you are filling it with some stones or carboard or whether you put one on top of other . Now when you want to take these out . you take the Top most out first and the you take the **next and** the next and the next . you can not take the one out from the inside place out before all of these are out of the bucket . Stack Works very similar to this .

So when you create a new stack ,it’s empty .

|  |
| --- |
|  |

Push and pop other operations for adding and deleting . here push means you are pushing a value in it in this bucket of stack and Pop means you are taking that value out from the bucket .

So when you say push three, the first value that goes inside is three ----------

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  | 3 |

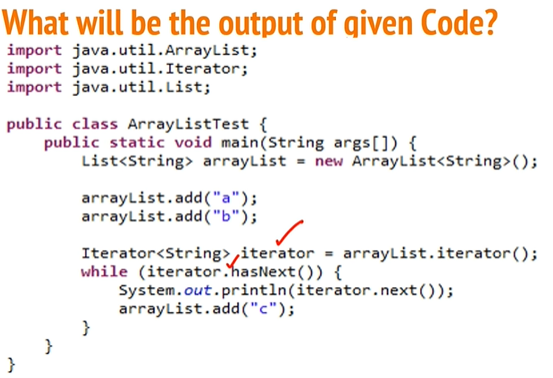
Then push 2 puts 2 inside the bucket ------------

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  | 2 | 3 |

When you say POP, the Top most value only 2 that you can remove is the top most . like I say we can not remove 3. So when you say POP the top most value 2 is removed . then after that 2 is no more .

And again Integer 5 is pushed. So it is 5, 3.

Now when you print your stack , the values would be printed 3 and 5. So when you print they are printed in the order they are inserted .So the first element will be printed first 3 and then followed by next. 3 and 5 is the right answer .



We have an ArrayList and then we have used an Iterator .

We have an ArrayList and we have added two elements to be précised in it --



We have created an Iterator to travel through the ArrayList. While Loop is simply traversing through this ArrayList using an Iterator .

While you are traversing through it , you are trying to add another element in the same ArrayList . 

What that does is you have a thread that is iterating over it .But the same thread that is Iterating is also trying to modify the value .

It is trying to read it but at the same time modify it, which is not permissible when you are working witj a collection. So This program when it reaches hasNext it prints and it reaches this value here [ arrayList.add(“c”); ] It will print –> a and then it will throw concurrent modification exception . Because you are trying to add a value , you are trying to manipulate your list while you are trying to Iterate through the List . Two things are not allowed – notArryIndexOutOfBound .

We are not trying to excess value of the array which is not exist. Array Index out of bound occurs .

The SYSO everything will work fine. So the Iterator will go to hasNext , it will take the false value that is a and it will print it.

As soon as it goes to the next one where it tries to add , it will give you exception , concurrent modification exception.

Another Easy way --When you are in the while loop there is a thread that is going to work on your while loop. So your thread of this object or of this program can not simultaneously do this while loop as well as do the addition

To operations , Iteration and addition cannot be performed simultaneously. it will permissible when working with collections and whenever you try to do that in the same thread that is where you get Concurrent modification exception and then the Next line will throw an exception

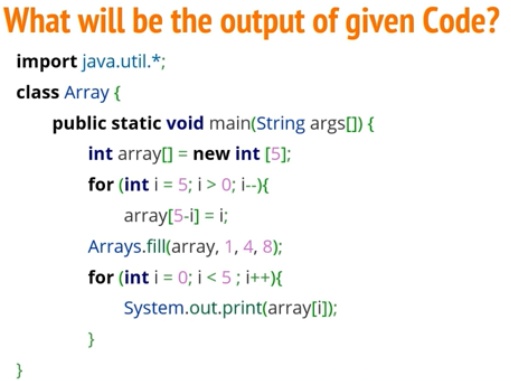
Example of ArrayIndex Out of bound Exception --

If you have Let’s say you have Array and you are trying to print Array let’s say- array [5]. That is when

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 5 | 8 | 10 |

O 1 2 3

You will get . you are trying to access the index 5 when it doesn’t even exits that will give you out of bound exception .In this case when you try to Iterate through and manipulate the same list , then you will get concurrent modification exception .



We created an array of size five and then we are initializing values into it and then we are ding a fill . and finally when you print , what is going to be the out put ?

You need 5,4,3,2,1 is just this first **for loop** where you are initializing the values . you are not printing anything here .When you do this **for loop,** the value in the Array would be 5,4,3,2,1.

But then after you have run this Array.fill() and then finally printing the values. What would be the output here?

What fill does it if you call dot fill and you give some value-, Let’s .fill(7). It will replace all these values of your Array with 7. So your output will become 7,7,7,7,7,7.

Now another variation of this fill function is how we have done here .

So we have given fill and we gave given the Array on which we have to work .fill() which is this Array {5,4,3,2,1} **. So Current Array holds these**{5,4,3,2,1} **values .so we have passed our Array here.**

Now after the Array[ .fill(array, 1,4,8)] they are same three values 1,4,8. So this is the means is the last is the value that you have to fill that is 8.

How you have to fill is what is important here. Like we have 5,4,3,2,1

8 8 8

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 4 | 3 | 2 | 1 |

0 1 2 3 4

So when you say 1 and 4 that means from 1 to 4 .fill(array., 1,4,8)remove all the values and fill it with 8.

So from 1 to 4 , 4 excluding whatever the last index is we are excluding it , we don’t change that value .So 1 till 4 , 3 values [ 1 -3] at index 1,2, and 3 will be changed and they will replaced with 8.

Now when you print this array at this point , your output would come out to be 5,8,8,8,1.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 5 | 8 | 8 | 8 | 1 |

That is how the fill operations works when it is given in this Array.fill(array, 1,4,8) context.

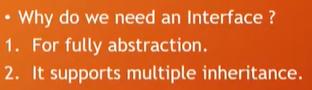
So you pass the Array, you pass the indexes [1 to 4 and 4 is excluding ] from which index till which index.

For Remember the last index is excluded , you have to go till that index, you don’t have to include that index . and 8 is the value with which you have to replace this values .

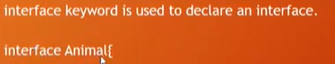
**Interface [Anisul islam ]**



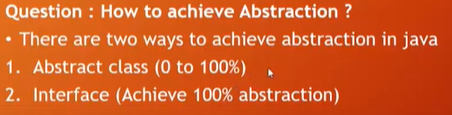
**Note – we can’t create object of Interface. Because it’s can not be instantiated**



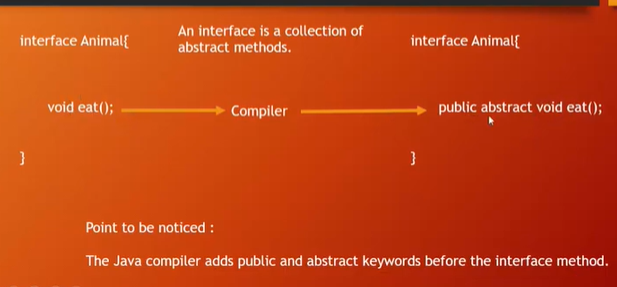




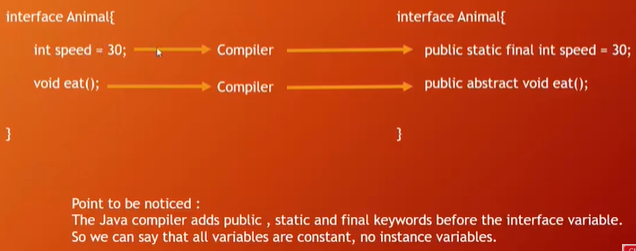
**}**

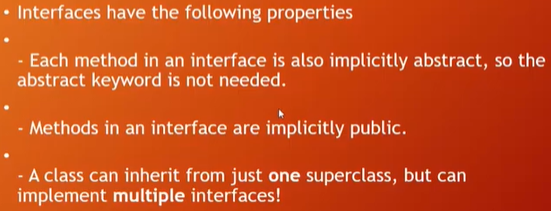


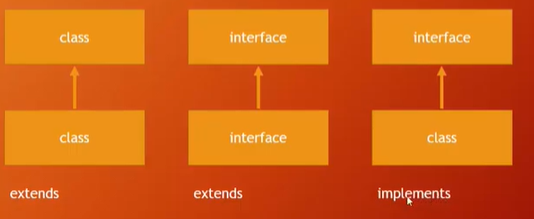


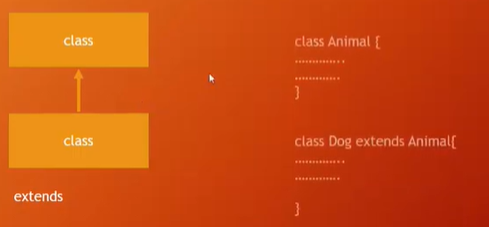








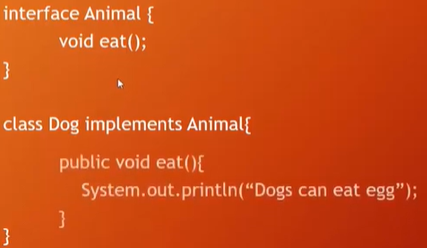












No Implementation /functionality of method in Interface, Because it is abstract

Here class implements the interface and the class can use the method and can give the body or anything else

Interface

Package interface\_demo;

public interface Animal{

public abstract void eat();

}

**Dog Class**

Package interface\_demo;

public class Dog implements Animal {

public void eat(){

System.out.printin(“Dog can bite ”);

}

}

**Note ---If a class implements the a Interface then you can see it shows compile error in the class name. that means this class should be implemented / used all the methods/ Abstract methods of the that Interface**.

**Note – we can’t create object of Interface. Because it’s can not be instantiated**

**Cat Class**

Package interface\_demo;

public class Cat implements Animal {

public void eat(){

System.out.printin(“cat can drink milk ”);

}

}

Java Main Class for Run

**Main Method**

Package interface\_demo;

public class Test {

public static void main(String [] args ){

Dog obj = new Dog();

Obj.eat();

Cat obj1 = new Cat();

Obj1.eat();

}

public void eat(){

System.out.printin(“Dog can bite ”);

}

}

**Set Interface [Anisul islam ]**

**Set VS List**

**List**  contain duplicate component or element . but

Set --In Set there is no duplicate component or element exist. Ie set does not work with duplicate component or element.

**Example of List** – we can put same name or number in the index of the List . but in Set it contain only unique element.

Package set\_demo;

Public class HashSetDemo {

public static void main(String [] args ){

HashSet<String> fruitsName = new HashSet<> () ;

fruitsName.add(“Apple”);

fruitsName.add(“Orange”);

fruitsName.add(“Peach”);

fruitsName.add(“Malta ”);

**for printing –**

System.out.printin(fruitsName);

For Size—

System.out.printin(“size = “+fruitsName.size());

Or for each loop--

for(String fruits : fruitsName)

{

System.out.printin(“all fruits are here : “ + fruitsName);

For Size—

System.out.printin(“all fruits are here : “ + fruitsName.size());

**Methods –**

fruitsName.**remove**(“Apple”);

System.out.println (fruitsName);

fruitsName.**cl**ear**(**);

System.out.println (fruitsName);

boolean check = fruitsName.**isEmpty(**);

System.out.println (check);

}

}

**ArrayList [Anisul islam ]**

**Array and ArrayList**

Array—

* Not Resizable , Size fixed , ie size is changeable ,because it is static
* for loop , for each loop
* array.length
* Fast
* Size fixed , ie size is changeable ,because it is static

ArrayList**— class uses a dynamic array for storing the elements.**

**Advantage-- it is better for sorting and accessing data,**

**Disadvantage –** Slow for manipulating data (deleting or Inserting data) ,because when we delete a number, Let’s see if we delete 2 index value then 3 number value will be shifted in the 2 number index. So because of data shifting data manipulation is slow. And on the other if I want to insert 50 in index 2 , after that index 2 value [previous value ] will be shifted into index 3. There is also shifting .

* Resizable , slow
* .for each loop, iterator
* array .size ()
* Size not fixed , ie size is changeable .because it is dynamic

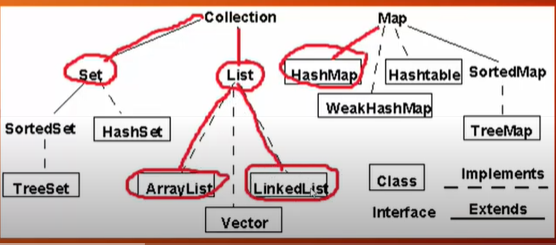
**Methods which are used in ArrayList**

**1.size() 5. clear() 9. set()**

**2.add() 6. isEmpty() 10.get()**

**3.remove() 7. Contains() 10.get() 4.removeAll() 8. indexOf()**

**11. equals() 12.addAll()**



ArrayList can contain duplicate elements – Example – we can put the same value in any index. Like 0 index value 30 and 1 index’s value 30.

Package ;

public class ArrayListDemo1{

public static void main (){

ArrayList < Integer> arraylist = new ArrayList<Interger>(); //ArrayList-Datatype-Name of ArrayList= new operator

Syso(“size = ” + number. size());}}

**Step 1 –** create an ArrayList with angular bracket and datatype which you want in the angular bracket

And then name of ArrayList with angular bracket and Constructor. and we can give any element or size in the parameter boxes . but in array it is not possible.

**Step 2-** Import the ArrayList , other wise it would be shown error.> under the util package.

**Step 3:** now we want to see the arrayList size, even we did not add any element in the parameter - new Array<Interger>(); we can give any size,bcoz it is not fixed.

**Step 4: if** we want to see this ArrayList Size , we have to give Syso. With the help of size method. And it will return the Size of ArrayList.

Now if I want to add the Element in the ArrayList

Package ;

public class ArrayListDemo1{

public static void main (){

ArrayList < Integer> arraylist = new Array<Interger>();

Syso(“size = ” + number. size()); // print Array Size

**// adding Element in one type of method**

number. Add(10); // 0 index

number. Add(20);// 1 index

number. Add(30); //2 index

number.add (0, Integer.SIZE) // adding Element in another type of method

// here 3 is index and Value 40 add

nember.add(3, 40)

//System. Out .print(number);

// Use for each loop

for(int x : number ){

System. Out .print(“ “ +x); // space are given using double quotation

System. Out .println(); // line separation

System. Out .println (“size = ” + number. size()); // print Array Size again

}

}

Output :

Before Element adding – O size will be printed

After adding Element 10,20,30,40, then size will be – 4

**There are various types/ way and method for ArayList value print**

**Types/ way**

**Step 1**—**System .out.println** (name of ArrayList)…number.add (0, Integer.SIZE), here 0 = index. I will give the index number in which our value will be added. This is the one way to print the value

Step 2: Different way value print in different way like using for each loop. So if we use for each loop then I have to comment this[ //System. Out .print(number);] and then use for each loop;-

a) like we print integer type value.

b) then variable name like x,

c) then colon [:] and then name of ArrayList , here ArayList name is number.

d) then system.out.print or println and print x = variable name = x

Step 3- Using Method of Iterator class we can do ArrayList

1. For loop comment
2. Taking help of Iterator class
3. First Take variable for Iterator class > then name of ArrayList then dot> then take iterator method. And this Iterator method will store into the itr all value of number Arraylist
4. Use while loop / take help from while loop -while(itr.hasNext()) { - value will stay till now [jotokhon value thakbe]
5. So we took a variable through iterator and then took a method using Arraylist and all value will be stored in itr.and itr will check whether there is any element or not. And itr will will be continued util it is finished .during continuation of the Loop, it will take print .so I have to give syso to print itr.next()

**Step 4 – We will use for each loop**

**Methods**

**We have a lot of methods to use in ArrayList—**

1. **add()**
2. **remove()**
3. **removeAll(**)

**Use Remove() ---**

1. **First write ArrayList name like here ArrayList name number**
2. **Dot and use remove()/ number.remove(O)**
3. **Put the index number Which index you are going to remove . let’s see 2nd position index you want to remove**
4. **After removing which value you will have contain. Then why syso (“after removing ArrayList Contains : ”+ number)**

**removeAll(**)

**a)First write ArrayList name like here ArrayList name= number**

**b) name of Arraylist which you want to remove all.**

**Dot and use removeAll()/ number.removeAll( variable name of the Arary)**

**Package ;**

public class ArrayListDemo1{

public static void main (){

ArrayList < Integer> arraylist = new Array<Interger>();

Syso(“size = ” + number. size()); // print Arraylist Size

// adding Element in one type of method

number. Add(10); // 0 index

number. Add(20);// 1 index

number. Add(30); //2 index

//here 3 is index and Value 40 add

nember.add(3, 40)

System. Out .print(“ArrayLsit contains : ”+ number);

System. Out .println();

System. Out .println (“size = ” + number. size()); // print Array Size again

**// removing elements**

number.remove(2);

System.out.println **(“after removing ArrayList Contains : ”+ number)**

// removing all elements

number.removeAll(number);

System.out.println **(“after removing all : ‘’ + number);**

**//Clear method**

number.clear();

System. Out .print(“After clear ArrayLsit contains : ”+ number);

**//**

**To check that the Arraylist is empty or not. For that we use isEmpty method .**

And then if array list is empty then it will print true and otherwise [not empty ] it will print false

Then we will use boolean datatype and use a variable to the value

boolean b =number.isEmpty();

System. Out .print(“ ArrayLsit empty : ”+ b);

Here it will print false because we have element in arraylist .

Now before using isEmptly if you use clear () method then it will return true. Because this araylist will be empty using clear (). So using clear() methods , ArayLsit will be Empty

number.clear();

boolean b =number.isEmpty();

System. Out .print(“ ArrayLsit empty : ”+ b);

// Contains ()—let’s see you are finding a Arraylist ,whether your elements contains or not. If it is contains then it will return true otherwise false. Lets see I am finding 30.

boolean contain = number.contains(30);

System. Out .print(“ 30 is in the List : ”+ contain);

// indexOf ()—let’s see you want to the index number of the specific value. it will return integer value . Let’s see I want to know the index number of 40.

int pos = number.indexOf (40);

System. Out .print(“ The index of 40 is : ”+ pos);

// Set () **. [Value set in index]**To use this method we can set the value into the index. – which position and who is the value to set .. this means replace the index of the value

number.set(3 , 50 ); //– which position and who is the value to set . index 3 , value [2 parameter ]

System. Out .print(“ After setting : ”+ number);

// Get() , **[Value get from index]** To use this method we can get any value from the specific index.

Which index value you want to see—like you want to see 0 index value. and it will return int type .

[one parameter] // this index will bring a value and this value will be stored

Int x =number.get( 0 );

System. Out .print(“ Index 0 : ”+ x);

Output 10

ArrayList<Integer> number1 = new ArrayList<> ();

ArrayList<Integer> number2 = new ArrayList<> ();

ArrayList<Integer> number3 = new ArrayList<> ();

number1.add(10);

number1.add(20);

number1.add(30);

number1.add(40);

**//ArrayList number1- value print**

**System.out.println(“number1 = ”+ number);**

number2.add(40);

number2.add(25);

number2.add(40);

number2.add(50);

**//ArrayList number2- value print**

**System.out.println(“number2 = ”+ number);**

number3.add(10);

number3.add(20);

number3.add(30);

number 3.add(40);

**Now Value assign of number 3**

**// we want to add some value of another number with number 3**

// addAll() To use addAll() we can add one arrayList with another ArrayList

number3.addAll(number1) **; // number1 value will add with number3**

**System.out.println(“Number3 = ”+ number 3);**

//equals() – To use equals() we can check equality with two ArrayList . whether they are equal or not. That means Whether number 1 is equal with number 2 or whatever else.

boolean check = number1.equals(number2) ; // check both are equal or not / exactly same or not

System.out.println(“Number1 == number 2 ”+ check);

//equals() – To use equals() we can check equality with two ArrayList . whether they are equal or not. That means Whether number 1 is equal with number 2 or whatever else.

boolean check = number1.equals(number2) ; // check both are equal or not / exactly same or not

System.out.println(“Number1 == number 2 ”+ check);

boolean check = number1.equals(number3) ; // check both are equal or not / exactly same or not

System.out.println(“Number1 == number 3 ”+ check);

);

}

**Sorting ArrayList**

Sorting means either ascending [small to big ]or Descending [ big to small ].

Package ;

public class ArrayListDemo2{

public static void main (){

ArrayList < Integer> number = new ArrayList<>();

// adding Random value

number. Add(10);

number. Add(-3);

number. Add(18);

number. Add(-11);

number. Add(90);

number. Add(1);

**System.out.println(“Before sorting : ”+ number);**

**// before sorting value will be shown same as displayed above.**

**Output -10,-3,18,-11,90,1**

**For ascending we will use class of collection --- collections.sort() and you want to sort number ArrayList**

Collections.sort(number);

System.out.println(“After sorting in ascending : ”+ number);

**For Descending -- If you reverse the ascending order then it will be descending -**

So This method |collections.reverseOrder()| will reverse the number of ascending ArrayList

collections.sort(number,collections.reverseOrder());

System.out.println(“After sorting in Descending : ”+ number);

collections.sort(number,collections.reverseOrder()); this lines means collections.sort(number) -ascending the number and then collections.reverseOrder());. Reverse the ascending number

i.e descending the number.

* We can do it when we will make the code first for ascending .
* But if we do first descending order then we have to write the following code---

collections.sort(number,collections.reverseOrder());

System.out.println(“After sorting in Descending : ”+ number);

………………………………………………………………….

**Note**

**Array and List store data based on Index.**

LinkedList



**Features of LinkedList**

1. There are nodes in double linked list
2.  in the middle that means 20 is value . so Value in the middle
3. Left site that means X is the address of nodes. Previous nodes’s address
4. Next nodes address at the right site that means 200 .
5. In each and every node has address . here the roughly address is 100,200,300,400 . it always address the next node.
6. Here x in first column 200 at the right , in the second column 300 in the second column and 400 is the third column , they are addresses to the next node.
7. Head is first , before head there is nothing that means null [X] and as the same for the tail .
8. In linked list they have relation with each other

* LinkedList class uses doubly linked list to store the elements
* Manipulating of data is fast (deleting or Inserting ). Because there is no data shifting problem. If you want to delete a node then another nodes will be joined automatically . there is no replacement here.
* It also contain duplicate elements.

**Package ; .. we use wrapper class in angular brackets**

public class LinkedListDemo1{

public static void main (){

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

**// Add Elements** 1) add( String e) that is single parameter

countryNames.add(“Bangladesh”);

countryNames.add(“India”);

countryNames.add(“Pakistan”);

countryNames.add(“Nepal”);

countryNames.add(“5, “Japan”);

// For Print

For(String country : countryNames){ // we will do in String datatype that’s why String , give one variable name and you are going to use collection or LinkedList / which linked list you want to use

for(String country : countryNames) // for each loop

{

System.out.println(country);

}

//For LinkedList size

System.out.println(“Size of the LinkedList : ” + countryNames.size());

// if you want to see first country which is the first / Zero index

System.out.println(“First element : ” + countryNames.getFirst() );

// if you want to see Last country which is the Last / Zero index

System.out.println(“Last element : ” + countryNames.getLast() );

// LinkedList Clear

countryNames.clear();

System.out.println(countryNames);

**// Other methods which we can use in the LinkedList class.**

countryNames.addFirst(“USA”); // which country you want to show first

countryNames.addLast(“ India ”); // which country you want to show Last

countryNames.remove(“ USA ”); // which country you want to remove

countryNames.remove( 3); // which index value you want to remove

countryNames.removeFirst(“ USA ”); // you want to remove USA because it was first

countryNames.removeLast(“ India ”); // you want to remove USA because it was Last

**Methods of adding**

Note - For adding there are two methods

1) add( String e)

2) add(int i , String e)

In the second method of Elements adding 🡪add(int i , String e) a) two parameters – First one index number and add the element. Let’s see you want to add index 5. And element in that index japan.

3)addFirst()  
4) addLast()

//First way to take print

System.out.println (countryName); // print

//First way to take print using for loop

// all country in countryNames and it (countryNames) will pass one by one country to the country[variable name]

for (String country : countryNames ){ //for each loop

System.out.println (countryName); // print

……………….

**To see Linked List Size –** to use Syso. And size methiod

System.out.println (“Size of the LinkedList : ” + countryNames.size()); // print

……………….

}

// adding Element in one type of method

number. Add(10); // 0 index

number. Add(20);// 1 index

number. Add(30); //2 index

//here 3 is index and Value 40 add

nember.add(3, 40)

System. Out .print(“ArrayLsit contains : ”+ number);

System. Out .println();

System. Out .println (“size = ” + number. size()); // print Array Size again

**// removing elements**

number.remove(2);

System.out.println **(“after removing ArrayList Contains : ”+ number)**

// removing all elements

number.removeAll(number);

System.out.println **(“after removing all : ‘’ + number);**

**//Clear method**

number.clear();

System. Out .print(“After clear ArrayLsit contains : ”+ number);

**//**

**To check that the Arraylist is empty or not. For that we use isEmpty method .**

And then if array list is empty then it will print true and otherwise [not empty ] it will print false

Then we will use boolean datatype and use a variable to the value

**HashMap in Collection**

* HashMap stores data based on Key value. it means it should have a key and under the key a value should be .Ie HashMap works with Key Value.
* Methods which are used in Map Interface . put, get . here put means data insert and get means data get **[data udder** ]

// Like We think a Customer . A customer has ID and we can store a value in the ID

Package hashmap\_demo;

Public class HashMaptDemo {

public static void main(String [] args ){

HashMap<Integer , String> customerName = new HashMap<Integer , String > () ;

**// insert data / put data**

**customerName.put(101, “Robi”);**

**customerName.put(102, “Robin”);**

**customerName.put(103, “Robina”);**

**customerName.put(104, “Sumabi”);**

**// get data**