

Array, ArrayList and Linked List

What you'll Learn

What is Array and Arraylist?

Creating Array and Arraylist

Displaying the elements in an Array and Arraylist

Searching & Sorting elements in an Array and Arraylist

adding & removing elements,Reverse order of elements

Linked list



What is Array

Normally, an array is a collection of similar type of elements which have a contiguous memory location.

Array is an object which contains elements of a similar data type.

The elements of an array are stored in a contiguous memory location.

Array is a data structure where we store similar elements.

We can store only a fixed set of elements in a Java array.

Array is index-based, the first element of the array is stored at the 0th index, 2nd element is stored on 1st index and so on.



Fixed size. Can not be increased or decreased once declared.

Can store a single type of primitives only.

Arrays does not have add or remove methods.

To delete an element in an array we need to traverse through out the array so this will reduce performance.



Types of Array

Single Dimensional Array

A one-dimensional array (or single dimension array) is a type of linear array.

Accessing its elements involves a single subscript which can either represent a row or column index.

Multidimensional Array

A multi-dimensional array is an array with more than one level or dimension.

For example, a 2D array, or two-dimensional array, is an array of arrays, meaning it is a matrix of rows and columns (think of a table).



Declaration of an Array

The general form of a one-dimensional array declaration is:

```
type var-name[]; OR type[] var-name;
```

An array declaration has two components: the type and the name.

type declares the element type of the array. The element type determines the data type of each element that comprises the array.

name is the array name given by the user.

Example:

```
int intArray[]; or  
int[] intArray;
```



Initialization of an Array

When an array is declared, only a reference of array is created.

To actually create or give memory to array, you create an array like this:

```
type var-name = new type [size];
```

type specifies the type of data being allocated.

size specifies the number of elements in the array.

var-name is the name of array variable that is linked to the array.

Example:

```
int Array = new int[20];
```



Insert Element - Array

```
import java.util.*;
class Main {
    public static void main(String[] args) {
        int len, p, ele;
        Scanner sc = new Scanner(System.in);
        System.out.print("Enter length of an array:");
        len = sc.nextInt();
        int arr[] = new int[len+1];
        System.out.println("Enter "+len+" elements:");
        for(int i = 0; i < len; i++){
            arr[i] = sc.nextInt(); }
        System.out.print("Enter the element which you want to insert:");
        ele = sc.nextInt();
        arr[len] = ele;
        System.out.print("After inserting : ");
        for(int i = 0; i < len; i++){
            System.out.print(arr[i]+","); }
        System.out.print(arr[len]);
    }
}
```


Increasing Size - Array

An array cannot be resized dynamically in Java.

One approach is to use `java.util.ArrayList`(or `java.util.Vector`) instead of a native array.

Another approach is to re-allocate an array with a different size and copy the contents of the old array to the new array.



Example

```
public class Main{
    public static void main(String args[]){
        changeArraySize();
    }
    private static void changeArraySize(){
        int[] intArray = new int[5];
        System.out.println("Before change: size(): " + intArray.length);
        intArray = java.util.Arrays.copyOf(intArray, intArray.length * 2);
        System.out.println("After change: size(): " + intArray.length);
    }
}
```

Addition of Two Array

```
class Main{
public static void main(String args[]){
    int a[][]={{1,3,4},{3,4,5}};
    int b[][]={{1,3,4},{3,4,5}};
    int c[][]=new int[2][3];
    for(int i=0;i<2;i++){
        for(int j=0;j<3;j++){
            c[i][j]=a[i][j]+b[i][j];
            System.out.print(c[i][j]+" ");
        }
        System.out.println();
    }
}
```

Find Total Mark and Percentage

```
import java.util.Scanner;
public class Main {
    public static void main(String[]
args) {
        int n, total = 0, percentage;
        Scanner s = new
Scanner(System.in);
        System.out.print("Enter no. of
subject:");
        n = s.nextInt();
        int marks[] = new int[n];
```

```
System.out.println("Enter marks out of
100:");
        for(int i = 0; i < n; i++) {
            marks[i] = s.nextInt();
            total = total + marks[i];
        }
        percentage = total / n;
        System.out.println("Sum:"+total);

System.out.println("Percentage:"+percen
tage);
    }
}
```

Arraylist

Java ArrayList class uses a dynamic array

It inherits AbstractList class and implements List interface

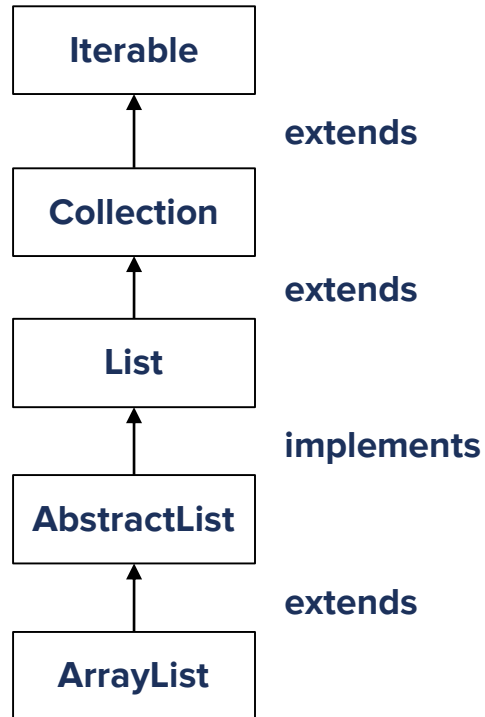
contain duplicate elements

maintains insertion order

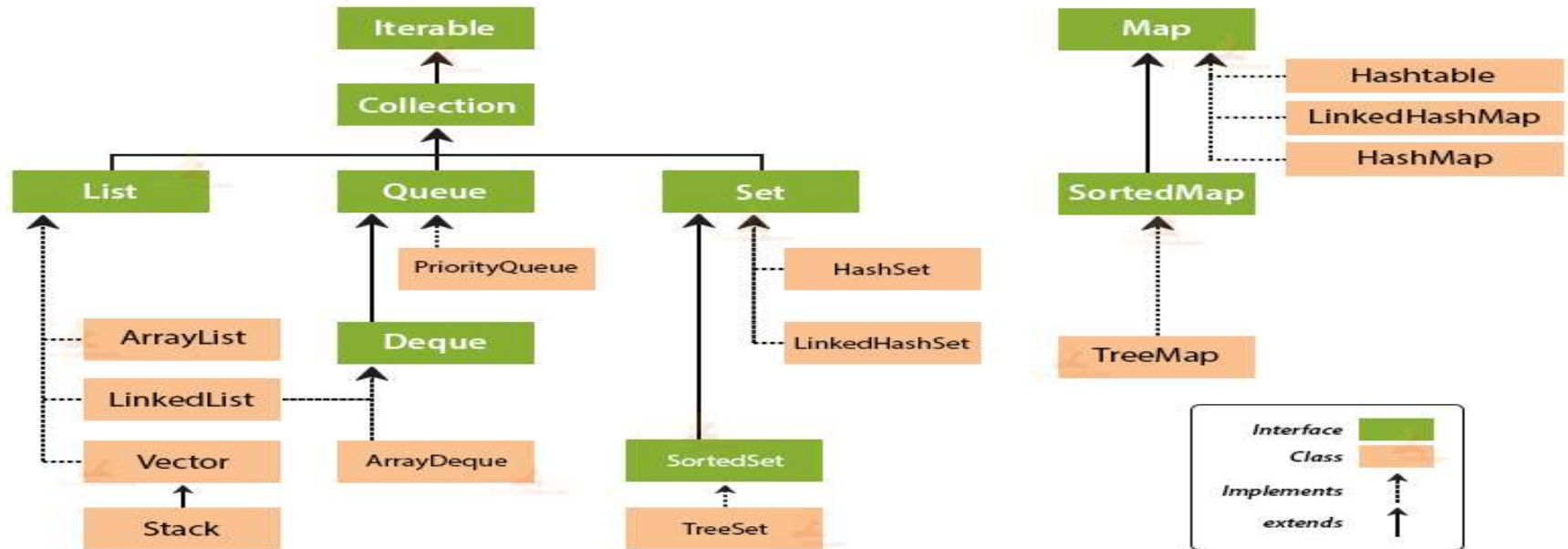
random access



Hierarchy



Collection Framework Hierarchy in Java



Arraylist Object Creation

```
import ArrayList
```

```
import java.util.ArrayList;
```

Create Object

```
ArrayList<Datatype> Ethnus = new ArrayList<Datatype>();
```



ArrayList - Constructor

ArrayList()

Constructs an empty ArrayList

ArrayList(Collection c)

containing the elements of the specified Collection

ArrayList(int initialCapacity)

empty ArrayList with the specified initial capacity



Arraylist - Methods

Methods	Return
add (int index, Object element) - Inserts the specified element at the specified position	void
add (Object o) - Appends the specified element	boolean
addAll (Collection c) - Appends all of the elements in the specified Collection	boolean
addAll (int index, Collection c) - Inserts all of the elements in the specified Collection	boolean
clear () - Removes all of the elements	void

ArrayList - Methods


Methods	Return
remove (int index) - Removes the element at the specified position	object
set (int index, Object element) - Replaces the element at the specified position in this ArrayList with the specified element	object
size () - Returns the number of components	int
toArray () - Returns an array containing all of the elements	object
toArray (Object[] a) - Returns an array containing all of the elements	object
trimToSize () - Trims the capacity of this ArrayList to be the ArrayList's current size	void

ArrayList - Methods

Methods	Return
contains (Object elem) - Returns true if this ArrayList contains the specified element	boolean
ensureCapacity (int minCapacity) - Increases the capacity	void
get (int index) - Returns the element at the specified position	object
indexOf (Object elem) - Searches for the first occurrence of the given argument	int
isEmpty () - Tests if this ArrayList has no components	boolean
lastIndexOf (Object elem) - Returns the index of the last occurrence of the specified object	int


add() and remove() Method

```
class Main{  
    public static void main ( String[] args){  
        ArrayList<String> Ethnus = new ArrayList<String>();  
        Ethnus.add("Codemithra.com");  
        Ethnus.add("eguru.ooo");  
        Ethnus.add("inpiq.com");  
        System.out.println("Our Product: " + Ethnus);  
        Ethnus.remove(1);  
        System.out.println("Our Product: " + Ethnus);  
    }  
}
```



Iterator

```
import java.util.*;
class Main{
    public static void main ( String[] args){
        ArrayList<String> Ethnus = new ArrayList<String>();
        Ethnus.add("Codemithra.com");
        Ethnus.add("eguru.ooo");
        Ethnus.add("inpiq.com");
        Iterator<String> iter = Ethnus.iterator();
        while(iter.hasNext()){
            System.out.print(iter.next()+" ");
        }
    }
}
```



isEmpty()

```
import java.util.ArrayList;

class Main{

    public static void main ( String[] args){

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

        System.out.println(Ethnus.isEmpty());

        Ethnus.add("Codemithra.com");

        Ethnus.add("eguru.ooo");

        Ethnus.add("inpiq.com");

        System.out.println(Ethnus.isEmpty());

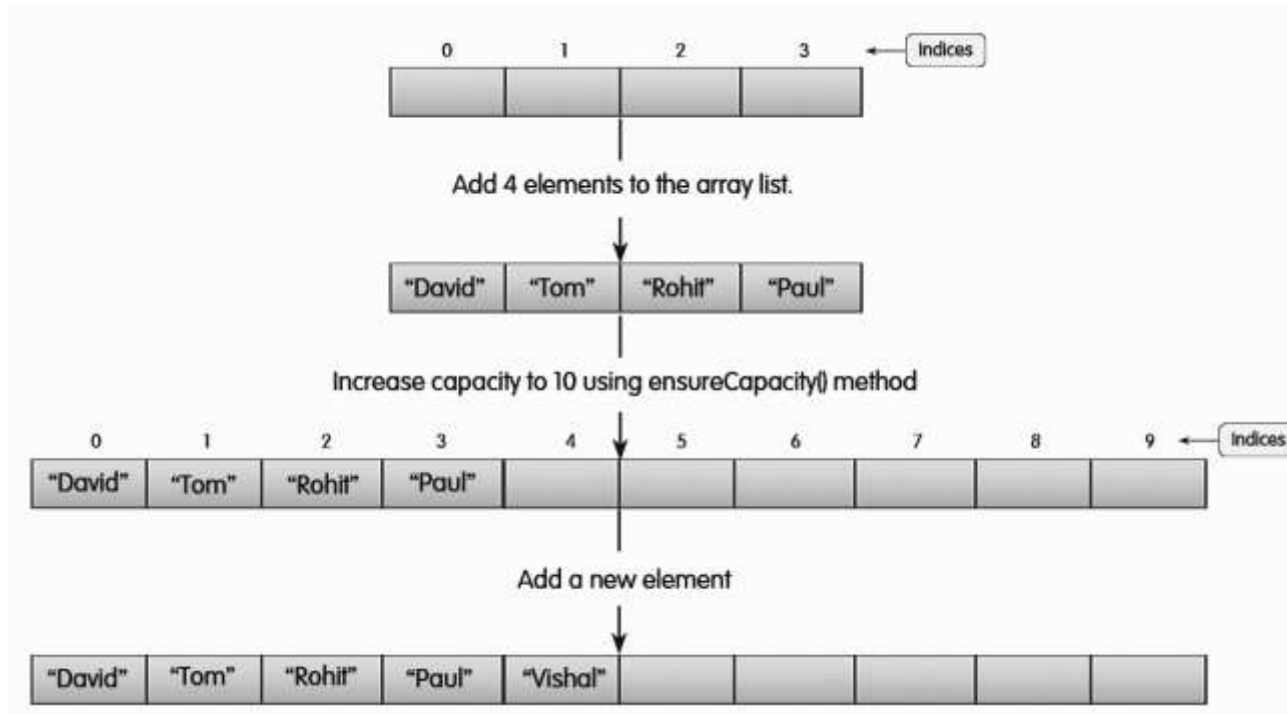
    }
```

ensureCapacity()

The `ensureCapacity()` method is used to increase the capacity of this `ArrayList` instance if necessary, to ensure that it can hold at least the number of elements which is not smaller than the specified size.




ensureCapacity()



ensureCapacity()

```
import java.util.*;
public class Main {
    public static void main(String[] args) {
        ArrayList<String> StudentList = new ArrayList<String>(4);
        StudentList.add("David");
        StudentList.add("Tom");
        StudentList.add("Rohit");
        StudentList.add("Paul");
        StudentList.ensureCapacity(10);
        StudentList.add("Vishal");
        for (String s: StudentList) {
            System.out.println(s);
        }
    }
}
```



What you'll Learn

What is linkedlist?

About the underlying data structure - doubly linked list

constructors, methods present in linked list

Implementation class

Coding examples



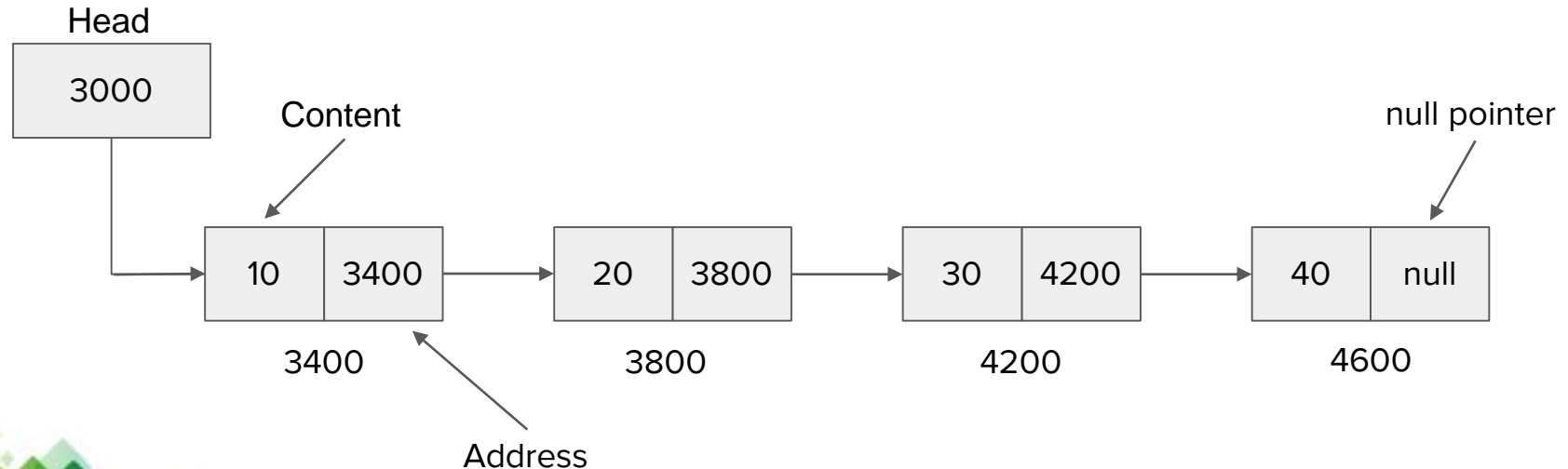
Linked List

Linear data structures

Not stored in contiguous locations

Separate object with a data part and address part

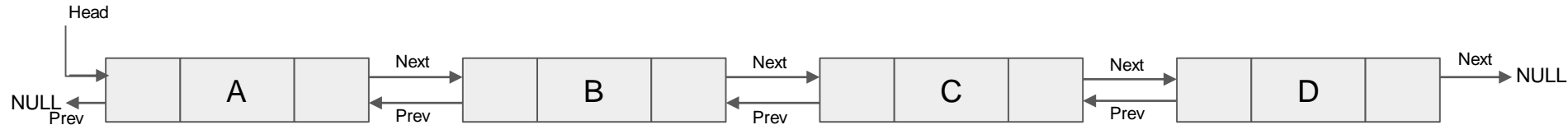
Start from the head and follow through the link



Double Linked List

Doubly-linked list implementation of the List and Deque interfaces

Implements all optional list operations, and permits all elements (including null)



Linked List Object Creation

import LinkedList

```
import java.util.LinkedList;
```

Create Object

```
LinkedList list = new LinkedList();
```



Linked List - Constructor

LinkedList()

Constructs an empty list

LinkedList(Collection<E> c)

Constructs a list containing the elements of the specified collection, in the order they are returned by the collection's iterator



Linked List Methods

METHOD	Return
add(E e) - Appends the specified element to the end of this list	Boolean
add(int index, E element) - Inserts the specified element at the specified position in this list	void
addAll(Collection< E> c) - Appends all of the elements in the specified collection	Boolean
addAll(int index, Collection<E> c) - Inserts all of the elements in the specified collection	Boolean
addFirst(E e) - Inserts the specified element at the beginning of this list	void
addLast(E e) - Appends the specified element to the end of this list	void
clear() - Removes all of the elements from this list	void
clone() - Returns a shallow copy of this LinkedList	Object
contains(Object o) - Returns true if this list contains the specified element	Boolean

Linked List Methods


METHOD	Return
get (int index) - Returns the element at the specified position in this list	E
getFirst () - Returns the first element in this list	E
getLast () - Returns the last element in this list	E
indexOf (Object o) - Returns the index of the first occurrence	int
lastIndexOf (Object o) - Returns the index of the last occurrence	int
pop () - Pops an element from the stack represented by this list	E
push (E e) - Pushes an element onto the stack represented by this list	E
remove (int index) - Removes the element at the specified position in this list	E
size () - Returns the number of elements in this list	int

add() - Example

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        System.out.println(Ethnus);
    }
}
```


add() and remove - Example

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        System.out.println("Etnus Product: "+ Ethnus);
        Ethnus.remove(0);
        System.out.println("Etnus Product: "+ Ethnus);
    }
}
```



removeFirst() - Example

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        System.out.println("Etnus Product: "+ Ethnus);
        Ethnus.removeFirst();
        System.out.println("Etnus Product: "+ Ethnus);
    }
}
```




contains()

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        System.out.println("Etnus Product: "+ Ethnus);
        boolean status = Ethnus.contains("codemithra.com");
        System.out.println(status);
    }
}
```

size()

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        System.out.println("Etnus Product: "+ Ethnus);
        int len = Ethnus.size();
        System.out.println("length "+ len);
    }
}
```



addAll()

```
import java.util.LinkedList;
class Main {
    public static void main(String[] args) {
        LinkedList<String> Ethnus = new LinkedList<String>();
        Ethnus.add("codemithra.com");
        Ethnus.add("inpiq.com");
        Ethnus.add("eguru.ooo");
        LinkedList<String> Ethnus1 = new LinkedList<String>();
        Ethnus1.add("aptimithra.com");
        Ethnus1.addAll(Ethnus);
        System.out.println("Ethnus Product: "+ Ethnus);
        System.out.println("Ethnus Product: "+ Ethnus1);
    }
}
```

THANK YOU

