**Collection Framework – Part\_18**

* **Arrays:**

Arrays class is a utility class to define several utility methods for array objects.

* **Sorting elements of Array:**

Arrays class defines the following sort methods to sort elements of primitive and object type arrays.

public static void sort(primitive[] p);

To sort according to natural sorting order.

public static void sort(Object[] o);

To sort according to natural sorting order.

public static void sort(Object[] o, Comparator c);

To sort according to customized sorting order.

* **Example:**

import java.util.Arrays;

import java.util.Comparator;

class ArraysSortDemo{

public static void main(String[] args){

int[] a = {10,5,20,11,6};

System.out.println(“Primitive array before sorting”);

for(int a1: a){

System.out.println(a1);

}

Arrays.sort(a);

System.out.println(“Primitive Array after sorting”);

for(int a1: a){

System.out.println(a1); // 5,6,10,11,20

}

String[] s = {“A”, “Z”, “B”};

System.out.println(“Object Array before sorting”); // A, Z, B

for(String s1: s){

System.out.println(s1);

}

Arrays.sort(s);

Arrays.sort(s, new MyComparator());

System.out.println(“Object Array after sorting by comparator”);

for(String s1: s){

System.out.println(s1); // Z, B, A

}

}

}

class MyComparator implements Comparator{

public int compare(Object obj1, Object obj2){

String s1 = obj1.toString();

String s2 = obj2.toString();

return s2.compartTo(s1);

}

}

Note:

We can sort primitive arrays only based on default natural sorting order. Whereas, we can sort object arrays either based on default natural sorting order or based on customized sorting order.

* **Searching the elements of Array:**

Arrays class defines the following binary search methods.

public static int binarySearch(primitive[] p, primitive target);

public static int binarySearch(Object[] a, Object target);

public static int binarySearch(Object[] a, Object target, Comparator c);

Note:

All rules of Arrays class binarySearch() method are exactly same as Collections class binarySearch() methods.

* **Example:**

import java.util.\*;

import static java.util.Arrays.\*;

class ArraySearchDemo{

public static void main(String[] args){

int[] a = {10, 5, 20, 11, 6};

Arrays.sort(a);// sort by natural order

System.out.prinltn(Arrays.binarySearch(a, 6)); // 1

System.out.prinltn(Arrays.binarySearch(a, 4)); // -5

String[] s = {“A”, “Z”, “B”};

Arrays.sort(s);

System.out.println(binarySearch(s, “Z”)); //2

System.out.println(binarySearch(s, “S”)); //-3

Arrays.sort(s, new MyComparator());

System.out.println(binarySearch(s, “Z”, new MyComparator()); //0

System.out.println(binarySearch(s, “S”, new MyComparator()); // -2

System.out.println(binarySearch(s, “N”)); //unpredictable result.

}

}

class MyCompartor implements Comparator{

public int compare(Object obj1, Object obj2){

String s1 = obj1.toString();

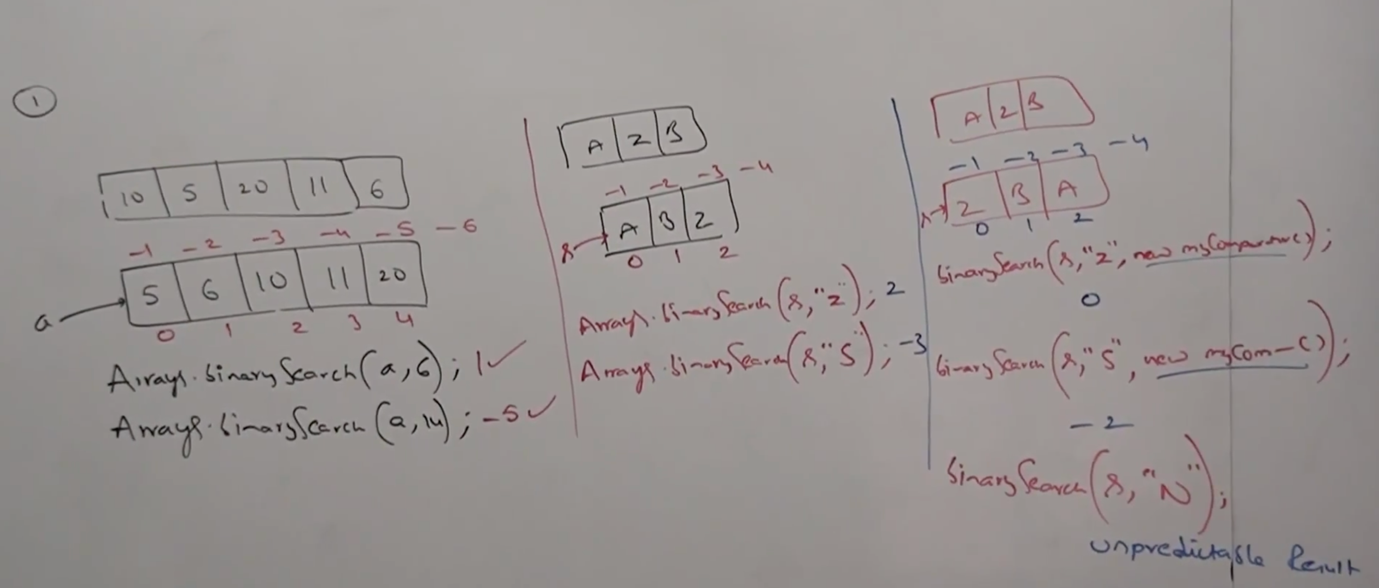
String s2 = obj2.toString();

return s2.compareTo(s1);

}

}

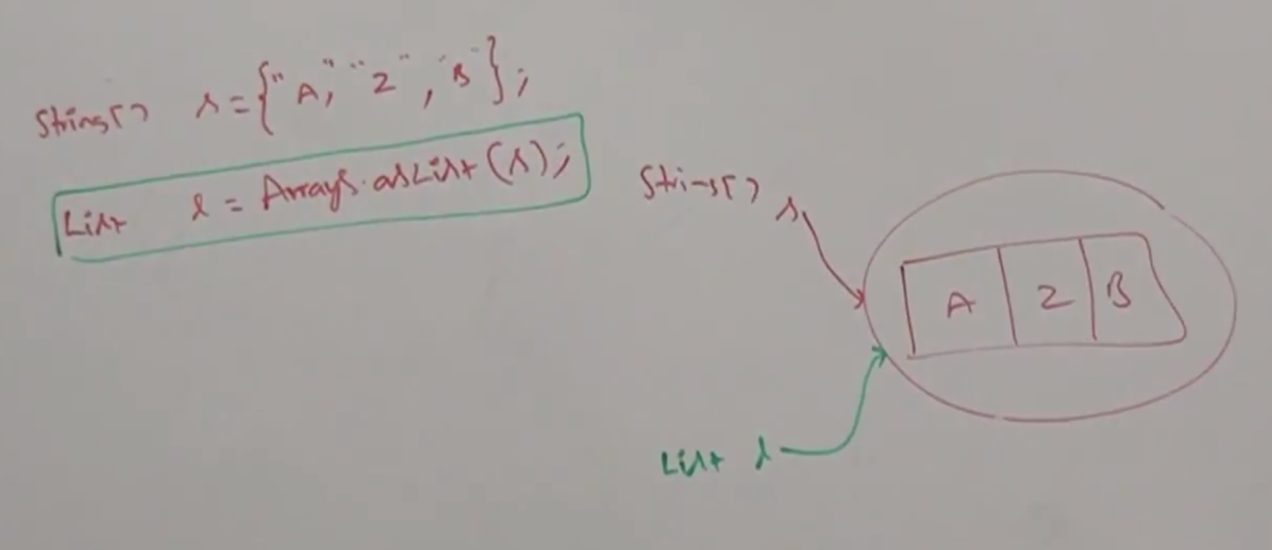
**Diagram:**

****

* **Conversion of Array to List:**

public static List asList(Object[] o);

Strictly speaking this method won’t create an independent List object. For the existing Array, we are getting List view.



By using array reference if we perform any change automatically that change will be reflected to the List. Similarly, by using List reference if we perform any change that change will be reflected automatically to the array.

By using List reference, we can’t perform any operation which varies the size, otherwise we will get runtime exception saying UnsupportedOperationException.

l.add(“M”);

l.remove(1);

RE: UnsupportedOperationException

l.set(1, “N”); // valid

By using List reference, we are not allowed to replace with heterogeneous objects, otherwise we will get runtime exception saying: ArrayStoreException

l.set(1, new Integer(10)); //RE: ArrayStoreException

* **Example:**

import java.util.\*;

class ArraysAsListDemo{

public static void main(String[] args){

String[] s = {“A”, “Z”, “B”};

List l = Arrays.asList(s);

System.out.println(l);

S[0] = “K”;

l.set(1, “L”);

for(String s1: s)

System.out.println(s1); //K,L,B

l.add(“durga”); //UsOE

l.remove(2); // UsOE

l.set(1, new Integer(10)); // ASE

}

}