

The image consists of two screenshots of an Eclipse IDE. The top screenshot shows a Java file named 'ClassA.java' with the following code:

```
3 import java.util.ArrayList;
4
5 public class ClassA
6 {
7     void meth1()
8     {
9         System.out.println("Implementing ArrayList\n");
10
11         ArrayList al=new ArrayList();
12
13         al.add(10); // Insertion order is
14         al.add("Java"); // Heterogeneous data is
15         al.add(null); // null value is
16         al.add(10); // Duplicates are ALLOWED
17         al.add('A'); // It is available from Java 1.2v
18         al.add(true); // Its default capacity is 10
19         al.add(1); // Its increases by HALF
20         al.add(99); // It is NOT Synchronized
21
22         System.out.println(al);
23     }
24     public static void main(String[] args)
25     {
26         ClassA aobj=new ClassA();
27         aobj.meth1();
28     }
29 }
```

The console output on the right shows:

```
<terminated> ClassA [Java Application] F:\Softwares\ eclipse-jee-2023-06-R-win32-x86_64
Implementing ArrayList

[10, Java, null, 10, A, true, 1, 99]
```

The bottom screenshot shows the same code, but with the line numbers 13 through 23 highlighted in blue. The console output is the same.

From java1.5 version generic which allows homogenous data.

The main use of generics is to avoid type safety problems

```
ArrayList<Integer> al=new ArrayList<Integer>();
```

```

1 package com.pack1;
2
3 import java.util.ArrayList;
4
5 public class ClassA
6 {
7     void meth1()
8     {
9         System.out.println("Implementing ArrayList\n");
10
11         ArrayList<Integer> al=new ArrayList<Integer>();
12
13         al.add(10); // Insertion order is maintained
14         al.add(85); // Heterogeneous data is allowed
15         al.add(null); // null value is allowed
16         al.add(10); // Duplicates are ALLOWED
17         al.add(45); // It is available from Java 1.2v
18         al.add(66); // Its default capacity is 10
19         al.add(1); // Its increases by HALF
20         al.add(99); // It is NOT Synchronized
21
22         System.out.println(al);
23
24         System.out.println("\nsize() : "+al.size());
25         System.out.println("get() : "+al.get(1)); // Jav
26
27         System.out.println("\nReteriving the elements i
28         for(int i=0;i<=al.size()-1;i++)
29         {
30             System.out.print(al.get(i)+" ");
31         }
32         System.out.println();
33
34         for(int i=al.size()-1;i>=0;i--)
35         {
36             System.out.print(al.get(i)+" ");
37         }
38
39         System.out.println("\n\nReteriving the elements by using for-each loop");
40         for(Object o:al) // for(Integer o:al) // for(int o:al)
41         {
42             System.out.print(o+" ");
43         }
44         System.out.println("\n\nReteriving the elements by using Iterator Interface");
45         Iterator<Integer> i=al.iterator(); //([10, 85, null, 10, 45, 66, 1, 99]
46         while(i.hasNext())
47         {
48             System.out.print(i.next()+" ");
49         }
50
51     }
52     public static void main(String[] args)
53     {
54         ClassA aobj=new ClassA();
55         aobj.meth1();
56     }
57 }
58
59
60

```

```

<terminated> ClassA [Java Application] [F:\Software\ eclipse-jee-2023-06-18-win32-x86_64\ eclipse\plugins\org.eclipse.jst.j2ee.ui\hotspot\je.full.win
Implementing ArrayList

[10, 85, null, 10, 45, 66, 1, 99]

size() : 8
get() : 85

Reteriving the elements in BOTH forward & in backward directions by us
10 85 null 10 45 66 1 99
99 1 66 45 10 null 85 10

Reteriving the elements by using for-each loop
10 85 Exception in thread "main" java.lang.NullPointerException: Cannot
at Training/com.pack1.ClassA.meth1(ClassA.java:39)
at Training/com.pack1.ClassA.main(ClassA.java:47)

```

hasNext() is going to return Boolean value true or false

[12, 13, #, %] after cursor element is present returns TRUE

[12, 13, #, %] after cursor no-element is present returns False

Next() is going to return the element which is present after cursor and cursor will moves to next element.

```

40     for(Object o:al) // for(Integer o:al) // for(int o:al)
41     {
42         System.out.print(o+" ");
43     }
44     System.out.println("\n\nReteriving the elements by using Iterator Interface");
45     Iterator<Integer> i=al.iterator(); // [10, 85, null, 10, 45, 66, 1, 99]
46     while(i.hasNext())
47     {
48         System.out.print(i.next()+" ");
49     }
50
51     System.out.println("\n\n=====METHODS=====");
52
53     System.out.println("isEmpty() : "+al.isEmpty());
54     al.clear();
55     System.out.println("isEmpty() : "+al.isEmpty());
56 }
57
58* public static void main(String[] args)
59 {
60     ClassA aobj=new ClassA();
61     aobj.meth1();
62 }
63 }
64
65
66
67

```

```

<terminated> ClassA [Java Application] F:\Software\ eclipse-jee-2023-06
Implementing ArrayList

[10, 85, null, 10, 45, 66, 1, 99]

size() : 8
get() : 85

Reteriving the elements in BOTH fo
10 85 null 10 45 66 1 99
99 1 66 45 10 null 85 10

Reteriving the elements by using f
10 85 null 10 45 66 1 99

Reteriving the elements by using I
10 85 null 10 45 66 1 99

=====METHODS=====
isEmpty() : false
isEmpty() : true

```

```

41     {
42         System.out.print(o+" ");
43     }
44     System.out.println("\n\nReteriving the elements by using Iterator Interface");
45     Iterator<Integer> i=al.iterator(); // [10, 85, null, 10, 45, 66, 1, 99]
46     while(i.hasNext())
47     {
48         System.out.print(i.next()+" ");
49     }
50
51     System.out.println("\n\n=====METHODS=====");
52
53     System.out.println("isEmpty() : "+al.isEmpty());
54     //al.clear();
55     //System.out.println("isEmpty() : "+al.isEmpty());
56     System.out.println("contains() : "+al.contains(10));
57 }
58
59* public static void main(String[] args)
60 {
61     ClassA aobj=new ClassA();
62     aobj.meth1();
63 }
64 }
65
66
67

```

```

Implementing ArrayList

[10, 85, null, 10, 45, 66, 1, 99]

size() : 8
get() : 85

Reteriving the elements in BOTH fo
10 85 null 10 45 66 1 99
99 1 66 45 10 null 85 10

Reteriving the elements by using f
10 85 null 10 45 66 1 99

Reteriving the elements by using I
10 85 null 10 45 66 1 99

=====METHODS=====
isEmpty() : false
contains() : true

```

```

40     for(Object o:al) // for(Integer o:al) // for(int o:al)
41     {
42         System.out.print(o+" ");
43     }
44     System.out.println("\n\nReteriving the elements by using Iterator Interface");
45     Iterator<Integer> i=al.iterator(); // [10, 85, null, 10, 45, 66, 1, 99]
46     while(i.hasNext())
47     {
48         System.out.print(i.next()+" ");
49     }
50
51     System.out.println("\n\n=====METHODS=====");
52
53     System.out.println("isEmpty() : "+al.isEmpty());
54     //al.clear();
55     //System.out.println("isEmpty() : "+al.isEmpty());
56     System.out.println("contains() : "+al.contains(10));
57
58     ArrayList<Integer> al2=new ArrayList<Integer>();
59     al2.add(10);
60     al2.add(99);
61     System.out.println("containsAll() : "+al.containsAll(al2));
62
63 }
64
65* public static void main(String[] args)
66 {
67     ClassA aobj=new ClassA();
68     aobj.meth1();
69 }
70 }
71

```

```

<terminated> ClassA [Java Application] F:\Software\ eclipse-jee-2023-06
Implementing ArrayList

[10, 85, null, 10, 45, 66, 1, 99]

size() : 8
get() : 85

Reteriving the elements in BOTH fo
10 85 null 10 45 66 1 99
99 1 66 45 10 null 85 10

Reteriving the elements by using f
10 85 null 10 45 66 1 99

Reteriving the elements by using I
10 85 null 10 45 66 1 99

=====METHODS=====
isEmpty() : false
contains() : true
containsAll() : true

```

```

51 System.out.println("\n\n=====METHODS=====");
52
53 System.out.println("isEmpty() : "+a1.isEmpty());
54 //a1.clear();
55 //System.out.println("isEmpty() : "+a1.isEmpty());
56 System.out.println("contains() : "+a1.contains(10));
57
58 ArrayList<Integer> a12=new ArrayList<Integer>();
59 a12.add(10);
60 a12.add(100);
61 System.out.println("containsAll() : "+a1.containsAll(a12));
62
63 System.out.println(a1);
64 a1.addAll(a12);
65 System.out.println(a1);
66 a1.add(2,95);
67 System.out.println(a1);
68 a1.set(1, 80);
69 System.out.println(a1);
70

```

Reteriving the elements by using for-each loop
10 85 null 10 45 66 1 99

Reteriving the elements by using Iterator Interface
10 85 null 10 45 66 1 99

```

=====METHODS=====
isEmpty() : false
contains() : true
containsAll() : false
[10, 85, null, 10, 45, 66, 1, 99]
[10, 85, null, 10, 45, 66, 1, 99, 10, 100]
[10, 85, 95, null, 10, 45, 66, 1, 99, 10, 100]
[10, 80, 95, null, 10, 45, 66, 1, 99, 10, 100]

```

```

71
72
73
74
75
76 public static void main(String[] args)
77 {
78     ClassA aobj=new ClassA();
79     aobj.meth1();
80 }
81 }

```

size() : 8
get() : 85

Reteriving the elements in BOTH forward & in backward dir
10 85 null 10 45 66 1 99
99 1 66 45 10 null 85 10

Reteriving the elements by using for-each loop
10 85 null 10 45 66 1 99

Reteriving the elements by using Iterator Interface
10 85 null 10 45 66 1 99

```

=====METHODS=====
isEmpty() : false
contains() : true
containsAll() : false
[10, 85, null, 10, 45, 66, 1, 99]
[10, 85, null, 10, 45, 66, 1, 99, 10, 100]
[10, 85, 95, null, 10, 45, 66, 1, 99, 10, 100]
[10, 80, 95, null, 10, 45, 66, 1, 99, 10, 100]
Exception in thread "main" java.lang.IndexOutOfBoundsException
    at java.base/jdk.internal.util.Preconditions.outOfBounds(Preconditions.java:65)
    at java.base/jdk.internal.util.Preconditions.outOfBoundsCheckIndex(Preconditions.java:75)
    at java.base/jdk.internal.util.Preconditions.checkIndex(Preconditions.java:91)
    at java.base/java.util.Objects.checkIndex(Objects.java:369)
    at java.base/java.util.ArrayList.remove(ArrayList.java:503)
    at Training/com.pack1.ClassA.meth1(ClassA.java:71)
    at Training/com.pack1.ClassA.main(ClassA.java:79)

```

Why because the compiler will think it has index position
In a1 we don't have 100th index position.


```

1 package com.pack1;
2
3 import java.util.ArrayList;
4 import java.util.Iterator;
5
6 public class ClassA
7 {
8     void meth1()
9     {
10         System.out.println("Implementing ArrayList\n");
11
12         ArrayList<Integer> al=new ArrayList<Integer>();
13
14         al.add(10); // Insertion order is maintained
15         al.add(85); // Heterogeneous data is allowed
16         al.add(null); // null value is allowed
17         al.add(10); // Duplicates are ALLOWED
18         al.add(45); // It is available from Java 1.2v
19         al.add(66); // Its default capacity is 10
20         al.add(1); // Its increases by HALF
21         al.add(99); // It is NOT Synchronized
22
23         System.out.println(al);
24
25         System.out.println("\nsize() : "+al.size());
26         System.out.println("get() : "+al.get(1)); // Java
27
28         System.out.println("\nReteriving the elements in BOTH forward & in backward directions by using for- loop");
29
30         for(int i=0;i<=al.size()-1;i++)
31         {
32             System.out.print(al.get(i)+" ");
33         }
34         System.out.println();
35         for(int i=al.size()-1;i>=0;i--)
36         {
37             System.out.print(al.get(i)+" ");
38         }
39
40         System.out.println("\n\nReteriving the elements by using for-each loop");
41         for(Object o:al) // for(Integer o:al) // for(int o:al)
42         {
43             System.out.print(o+" ");
44         }
45         System.out.println("\n\nReteriving the elements by using Iterator Interface");
46         Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
47         while(i.hasNext())
48         {
49             System.out.print(i.next()+" ");
50         }
51
52         System.out.println("\n\n-----METHODS-----");
53
54         System.out.println("isEmpty() : "+al.isEmpty());

```

```

53      System.out.println("isEmpty() : "+al.isEmpty());
54      //al.clear();
55      //System.out.println("isEmpty() : "+al.isEmpty());
56      System.out.println("contains() : "+al.contains(10));
57
58      ArrayList<Integer> al2=new ArrayList<Integer>();
59      al2.add(10);
60      al2.add(100);
61      System.out.println("containsAll() : "+al.containsAll(al2));
62
63      System.out.println(al);
64      al.addAll(al2);
65      System.out.println(al);
66      al.add(2,95);
67      System.out.println(al);
68      al.set(1, 80);
69      System.out.println(al);
70
71      al.remove((Object)100);
72      System.out.println(al);
73
74      al.retainAll(al2);
75      System.out.println(al);
76  }
77  public static void main(String[] args)
78  {
79      ClassA aobj=new ClassA();
80      aobj.meth1();
81  }
82  }

```