```java
import java.util.ArrayList;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing ArrayList\n");

        ArrayList al=new ArrayList();

        al.add(10); // Insertion order is
        al.add("Java"); // Heterogeneous data is
        al.add(null);// null value is
        al.add(10);// Duplicates are ALLOWED
        al.add('A'); // It is available from Java 1.2v
        al.add(true);// Its default capacity is 10
        al.add(1); // Its increases by HALF
        al.add(99); // It is NOT Synchronized

        System.out.println(al);
    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
Implementing ArrayList

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

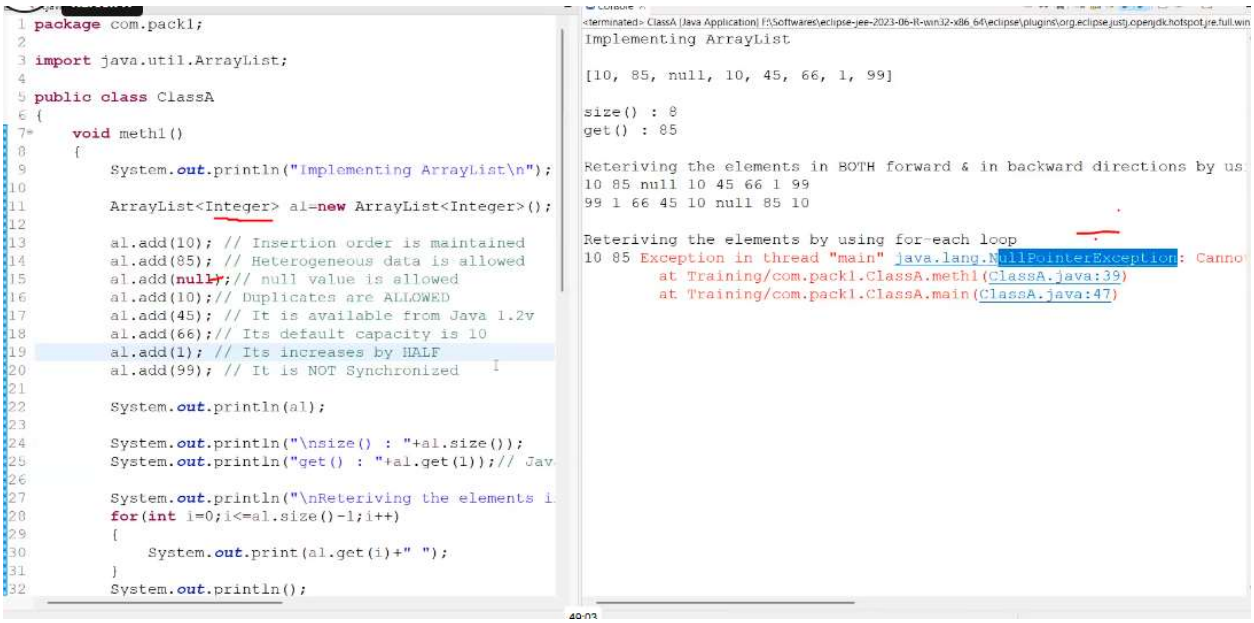
```java
public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing ArrayList\n");

        ArrayList al=new ArrayList();

        al.add(10); // Insertion order is maintained
        al.add("Java"); // Heterogeneous data is allowed
        al.add(null);// null value is allowed
        al.add(10);// Duplicates are ALLOWED
        al.add('A'); // It is available from Java 1.2v
        al.add(true);// Its default capacity is 10
        al.add(1); // Its increases by HALF
        al.add(99); // It is NOT Synchronized

        System.out.println(al);

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
Implementing ArrayList

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

From java1.5 version generic which allows homogenous data.

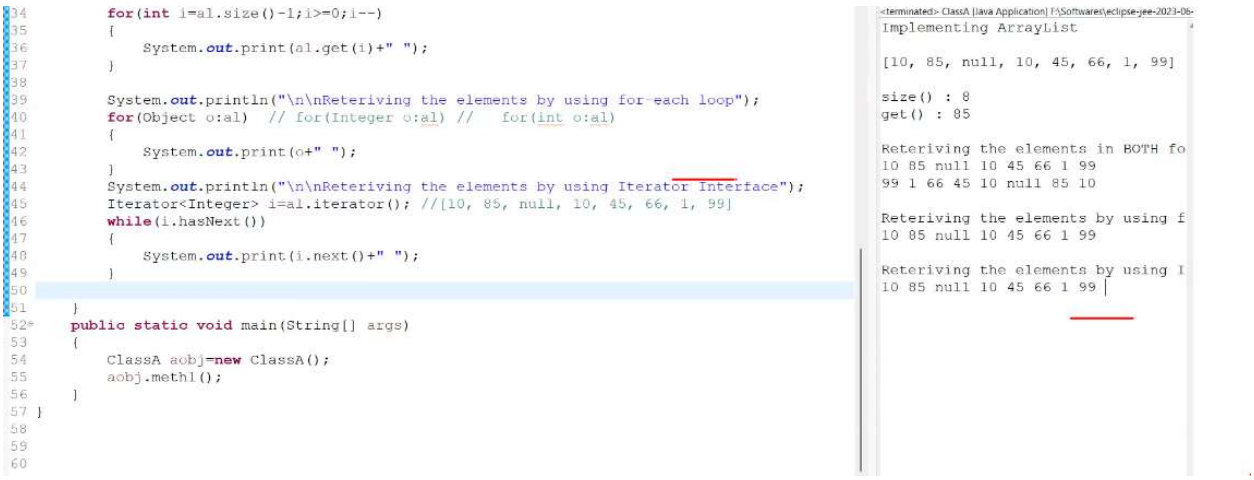
The main use of generics is to avoid type safety problems

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



```
package com.pack1;

import java.util.ArrayList;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing ArrayList\n");

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

        al.add(10); // Insertion order is maintained
        al.add(85); // Heterogeneous data is allowed
        al.add(null);// null value is allowed
        al.add(10);// Duplicates are ALLOWED
        al.add(45); // It is available from Java 1.2v
        al.add(66);// Its default capacity is 10
        al.add(1); // Its increases by HALF
        al.add(99); // It is NOT Synchronized

        System.out.println(al);

        System.out.println("\nsize() : "+al.size());
        System.out.println("get() : "+al.get(1));// Jav

        System.out.println("\nReteriving the elements i
        for(int i=0;i<=al.size()-1;i++)
        {
            System.out.print(al.get(i)+" ");
        }
        System.out.println();
```

```
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: Canno
        at Training/com.pack1.ClassA.meth1(ClassA.java:39)
        at Training/com.pack1.ClassA.main(ClassA.java:47)
```



```
        for(int i=al.size()-1;i>=0;i--)
        {
            System.out.print(al.get(i)+" ");
        }

        System.out.println("\n\nReteriving the elements by using for-each loop");
        for(Object o:al)  // for(Integer o:al) //   for(int o:al)
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the elements by using Iterator Interface");
        Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
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
```

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.

```
        for(Object o:al)  // for(Integer o:al) //   for(int o:al)
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the elements by using Iterator Interface");
        Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

        System.out.println("\n\n=========METHODS===========");

        System.out.println("isEmpty() : "+al.isEmpty());
        al.clear();
        System.out.println("isEmpty() : "+al.isEmpty());

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

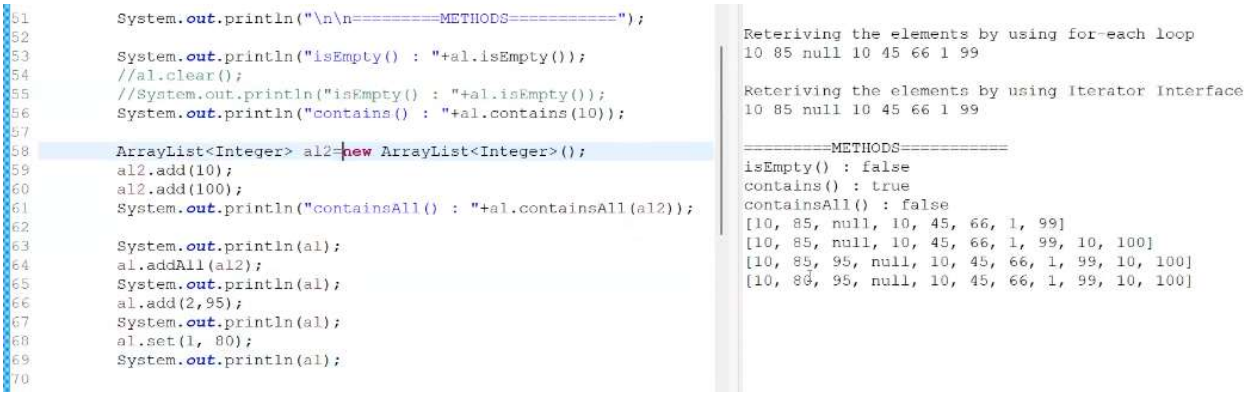
```
Implementing ArrayList

[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
```

```
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the elements by using Iterator Interface");
        Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

        System.out.println("\n\n=========METHODS===========");

        System.out.println("isEmpty() : "+al.isEmpty());
        //al.clear();
        //System.out.println("isEmpty() : "+al.isEmpty());
        System.out.println("contains() : "+al.contains(10));

    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
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
```

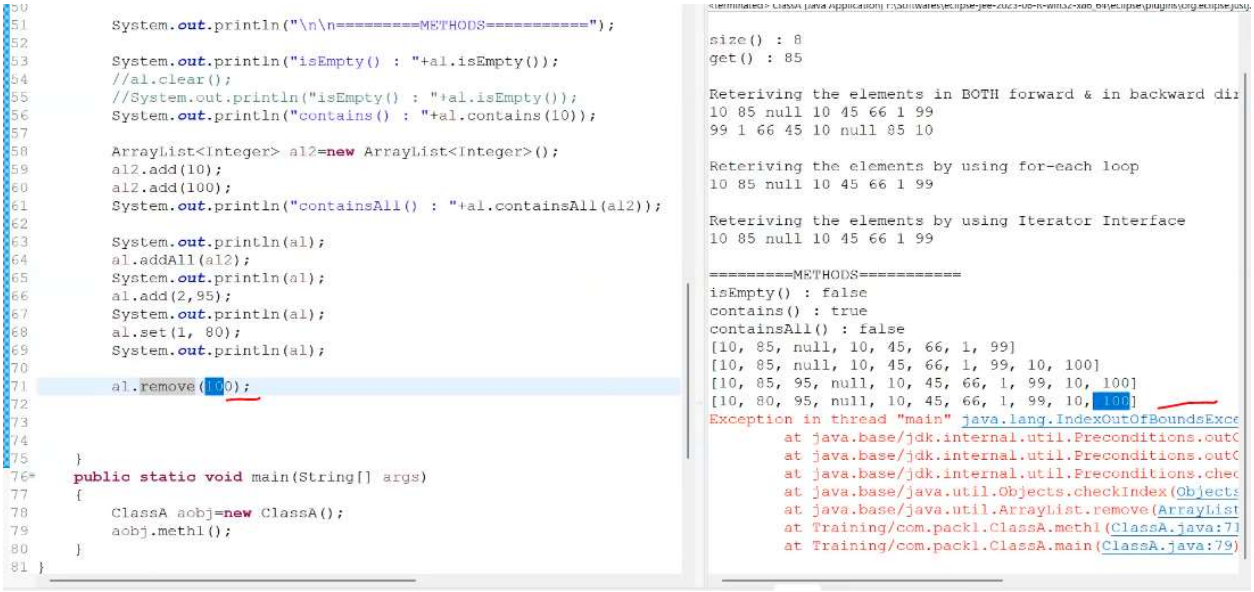
```
        for(Object o:al)  // for(Integer o:al) //   for(int o:al)
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the elements by using Iterator Interface");
        Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

        System.out.println("\n\n=========METHODS===========");

        System.out.println("isEmpty() : "+al.isEmpty());
        //al.clear();
        //System.out.println("isEmpty() : "+al.isEmpty());
        System.out.println("contains() : "+al.contains(10));

        ArrayList<Integer> al2=new ArrayList<Integer>();
        al2.add(10);
        al2.add(99);
        System.out.println("containsAll() : "+al.containsAll(al2));


    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
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
```



```
        System.out.println("\n\n=========METHODS===========");

        System.out.println("isEmpty() : "+al.isEmpty());
        //al.clear();
        //System.out.println("isEmpty() : "+al.isEmpty());
        System.out.println("contains() : "+al.contains(10));

        ArrayList<Integer> al2=new ArrayList<Integer>();
        al2.add(10);
        al2.add(100);
        System.out.println("containsAll() : "+al.containsAll(al2));

        System.out.println(al);
        al.addAll(al2);
        System.out.println(al);
        al.add(2,95);
        System.out.println(al);
        al.set(1, 80);
        System.out.println(al);
```

```
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]
```



```
        System.out.println("\n\n=========METHODS===========");

        System.out.println("isEmpty() : "+al.isEmpty());
        //al.clear();
        //System.out.println("isEmpty() : "+al.isEmpty());
        System.out.println("contains() : "+al.contains(10));

        ArrayList<Integer> al2=new ArrayList<Integer>();
        al2.add(10);
        al2.add(100);
        System.out.println("containsAll() : "+al.containsAll(al2));

        System.out.println(al);
        al.addAll(al2);
        System.out.println(al);
        al.add(2,95);
        System.out.println(al);
        al.set(1, 80);
        System.out.println(al);

        al.remove(100);


    }
    public static void main(String[] args)
    {
        ClassA aobj=new ClassA();
        aobj.meth1();
    }
}
```

```
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.IndexOutOfBoundsExce
        at java.base/jdk.internal.util.Preconditions.outC
        at java.base/jdk.internal.util.Preconditions.outC
        at java.base/jdk.internal.util.Preconditions.chec
        at java.base/java.util.Objects.checkIndex(Objects
        at java.base/java.util.ArrayList.remove(ArrayList
        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.

```java
package com.pack1;

import java.util.ArrayList;
import java.util.Iterator;

public class ClassA
{
    void meth1()
    {
        System.out.println("Implementing ArrayList\n");

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

        al.add(10); // Insertion order is maintained
        al.add(85); // Heterogeneous data is allowed
        al.add(null);// null value is allowed
        al.add(10);// Duplicates are ALLOWED
        al.add(45); // It is available from Java 1.2v
        al.add(66);// Its default capacity is 10
        al.add(1); // Its increases by HALF
        al.add(99); // It is NOT Synchronized

        System.out.println(al);

        System.out.println("\nsize() : "+al.size());
        System.out.println("get() : "+al.get(1));// Java


        System.out.println("\nReteriving the elements in BOTH forward & in backward directions by using for- loop");
        for(int i=0;i<=al.size()-1;i++)
        {
            System.out.print(al.get(i)+" ");
        }
        System.out.println();
        for(int i=al.size()-1;i>=0;i--)
        {
            System.out.print(al.get(i)+" ");
        }

        System.out.println("\n\nReteriving the elements by using for-each loop");
        for(Object o:al)  // for(Integer o:al) //   for(int o:al)
        {
            System.out.print(o+" ");
        }
        System.out.println("\n\nReteriving the elements by using Iterator Interface");
        Iterator<Integer> i=al.iterator(); //[10, 85, null, 10, 45, 66, 1, 99]
        while(i.hasNext())
        {
            System.out.print(i.next()+" ");
        }

        System.out.println("\n\n=========METHODS==========");

        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  }
```