Object-Oriented Programming (CS F213)

Module V: Collections in Java

CS F213 RL 12.2: ArrayList class in Java

BITS Pilani

Dr. Pankaj Vyas
Department of Computer Science, BITS-Pilani, Pilani Campus

CS F213 RL 12.2 : Topics

- ArrayList class in Java

ArrayList class in Java

- Supports Dynamic Arrays .
- Variable Length Array of Object References
- ArrayList Can Increase or Decrease in size.

public class ArrayList<E> extends AbstractList<E> implements List<E>, RandomAccess, Cloneable, Serializable

Where <E> Type of the Objects/Elements Stored in an ArrayList

Types of ArrayLists

1. ***Un-parameterized ArrayLists***

   - ❑ ***Supported in Earlier Versions of Java (1.3 and Earlier)***
   - ❑ ***Can Store/Handle Elements of Type Object.***

2. ***Parameterized ArrayLists***

   - ❑ ***Supported in ater versions after 1.4 onwards***
   - ❑ ***Can Handle/Store elements of Only Mentioned Type***

Note :To Use Un-parameterized ArrayLists, Compile the File Using (-Xlint) Option :

```
javac  -Xlint   <source-file-name>
```

ArrayList Constructors

1. **ArrayList() /ArrayList<T>() → Creates an Empty List, size =0**

   - **Examples :**

     Un-Parameterized Form

     Parameterized Form

     ```
     ArrayList  arr = new ArrayList();
     ArrayList<Box> boxes = new ArrayList<Box>();
     ArrayList<Student> students = new ArrayList<Student>();
     ```

2. **ArrayList(Collection c)/ArrayList<T>(Collection c) → Creates an ArrayList which is initialized with elements from other collection 'c'**

3. **ArrayList(int cap)/ArrayList<T>(int cap) → Creates an ArrayList with initial capacity.**

   - **Examples**

     Un-Parameterized Form

     Parameterized Form

     ```
     ArrayList  arr = new ArrayList(10);
     ArrayList<Box> boxes = new ArrayList<Box>(10);
     ArrayList<Student> students = new ArrayList<Student>(20);
     ```

Example 1:
Un-Parameterized ArrayList

*To Use ArrayList import java.util.**

Empty ArrayList size= 0, Type is Un-parameterized

Un-Parameterized ArrayList with size = 0 and capacity = 20

```
import java.util.*;
class ArrayListTest
{
		public static void main(String args[])
		{
				ArrayList arr = new ArrayList();

				ArrayList arr1 = new ArrayList(20);

				System.out.println(arr.size());
				System.out.println(arr1.size());

				// Adding Elements
```

Example 1: Un-Parameterized ArrayList …

arr.add(10);	*Adds integer 10 at index 0*
arr.add("A");	*Adds String "A" at index 1*
arr.add(new Double(12.56));	*Adds 12.56 at index 2*
arr.add(new Boolean(true));	*Adds boolean true at index 3*
arr.add(2,new Integer(30));	*Adds integer 30 at index 2*
// arr.add(6,new Integer(50));	***// IndexOutOfBoundsException***
System.out.println(arr.size());	*5*
arr1.addAll(arr); →	***Adds all elements of arr to end of arr1***

```
    } // End of Method
}// End of class
```

Example 2: Parameterized ArrayList

```
// FileName : ArrTest2.java
import java.util.*;
class ArrayListTest
{
        public static void main(String args[])
        {
                ArrayList<String> names = new ArrayList<String>();

                // Adding Elements
                names.add(“Java”);              // Adds Element at index 0
                names.add(1,“Mike”);            // Adds Element at index 1
                names.add(0,“Rahul”);           // Adds Element at index 0
                names.add(2,“Object”);          // Adds Element at index 2
                names.add(“Fortran”);           // Adds Element at index 4

                System.out.println(names.size());
                System.out.println(names);
```

Example 2: Parameterized ArrayList …

```
names.set(1,"Testing");     // Updates The Existing Element at index 1 with "Testing" (No Size Change)
names.add("Java");
names.add("Testing");
names.add("Java");

System.out.println(names);

names.remove(2);            // Removes Element From Index 2
System.out.println(names);

names.remove("Java");       // Removes Element "Java" if Exists Otherwise No Change in The List
System.out.println(names);

System.out.println(names.indexOf("Testing");
System.out.println(names.lastIndexOf("Testing");

}// End of Method
}// End of Class
```

Traversing/Iterator ArrayLists

1. Using for (…) Loop

```
ArrayList<String> names = new ArrayList<String>();

names.add(“Java”);                  // Adds Element at index 0
names.add(1,“Mike”);                // Adds Element at index 1
names.add(0,“Rahul”);               // Adds Element at index 0
names.add(2,“Object”);              // Adds Element at index 2
names.add(“Fortran”);               // Adds Element at index 4

// Forward Traversal
for(int i =0; names.size(); i++)
            System.out.println(names.get(i));

// Backward Traversal
for(int i =names.size()-1; i >=0 ; i--)
            System.out.println(names.get(i));
```

Traversing/Iterator ArrayLists

2. Using for each Loop

```
ArrayList<String> names = new ArrayList<String>();

names.add(“Java”);                  // Adds Element at index 0
names.add(1,“Mike”);                // Adds Element at index 1
names.add(0,“Rahul”);               // Adds Element at index 0
names.add(2,“Object”);              // Adds Element at index 2
names.add(“Fortran”);               // Adds Element at index 4

// Forward Traversal
for(String    i       =       names)
              System.out.println(i);
```

Thank You