JAVA ArrayList Class

Furkan Ekici
`Turkcell Camp`
furkanekici1997@gmail.com

February 1, 2022

Contents

Abstract

ArrayLists are dynamic versions of classical arrays. Sizes may vary. Thus, they provide more flexibility when programming. In this report, what ArrayLists do, how they are used, their differences from normal arrays and their methods will be discussed.

1 What is ArrayList

ArrayList class can be thought of as an advanced version of classical arrays. Unlike conventional arrays, ArrayLists can dynamically change size. It contains many features such as adding new data, deleting existing data, etc. The ArrayList in java.util is a generic class. Generic means it can work with any class. It manages an Object Array in it. Object is a super class of all classes. In Java, a class may not seem to inherit from any class. But if extends is not written next to it when creating the class, Java implicitly puts 'extends Object' next to it. That is, every class inherits from the Object class. Therefore, managed array in ArrayList can hold reference of each class. This makes the generic structure possible.

1.1 Relationship with other Class and Interfaces

It inherits from the AbstractList class, while implementing the List, RandomAccess, Cloneable, and Serializable interfaces.

1.2 Data Field Variables

- DEFAULT_CAPACITY : When no size is specified, this constant variable is used to size the array managed in the ArrayList. It holds the value 10.
- EMPTY_ELEMENTDATA and DEFAULT_ELEMENT_ELEMENTDATA : Both are an empty, constant array of Object class.
- elementData : It is an array of Objects managed in ArrayList.
- size : It keeps size of Object array.

1.3 Constructors

There 3 constructors in ArrayList class.

- ArrayList(): This is no-argument constructor of ArrayList class. Usually this constructor is used. When ArrayList object is created by using this constructor, ArrayList object keeps 10 elements as a default.
- ArrayList(int capacity): This is used to create an ArrayList class of desired size according to capacity variable.
- ArrayList(Collection c): This is used to convert items from any Collection object to ArrayList object.
  [1]

1.4 Capacity of ArrayList

ArrayList wants to use memory quite efficiently. For this, it organizes its capacity according to the data it has. If there is not enough space left, it increases its size by 1.5 times. If there is too much free space, it can reduce in size. So it uses memory efficiently. [1]

1.5 Implementation of ArrayList

First of all java.util.ArrayList is imported. Then, it is created as a normal object.

```
ArrayList<T> listName = new ArrayList<T>();
```

- T can be any class. But cannot be any primitive data type. For example int or double. Instead of primitive data types, Wrapper classes can be used. Wrapper class is class form of primitive data types.
- In second part of the code(new ArrayList<T>();), T is not necessary. It can be empty.

2 ArrayList vs Array

Arrays are fixed size structures. Once created, elements cannot be added or removed. Only the contents of the elements inside can be changed. ArrayList, on the other hand, contains these features that Arrays cannot.

Operation	Array	ArrayList
Accessing an element	+	+
Updating an element	+	+
Returning size	+	+
Adding an element	-	+
Inserting an element any index	-	+
Removing an element using index	-	+
Removing specific element	-	+
Removing all elements	-	+

[2]

3 ArrayList Methods and Examples

- Creating an ArrayList : `ArrayList<String> list = new ArrayList<>();`
- Accessing an element : `list.get(index);`
- Updating an element : `list.set(index, "London");`
- Returning size : `list.size();`
- Adding a new element : `list.add("London");`
- Inserting a new element : `list.add(index, "London");`
- Removing an element using index : `list.remove(index);`
- Removing specific object : `list.remove(Object);`
- Removing all elements : `list.clear();`

  [2]

References

[1] @jagadeepmahendran. *Internal Working of ArrayList in Java*. URL: `https://www.geeksforgeeks.org/internal-working-of-arraylist-in-java/`. (accessed: 01.02.2022).

[2] Havva Esin Ünal. "Introduction to Java Programming Week13". In: *Çukurova University* (2021), pp. 18–24.