1Z0-808 Exam Topic Reviewer

TopicId: 1025
Topic: ArrayList and Basic Collections

August 5, 2025

Beyond Arrays: The Power of `ArrayList`

Arrays are powerful but rigid. Their biggest limitation is their fixed size. To solve this, Java provides the Collections Framework, and your go-to class from this framework will be `ArrayList`. It's essentially a resizable array on steroids. For the exam, you need to understand how it differs from an array and how to use its core methods.

0.1 Array vs. `ArrayList`

Let's get this straight, as the exam will test the differences.

Feature	**Array**
Size	Fixed, defined at instar
Data Types	Primitives and Objects
Type Safety	Compile-time check
Get Size	`length` property
API	Basic access via `[]`

0.2 Working with `ArrayList`

To use `ArrayList`, you must import it: `import java.util.ArrayList;`.

```
// The diamond operator <> infers the type.
ArrayList<String> names = new ArrayList<>();
List<Integer> numbers = new ArrayList<>(); // Also valid, programming to the inte

names.add("Alice"); // Adds to the end
names.add(0, "Bob"); // Adds "Bob" at index 0, shifts "Alice" to index 1

System.out.println(names.get(1)); // Prints "Alice"

String oldName = names.set(0, "Bill"); // Replaces "Bob" with "Bill", returns "Bo

System.out.println(names.size()); // Prints 2
```

0.3 The `remove()` Method Trap

This is a classic source of confusion and a perfect exam question. `ArrayList` has two `remove` methods:

- `E remove(int index)`: Removes the element at the specified position.
- `boolean remove(Object o)`: Removes the first occurrence of the specified element.

When you have an `ArrayList<Integer>`, the compiler gets confused. Which one should it call?

```
List<Integer> list = new ArrayList<>();
list.add(10);
list.add(20);
list.add(30);
list.remove(1); // Calls remove(int index). Removes element at index 1 (the 20).
// List is now [10, 30]

// To remove by object value, you must cast or wrap it:
list.remove(Integer.valueOf(10)); // Calls remove(Object o). Removes the 10.
// List is now [30]
```

Be extremely careful with this distinction.

0.4 Important Utility Methods

A few other methods you must know:

- `clear()`: Removes all elements.
- `isEmpty()`: Returns `true` if the list has no elements.
- `contains(Object o)`: Returns `true` if the list contains the specified element.
- `Arrays.asList()`: A handy but tricky utility.

  ```
  String[] array = {"A", "B"};
  List<String> list = Arrays.asList(array); // Creates a List view of the arra
  // list.remove("A"); // Throws UnsupportedOperationException!
  // list.add("C");    // Also throws UnsupportedOperationException!
  list.set(0, "Z");   // This is OK. The original array becomes {"Z", "B"}.
  ```

  The list returned by `Arrays.asList()` is **not** a `java.util.ArrayList`. It's a fixed-size list backed by the original array. You cannot change its size.

Key Takeaways for the 1Z0-808 Exam

- Know the key differences between Array and ArrayList (size, type, API).
- Memorize the core `ArrayList` methods: `add`, `get`, `set`, `remove`, `size`.
- Understand the `remove(index)` vs. `remove(Object)` ambiguity with `ArrayList<Intege`
- Be aware of the fixed-size, backed nature of the `List` returned from `Arrays.asList()`.