Guided LAB - 303.11.2 - ArrayList and ArrayList Methods

Lab Objective:

In this lab, we will demonstrate how to declare and initialize ArrayList, and how to manipulate ArrayList using built-in methods.

By the end of this lab, learners will be able to use ArrayList and arraylist methods.

Introduction

Key Points about ArrayList in Java:

- An ArrayList is a resizable array, also called a dynamic array. It grows its size to accommodate new elements, and shrinks its size when the elements are removed.
- An ArrayList internally uses an array to store the elements. Just like arrays, it allows you to retrieve the elements by index.
- A Java ArrayList allows for duplicate and null values.
- A Java ArrayList is an ordered collection, and maintains the insertion order of the elements.
- You cannot create an ArrayList of primitive types such as `int`, `char`, etc.; you need to use boxed types such as `Integer`, `Character`, `Boolean`, etc.
- A Java ArrayList is not synchronized. If multiple threads try to modify an ArrayList simultaneously, the final outcome will be

non-deterministic. You must explicitly synchronize access to an ArrayList if multiple threads are going to modify it.

Instructions:

Example 1: Creating an ArrayList and Adding New Elements

This example shows:

- How to create an ArrayList using the ArrayList() constructor.
- How to add new elements to an ArrayList using the add() method.

Create a class named ***“CreateArrayListExample”*** and add the code below:

```
import java.util.ArrayList;
import java.util.List;

public class CreateArrayListExample {

    public static void main(String[] args) {
        // Creating an ArrayList of String
        List<String> animals = new ArrayList<>();

        // Adding new elements to the ArrayList
        animals.add("Lion");
        animals.add("Tiger");
        animals.add("Cat");
        animals.add("Dog");

        System.out.println(animals);

  // Adding an element at a particular index in an ArrayList
        animals.add(2, "Elephant");
        System.out.println(animals);
   }
}
```

```
# Output
[Lion, Tiger, Cat, Dog]
[Lion, Tiger, Elephant, Cat, Dog]
```

Example 2: Creating an ArrayList From Another Collection

This example shows:

- How to create an ArrayList from another collection using the ***ArrayList(Collection c)*** constructor.
- How to add all of the elements from an existing collection to the new ArrayList using the ***addAll()*** method.

Create a new class named **"`CreateArrayListFromCollectionExample`"** and add the code below in that class.

```
import java.util.ArrayList;
import java.util.List;
public class CreateArrayListFromCollectionExample {

    public static void main(String[] args) {
        List<Integer> firstFivePrimeNumbers = new ArrayList<>();
        firstFivePrimeNumbers.add(2);
        firstFivePrimeNumbers.add(3);
        firstFivePrimeNumbers.add(5);
        firstFivePrimeNumbers.add(7);
        firstFivePrimeNumbers.add(11);

        // Creating an ArrayList from another collection
        List<Integer> firstTenPrimeNumbers = new
ArrayList<>(firstFivePrimeNumbers);

        List<Integer> nextFivePrimeNumbers = new ArrayList<>();
        nextFivePrimeNumbers.add(13);
        nextFivePrimeNumbers.add(17);
        nextFivePrimeNumbers.add(19);
        nextFivePrimeNumbers.add(23);
        nextFivePrimeNumbers.add(29);

        // Adding an entire collection to an ArrayList
        firstTenPrimeNumbers.addAll(nextFivePrimeNumbers);

        System.out.println(firstTenPrimeNumbers);
    }

}
```

```
# Output

[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

Example 3: Accessing Elements from an ArrayList

This example shows:

- The use of the isEmpty() method to check if an ArrayList is empty.
- The use of the size() method to find the size of an ArrayList.
- The use of the get() method to access an element at a particular index in an ArrayList.
- The use of the set() method to modify the element at a particular index in an ArrayList.

Create a new class named ***"CreateArrayListFromCollectionExample"*** and add the code below in that class.

```
import java.util.ArrayList;
import java.util.List;

public class AccessElementsFromArrayListExample {
    public static void main(String[] args) {
        List<String> topCompanies = new ArrayList<>();

        // Check if an ArrayList is empty
        System.out.println("Is the topCompanies list empty? : " +
topCompanies.isEmpty());

        topCompanies.add("Google");
        topCompanies.add("Apple");
        topCompanies.add("Microsoft");
        topCompanies.add("Amazon");
        topCompanies.add("Facebook");

        // Find the size of an ArrayList
        System.out.println("Here are the top " + topCompanies.size() + " companies
in the world");
        System.out.println(topCompanies);

        // Retrieve the element at a given index
        String bestCompany = topCompanies.get(0);
        String secondBestCompany = topCompanies.get(1);
        String lastCompany = topCompanies.get(topCompanies.size() - 1);

        System.out.println("Best Company: " + bestCompany);
        System.out.println("Second Best Company: " + secondBestCompany);
        System.out.println("Last Company in the list: " + lastCompany);

         // Modify the element at a given index
         topCompanies.set(4, "Walmart");
         System.out.println("Modified top companies list: " +
topCompanies);
    }
}
```

```
# Output
Is the topCompanies list empty? : true
Here are the top 5 companies in the world
[Google, Apple, Microsoft, Amazon, Facebook]
Best Company: Google
Second Best Company: Apple
Last Company in the list: Facebook

Modified top companies list: [Google, Apple, Microsoft, Amazon, Walmart]
```

Example 4: Removing Elements from an ArrayList

This example shows:

1. How to remove an element at a given index in an ArrayList | remove(int index).
2. How to remove an element from an ArrayList | remove(Object o).
3. How to remove all of the elements from an ArrayList that exist in a given collection | removeAll().
4. How to remove all of the elements matching a given predicate | removeIf().
5. How to clear an ArrayList | clear().

Create a new class named ***“RemoveElementsFromArrayList”*** and add the code below in that class.

```
import java.util.ArrayList;
import java.util.List;
import java.util.function.Predicate;

public class RemoveElementsFromArrayList {
    public static void main(String[] args) {
        List<String> programmingLanguages = new ArrayList<>();
        programmingLanguages.add("C");
        programmingLanguages.add("C++");
        programmingLanguages.add("Java");
        programmingLanguages.add("Kotlin");
        programmingLanguages.add("Python");
        programmingLanguages.add("Perl");
        programmingLanguages.add("Ruby");

        System.out.println("Initial List: " + programmingLanguages);

        // Remove the element at index `5`
        programmingLanguages.remove(5);
        System.out.println("After remove(5): " + programmingLanguages);

        // Remove the first occurrence of the given element from the ArrayList
        // (The remove() method returns false if the element does not exist in the
ArrayList)
        boolean isRemoved = programmingLanguages.remove("Kotlin");
        System.out.println("After remove(\"Kotlin\"): " + programmingLanguages);

        // Remove all the elements that exist in a given collection
        List<String> scriptingLanguages = new ArrayList<>();
        scriptingLanguages.add("Python");
        scriptingLanguages.add("Ruby");
        scriptingLanguages.add("Perl");

        programmingLanguages.removeAll(scriptingLanguages);
        System.out.println("After removeAll(scriptingLanguages): " + programmingLanguages);

        // Remove all elements from the ArrayList
        programmingLanguages.clear();
        System.out.println("After clear(): " + programmingLanguages);
    }
}
```

```
# Output
Initial List: [C, C++, Java, Kotlin, Python, Perl, Ruby]
After remove(5): [C, C++, Java, Kotlin, Python, Ruby]
After remove("Kotlin"): [C, C++, Java, Python, Ruby]
After removeAll(scriptingLanguages): [C, C++, Java]
After clear(): []
```

Example 5: Iterating over an ArrayList

The following example shows how to iterate over an ArrayList using:

1. iterator().
2. iterator() and forEachRemaining() method.
3. listIterator().
4. Simple *for* loop.
5. *Enhanced* ***for*** *loop* with index.

Create a new class named ***"IterateOverArrayList"*** and add the code below in that class.

```
import java.util.*;
public class IterateOverArrayList {
  public static void main(String[] args) {
     List<String> tvShows = new ArrayList<>();
     tvShows.add("Breaking Bad");
     tvShows.add("Game Of Thrones");
     tvShows.add("Friends");
     tvShows.add("Prison break");
     System.out.println("\n=== Iterate using an iterator() ===");
     Iterator<String> tvShowIterator = tvShows.iterator();
     while (tvShowIterator.hasNext()) {
        String tvShow = tvShowIterator.next();
        System.out.println(tvShow);
     }

System.out.println("==Iterate using an iterator() and forEachRemaining() method ===");
     tvShowIterator = tvShows.iterator();
     tvShowIterator.forEachRemaining(tvShow -> {
        System.out.println(tvShow);
     });


     System.out.println("\n=== Iterate using simple for-each loop ===");
     for(String tvShow: tvShows) {
        System.out.println(tvShow);
     }

     System.out.println("\n=== Iterate using for loop with index ===");
```

```
    for(int i = 0; i < tvShows.size(); i++) {
      System.out.println(tvShows.get(i));
    }
    System.out.println("\n=== Iterate iterator ===");
    ListIterator iterator = tvShows.listIterator();
    System.out.println("Elements in forward direction");

    System.out.println("\n====== Iterate using while loop======");

    while (iterator.hasNext())
    {
      System.out.println(iterator.next());
    }

    System.out.println("=========Elements in backward direction======");

    while (iterator.hasPrevious())
    {
      System.out.println(iterator.previous());
    }
  }
}
```

# Output:

```
=== Iterate using an iterator() ===


Breaking Bad
Game Of Thrones
Friends
Prison break

=== Iterate using an iterator() and Java 8 forEachRemaining() method ===
Breaking Bad
Game Of Thrones
Friends
Prison break

=== Iterate using simple for-each loop ===
Breaking Bad
Game Of Thrones
Friends
Prison break
```

```
=== Iterate using for loop with index ===
Breaking Bad
Game Of Thrones
Friends
Prison break

=== Iterate iterator ===
Elements in forward direction

====== Iterate using while loop=======
Breaking Bad
Game Of Thrones
Friends
Prison break
=========Elements in backward direction======
Prison break
Friends
Game Of Thrones
Breaking Bad
```

Example 6: Searching for Elements in an ArrayList

The example below shows how to:

- Check if an ArrayList contains a given element | contains().
- Find the index of the first occurrence of an element in an ArrayList | indexOf().
- Find the index of the last occurrence of an element in an ArrayList | lastIndexOf().

Create a new class named ***"SearchElementsInArrayListExample"*** and add the code below in that class.

```java
import java.util.ArrayList;
import java.util.List;

public class SearchElementsInArrayListExample {
    public static void main(String[] args) {
        List<String> names = new ArrayList<>();
        names.add("John");
        names.add("Alice");
        names.add("Bob");
        names.add("Steve");
        names.add("John");
        names.add("Steve");
        names.add("Maria");

      // Check if an ArrayList contains a given element
        System.out.println("Does names array contain \"Bob\"? : " +
names.contains("Bob"));

      // Find the index of the first occurrence of an element in an ArrayList
        System.out.println("indexOf \"Steve\": " + names.indexOf("Steve"));
        System.out.println("indexOf \"Mark\": " + names.indexOf("Mark"));

      // Find the index of the last occurrence of an element in an ArrayList
        System.out.println("lastIndexOf \"John\" : " + names.lastIndexOf("John"));
        System.out.println("lastIndexOf \"Bill\" : " + names.lastIndexOf("Bill"));
    }
}
```

```
# Output

Does names array contain "Bob"? : true
indexOf "Steve": 3
indexOf "Mark": -1
lastIndexOf "John" : 4

lastIndexOf "Bill" : -1
```

Submission Instructions:

Include the following deliverables in your submission -

- Submit your source code using the Start Assignment button in the top-right corner of the assignment page in Canvas.

CANVAS STAFF USE ONLY: Canvas Submission Guideline:

Instructions for Canvas Assignment Creation
Assignment Name: GLAB - 303.11.2 - ArrayList and ArrayList methods **Points:** 100 **Assignment Group:** Module 303: Java SE Review (Not Graded) **Display Grade As:** Complete/Incomplete **Do not count this assignment towards the final grade:** Checked **Submission Types:** File uploads **Everything else is the default.**