Topics: Collections Interface, List Interface, ArrayList implementation
Ref: Polymorphism and Interfaces Worksheet.

The Java Collection framework is a set of interfaces and classes that help manage objects. The List interface extends Collection and two common concrete classes that implementations List are ArrayList and LinkedList. We will explore ArrayList in this worksheet.

Standard arrays in Java have notable limitations, such as their fixed length. Once an array is filled to capacity, it cannot be re-sized and a new array must be created of larger sizer. Insertion and deletion for a standard array are also cumbersome, requiring existing array elements to be shifted around.

The ArrayList class provides a dynamic array-like data structure which can grow or shrink dynamically as elements are added or removed. The type of objects that it stores needs to be specified but the size ie number of objects does not need to be specified.

Here is how to create an ArrayList object and add some elements into it that demonstrates this.

```
ArrayList<Integer> arraylist = new ArrayList<>();

arraylist.add(2);
arraylist.add(4);
arraylist.add(6);
```

Copy paste this code and see it run. Be sure to import **java.util.ArrayList**
Change the declaration to

```
List<Integer> arraylist = new ArrayList<>();
```

Does this work? Why or why not?

Note that collections like ArrayList store objects, not primitives such as *int.* So, the above *arraylist* stores elements of type *Integer*. This is a 'wrapper' class which encapsulates an *int* and provides functionalities to the primitive *int* data types. Java will automatically convert a primitive type to the corresponding object if needed.

Ex. 1) Write Java code to create a standard integer array called *std_array* of size 3, and an Integer ArrayList called 'array_list'. Provide each the following elements: [0, 2, 4].

Ex. 2) Now, expand the Problem 1 code by inserting the elements [6, 8, 10] at the end of both *std_array* and *array_list*. You will need to re-size *std_array*. You could do this by writing a method that creates a new array that is twice the size of an incoming array and copies the elements over before returning it.

Look up other ArrayList method to retrieve (get), delete (remove) and change (set) elements. Notice that arraylists have indices much the same as a regular array! Many of the ArrayList methods use indices to search and manipulate arraylists. For example:

We can retrieve the element 4 at index 2 in the ArrayList from Problem 2 using the get method.

```
// array_list contains Integers [0,2,4,6,8,10]
array_list.get(2);
```

Ex. 3) To practice getting items, write a Java method which can reverse our *array_list.* First, create a new empty ArrayList called *temp.* Then iterate through *array_list* backwards, and within each loop cycle, retrieve an element from *array_list* using *get()*, and use *add()* to insert it into *temp*. Use the method *size()* to get the size of an ArrayList.

ArrayLists also have the *remove()* method to delete elements by index position. The method dynamically re-arranges the remaining elements after the deletion.

Ex. 4) Given the following Integer ArrayList *scores_list* of student exam scores: [84, 70, 90, 67, 77, 94, 92, 71, 80, 95], write some lines of code that remove all the odd-valued scores, i.e. 67, 77, 71 and 95

It's often important to remove duplicate elements in an ArrayList. Look up the remove method version that removes objects by value. Note that remove(value) will remove the *first* occurrence of the provided value. What if there are multiple occurrences of the value?

Ex. 5) Write a Java method to remove multiple occurrences of a certain value in a given String ArrayList. Use the *contains()* method to help you, which checks if a certain value is present in the list and returns the appropriate *boolean*.

Bonus) Write a Java method called *uniqueArrayList()* which takes an Integer ArrayList and returns an ArrayList of its unique elements.

Ex. 6) Write a few lines of code to create an ArrayList of Cones (use the class in the Inheritance worksheet). Add five Cone objects to this arraylist. Now insert a new Cone object at every other position in this arraylist. Finally delete all the Cone objects of the original arraylist that you created.