



MACQUARIE
University

Department of Computing

COMP125 Fundamentals of Computer Science
Workshop - ArrayLists 1

Learning outcomes

By the end of this session, you will have learnt the basics about containers and `ArrayList` class.

Questions

1. Why do we need classes that hold collections when we already have arrays? Give three reasons.
 - a. We have to re-size arrays manually if they get full and more items need to be added.
 - b. Everything on arrays needs to be done using just `arr.length` and `arr[i]` operations.
 - c. We might want to customize what operations we want over our collection, and their interpretation.
 - d. We might want a more specific ordering in our collection.
2. An `ArrayList` is a resizable collection of objects. If you don't parameterise an `ArrayList`, it can hold a variety of objects. That is, each item of the `ArrayList` can be of a different class.

A parameter-less `ArrayList` is created as -

```
1 ArrayList list = new ArrayList();
```

where `list` is the `ArrayList` object.

You can parameterize an `ArrayList` so that it stores objects of a specific class. A parameterized `ArrayList` is created as -

```
1 ArrayList<ClassType> list = new ArrayList();
```

where `list` is the `ArrayList` object.

For example,

```
1 ArrayList<String> list = new ArrayList();
```

can only hold String objects.

A subset of methods (the important ones) applicable to an ArrayList object is given below -

- `int size()`: returns the number of items in the list
- `Object get(int index)`: returns the Object at the specified index, if any; and null otherwise.
- `add(Object obj)`: adds the specified Object to the end of the list and returns true, if it can; and false otherwise.
- `add(int idx, Object obj)`: adds the specified Object at given index. Shifts all items at index idx onwards to the right.
- `contains(Object obj)`: returns true if the specified exists, and false otherwise.
- `indexOf(Object obj)`: returns the index of the specified Object if it exists, and -1 otherwise.
- `remove(Object obj)`: removes the specified Object to the list and returns true, if it can; and false otherwise.
- `set(int index, Object obj)`: updates the item at given index to the object passed. Returns the item that the new object has replaced.

Write a piece of code that performs the following operations in the given order -

- a. Create an ArrayList list to hold String objects
- b. Add "hello" to list
- c. Add "this" to list
- d. Add "is" to list
- e. Add "your" to list
- f. Add "captain" to list
- g. Add "speaking" to list
- h. Remove the 5th item (at index 4) from list
- i. Insert "brother" at index 4 in list.
- j. Change the 6th item (at index 5) to "talking"
- k. Display the number of items in list
- l. Display all items of the list
- m. Display each item in list on a separate line.
- n. Store in a variable loc the index where "brother" is found in the list, and display it.
- o. Display the first character of each item of the list
- p. Create a String consisting of the first characters of each item. For example, if the items are "this", "is", "fun", your String should be "tif"
- q. Count the number of items that begin with an 't' or 'T'
- r. Count the number of items that are more than 3 characters long
- s. Create an arraylist of items that are more than 3 characters long and display it
- t. Create a char array consisting of the last characters of each item. For example, if the items are "this", "is", "fun", your array should be {'s', 's', 'n'}
- u. Replace each item by their uppercase version, that is capitalize all Strings

Solution:

```
1 ArrayList<String> list = new ArrayList();
2 list.add("hello");
3 list.add("this");
4 list.add("is");
5 list.add("your");
6 list.add("captain");
7 list.add("speaking");
8 list.remove(4);
9 list.add(4, "brother");
10 list.set(5, "talking");
11 System.out.println(list.size());
12 System.out.println(list);
13 for(int i=0; i<list.size(); i++)
14     System.out.println(list.get(i));
15 int loc = list.indexOf("done");
16 for(int i=0; i<list.size(); i++)
17     System.out.println("item_"+(i+1)+":_"+list.get(i));
18 System.out.println();
19
20 for(int i=0; i<list.size(); i++)
21     System.out.println("First_character_of_item_"+(i+1)+":_"+list.get(i).charAt(0));
22 System.out.println();
23
24 int count = 0;
25 for(int i=0; i<list.size(); i++)
26     if(list.get(i).substring(0, 1).equalsIgnoreCase("a"))
27         count++;
28 System.out.println(count+"_items_begin_with_'a'");
29
30 count = 0;
31 for(int i=0; i<list.size(); i++)
32     if(list.get(i).length() > 3)
33         count++;
34 System.out.print(count+"_items_are_longer_than_3_characters:");
35
36 LinkedList<String> longOnes = new LinkedList<String>();
37 for(int i=0; i<list.size(); i++)
38     if(list.get(i).length() > 3)
39         longOnes.add(list.get(i));
40 System.out.println(longOnes);
41
42 String firstChars = "";
43 for(int i=0; i<list.size(); i++)
44     firstChars+=list.get(i).charAt(0);
45 System.out.println("String_of_first_characters:"+firstChars);
46
47 String lastChars = "";
48 for(int i=0; i<list.size(); i++)
49     lastChars+=list.get(i).charAt(list.get(i).length()-1);
50 System.out.println("String_of_last_characters:"+lastChars);
51
52 for(int i=0; i<list.size(); i++)
53     list.set(i, list.get(i).toUpperCase());
54 System.out.println("Capitalized_list:"+list);
```

3. Following is an incomplete class definition for a custom-built array-based list. Complete the

method `removeFirst`. Details provided as javadoc method header comment.

```
1 public class MyArrayList {
2     private double[] data;
3     private int nItems;
4     ...
5     /**
6      * remove first item in the list, if any and
7      * return it from the method
8      *
9      * return null if list is empty
10    */
11    public Double removeFirst() {
12        if(nItems == 0)
13            return null;
14        double removedItem = data[0];
15        for(int i=1; i < nItems; i++) {
16            data[i-1] = data[i];
17        }
18        nItems--;
19        return removedItem;
20    }
21 }
```

4. Add a method `product` that when passed an `ArrayList` of `Double` objects, returns the product of all items in the `ArrayList`. The method should return 0 if the list is null or empty.

```
1 public static double product(ArrayList <Double> list)
```

Solution:

```
1 public static double product(ArrayList <Double> list) {
2     if(list == null || list.size() == 0)
3         return 0;
4     double result = 1;
5     for(Double item: list)
6         result*=item;
7     return result;
8 }
```

5. Add a method `sumPositive` that when passed an `ArrayList` of `Integer` objects, returns the sum of all positive values in the `ArrayList`. The method should return 0 if the list is null or empty.

```
1 public static int sumPositive(ArrayList <Integer> list)
```

Solution:

```
1 public static int sumPositive(ArrayList <Integer> list) {
2     if(list == null || list.size() == 0)
3         return 0;
4     int result = 0;
5     for(Integer item: list)
6         if(item > 0)
7             result+=item;
8     return result;
9 }
```

6. Add a method `count` that when passed an `ArrayList<Integer>` `list` and an `Integer` `target`, returns the number of times `target` exists in `list`.

```
1 public static int count(ArrayList<Integer> list, Integer target)
```

Solution:

```
1 int count(ArrayList<Integer> list, Integer target) {  
2     if(list == null)  
3         return 0;  
4     int result = 0;  
5     for(Integer item: list)  
6         if(item == target)  
7             result++;  
8     return result;  
9 }
```

7. Write a method that when passed an `ArrayList` of characters, returns an array containing the characters of the `ArrayList`. For example, if the `ArrayList` passed is `['v', 'e', 'n', 'd', 'e', 't', 't', 'a']`, the array returned should be `{'v', 'e', 'n', 'd', 'e', 't', 't', 'a'}`. You may NOT use built-in methods to convert an `ArrayList` to an array.

Solution:

```
1 public static char[] toArray(ArrayList<Character> list) {  
2     if(list == null)  
3         return null;  
4     char[] result = new char[list.size()];  
5     int i = 0;  
6     for(Character item: list) {  
7         result[i] = item;  
8         i++;  
9     }  
10    return result;  
11 }
```

8. Complete the method `squared` that when passed an `ArrayList<Integer>` `list`, squares all items of `list`. So if the `list` that is passed is `[3, 1, 7]`, after the method executes, it becomes `[9, 1, 49]`.

Hint 1: the method on `ArrayList` that you'll need are,

- `size()`
- `get(int index)`
- `set(int index, int value)`

```
1 public static void squared(ArrayList<Integer> list)
```

Solution:

```
1 void squared(ArrayList<Integer> list) {  
2     if(list == null)  
3         return;  
4 }
```

```
5         for(int i=0; i<list.size(); i++)
6             list.set(i, list.get(i) * list.get(i));
7     }
```

9. **(Challenging)** Write a method that when passed an arraylist of arraylists of integers, returns an arraylist containing items that are exclusive to each list. For example, if the list passed is [[8, 1, 4, 2, 4, 2, 1], [6, 4, 9, 8, 8, 8], [5, 3, 8, 8, 5, 6]], the method should return an ArrayList containing [1, 2, 2, 1, 9, 5, 3, 5]

Solution:

```
1 public static ArrayList<Integer> exclusiveItems (ArrayList<ArrayList<Integer>> megaList) {
2     ArrayList<Integer> result = new ArrayList();
3     for(int i=0; i < megaList.size(); i++) {
4         for(Integer item: megaList.get(i)) {
5             boolean dup = false;
6             for(int k=0; k < megaList.size() && !dup; k++) {
7                 if(i != k && megaList.get(k).contains(item)) {
8                     dup = true;
9                 }
10            }
11            if(!dup)
12                result.add(item);
13        }
14    }
15    return result;
16 }
```

10. Discuss the time complexities of following operations on an arraylist (best and worst cases).
- a. inserting item at an arbitrary position
 - b. removing item from an arbitrary position
 - c. accessing item at an arbitrary position

Operation	Best case	Worst case
Accessing an item	$O(1)$	$O(1)$
Inserting an item	$O(1)$	$O(n)$
Removing an item	$O(1)$	$O(n)$