*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 -

   ```
       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 -

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

   where `list` is the `ArrayList` object.

For example,

```
    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:**

```java
ArrayList<String> list = new ArrayList();
list.add("hello");
list.add("this");
list.add("is");
list.add("your");
list.add("captain");
list.add("speaking");
list.remove(4);
list.add(4, brother);
list.set(5, "talking");
System.out.println(list.size());
System.out.println(list);
for(int i=0; i<list.size(); i++)
    System.out.println(list.get(i));
int loc = list.indexOf("done");
for(int i=0; i<list.size(); i++)
        System.out.println("item␣"+(i+1)+":␣"+list.get(i));
System.out.println();

for(int i=0; i<list.size(); i++)
        System.out.println("First␣character␣of␣item␣"+(i+1)+":␣"+list.get(i).charAt
System.out.println();

int count = 0;
for(int i=0; i<list.size(); i++)
        if(list.get(i).substring(0, 1).equalsIgnoreCase("a"))
                count++;
System.out.println(count+"␣items␣begin␣with␣'a'");

count = 0;
for(int i=0; i<list.size(); i++)
        if(list.get(i).length() > 3)
                count++;
System.out.print(count+"␣items␣are␣longer␣than␣3␣characters:␣");

LinkedList<String> longOnes = new LinkedList<String>();
for(int i=0; i<list.size(); i++)
        if(list.get(i).length() > 3)
                longOnes.add(list.get(i));
System.out.println(longOnes);

String firstChars = "";
for(int i=0; i<list.size(); i++)
        firstChars+=list.get(i).charAt(0);
System.out.println("String␣of␣first␣characters:␣"+firstChars);

String lastChars = "";
for(int i=0; i<list.size(); i++)
        lastChars+=list.get(i).charAt(list.get(i).length()-1);
System.out.println("String␣of␣last␣characters:␣"+lastChars);

for(int i=0; i<list.size(); i++)
        list.set(i, list.get(i).toUpperCase());
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.

```java
public class MyArrayList {
	private double[] data;
	private int nItems;
	...
	/**
	remove first item in the list, if any and
	return it from the method

	return null if list is empty
	*/
	public Double removeFirst() {
		if(nItems == 0)
			return null;
		double removedItem = data[0];
		for(int i=1; i < nItems; i++) {
			data[i-1] = data[i];
		}
		nItems--;
		return removedItem;
	}
}
```

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.

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

**Solution:**

```java
public static double product(ArrayList <Double> list) {
	if(list == null || list.size() == 0)
		return 0;
	double result = 1;
	for(Double item: list)
		result*=item;
	return result;
}
```

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.

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

**Solution:**

```java
public static int sumPositive(ArrayList <Integer> list) {
	if(list == null || list.size() == 0)
		return 0;
	int result = 0;
	for(Integer item: list)
		if(item > 0)
			result+=item;
	return result;
}
```

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`.

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

**Solution:**

```java
int count(ArrayList<Integer> list, Integer target) {
        if(list == null)
                return 0;
        int result = 0;
        for(Integer item: list)
                if(item == target)
                        result++;
        return result;
}
```

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:**

```java
public static char[] toArray(ArrayList <Character> list) {
        if(list == null)
                return null;
        char[] result = new char[list.size()];
                int i = 0;
                for(Character item: list) {
                        result[i] = item;
                        i++;
                }
        return result;
}
```

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)`

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

**Solution:**

```java
void squared(ArrayList<Integer> list) {
        if(list == null)
                return;

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

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:**

```java
public static ArrayList<Integer> exclusiveItems(ArrayList<ArrayList<Integer>> megaL
        ArrayList<Integer> result = new ArrayList();
        for(int i=0; i < megaList.size(); i++) {
                for(Integer item: megaList.get(i)) {
                        boolean dup = false;
                        for(int k=0; k < megaList.size() && !dup; k++) {
                                if(i != k && megaList.get(k).contains(item)) {
                                        dup = true;
                                }
                        }
                        if(!dup)
                                result.add(item);
                }
        }
        return result;
}
```

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 | $O(1)$ | $O(n)$ |