U7L1: Introducing ArrayLists

Why Use ArrayList?

The ArrayList class provides several abstractions not available with plain Java arrays. ArrayLists are flexible, dynamic, and often easier to work with.

Advantages over Arrays

- We don't need to declare a size up front. We may add as many elements as we need.
- We can insert elements into a specific location and the elements will shift up an index to make room.
- We can remove elements at a specific location and the remaining elements will shift down an index.
- It has a built in toString(), no need to use `Arrays.toString()`

Importing ArrayList

ArrayList is included in the `java.util` package of the standard library. To import:

```
1 import java.util.ArrayList;
```

Side Note: As with all Java imports, we aren't actually importing the ArrayList code. The import statement allows us to reference the type `ArrayList` without having to type the fully qualified name everytime `java.util.ArrayList`.

with import

```
1 import java.util.ArrayList;
2 public class Demo{
3     ArrayList list;
```

without import

```
1 import java.util.ArrayList;
2 public class Demo{
3     java.util.ArrayList list;
```

ArrayList is a Generic Class

Generic Classes are classes that are designed to work with any type of reference object. When declaring a Generic type we should always include the *type* of object it will work with inside a pair of <>. You can think of this as a *datatype parameter* being passed to the Class. In the context of `ArrayList`, the type will tell Java what type of references the list will store.

Declaring a list of Strings

```
1 ArrayList<String> names;
```

Generic classes such as ArrayList are designed to work with references, **NOT** primitives. For this reason we must use the *Wrapper Class* names when dealing with primitive data.

Declaring a list of Integers

```
1 ArrayList<Integer> scores;
```

Initializing an ArrayList

To initialize an array list we will use the followin simple constructor. NOTe: The list will intially be empty.

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

Recent versions of Java (8+)allow us to use the empty "diamond" as long as the type is declared before the variable

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

Adding Elements

To add names to our list we will use the `add()` method, names will automatically be added to the next available position in the list.

```
1 names.add("Shaggy");
2 names.add("Scooby");
```

names

0	1
"Shaggy"	"Scooby"

Inserting Elements

The `add()` method is overloaded. If we pass two parameters as shown we can place the element at a specific place.

names before

0	1
"Shaggy"	"Scooby"

```
1 names.add(1, "Velma");
```

names after

0	1	2
"Shaggy"	"Velma"	"Scooby"

notice that "Scooby" shifted to the right

Removing Elements

We can also remove elements at a specific location by using the `remove()` method.

names before

0	1	2
"Shaggy"	"Velma"	"Scooby"

```
1 names.remove(0);
```

names after

0	1
"Velma"	"Scooby"

notice "Velma" and "Scooby" shifted to the left.

Getting and Setting Elements

To access an element in an ArrayList we use the `.get()` method.

```
1 String str = names.get(1); // assigns "Velma" to str
```

To set an element in an ArrayList we use the `.set()` method.

names before

0	1	2
"Shaggy"	"Velma"	"Scooby"

```
1 names.set(2, "Fred");
```

names after

0	1	2
"Shaggy"	"Velma"	"Fred"

NOTE: We don't use [] to access values like we do with arrays.

Getting the Size of a List

To get the number of elements of an ArrayList we use the `.size()` method.
NOTE: We don't use the `length` attribute like we do with arrays.

names

0	1	2
"Shaggy"	"Velma"	"Fred"

```
1 System.out.println(names.size()); // prints 3
```

Factor Array Demo

```
 1 ArrayList<Integer> factors = new ArrayList<>();
 2
 3 System.out.print("Enter a number: ");
 4 int n = scan.nextInt();
 5 // make a loop that goes through numbers less than n and
 6 // checks if they are factors...then add them to the factor list
 7 for(int j = 1;  j <= n; j++) {
 8     if(n % j == 0) {
 9         factors.add(j);
10     }
11 }
12 System.out.println(factors);
```

LAB 024 - Deli Order

Write a program that keeps track of the order that customers that will be served in a Deli line. The program should include the following.

- a static ArrayList that holds the customers names.
- a public static final String VIP that holds the name of your most special customer! :)
- a `printNames()` static method that prints all customer names on a separate line.
- initialize the list array but DON'T populate it. The user will fill it in the main method.

continued on next page...

LAB 024 - Deli Order Main Method

- Use a Scanner to do the following inside of `main()`
  - Loop repeatedly
  - Ask the User to choose one of the following options
    1. Complete an order
       - remove the customer at position 0. (validate)
    2. Remove a customer from the line
       - ask the user what index to remove (validate input) and use the `remove()` method to take them off the list.
    3. Add a new customer to the line.
       - If its the VIP, move them to position 1 (assume position 0 is currently being served)
       - everyone else goes to the back of the line.
    4. Give the user an option to quit
  - print the list after each iteration