

<https://runestone.academy/runestone/books/published/apcsareview/ListBasics/listbasics.html>

Answers are in **Bold**

Chapter 9: List and ArrayList

9.3, 9.4, 9.5, 9.6 Interfaces and Lists:

Lists in Java are made through what is called an interface. When using the interface, it is just a special type of class that uses public abstract methods. This means that they have only a header, and no code in the methods. It can be thought of as a contract, meaning a class can use the list class as long as they provide the code for its methods. On the exam there are 6 list methods that are needed. `Int size()`, `boolean add(E, obj)`, `void add(int index, E obj)`, `E get(int index)`, `E set(int index, E obj)`, `E remove(int index)`. Note that lists **ONLY hold objects, they don't hold primitive types, but you can add ints by wrapping them in Integer objects. Lists are useful over arrays because they are not of definite size.**

9.7 The ArrayList Class

ArrayList is a class that essentially turns an array into a list. There are other classes that do similar things, implementing the List class and methods for it under something, but the only one needed for the exam is ArrayList.

9.8 The Import Statement

In order to use ArrayList in Java, you need to import the package, packages are just multiple classes. You can do this by either importing the entire package, or just the class from the package. Ex: `import java.util.List`; or `import java.util.*`; to import the entire package. Import statements **MUST be imported before any other code, including class header.**

8-3-1: Which of the following is true about import statements?

- ☐ A. You can only have one import statement in a source file.
- ☐ B. You must specify the class to import.
- ☒ C. Import statements must be before other code in a Java source file.
- ☐ D. You must import `java.lang.String` to use the short name of `String`.

Check Me

Compare me


✓ Import statements have to be the first Java statements in a source file.

Activity: 1 -- Multiple Choice (qlib_1)

9.9, 9.10 Declaring and Creating Lists

When declaring lists in Java using ArrayList, you use `List<Type> name = null`; where you replace Type with whatever primitive type you will be placing inside the list. But this only declares it like with arrays, you also need to create the list using `new ArrayList<Type>()`; So

in order to fully create and use a list in Java you must use a command like `List<Type> nameList = new ArrayList<Type>();`



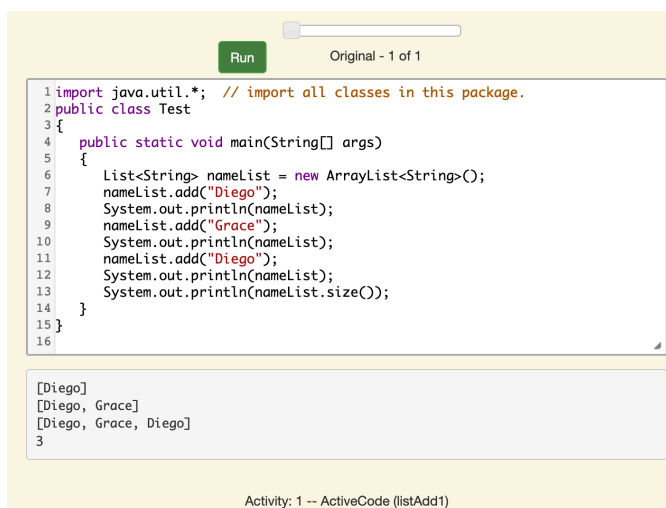
```
1 import java.util.*; // import everything at this level
2 public class Test
3 {
4     public static void main(String[] args)
5     {
6         List<String> nameList = new ArrayList<String>();
7         System.out.println("The size of nameList is: " + nameList.size());
8         List<String> list2 = null;
9         System.out.println("The size of list2 is: " + list2.size());
10    }
11 }
12
```

The size of nameList is: 0

For example to make an integer list I could use `List<Integer> numList = new ArrayList<Integer>();`

9.11 Adding Values to a List

Using the add(obj) statement, you can add values to a list. There is another add method add(index, obj) that adds the passed object at the index specified, moving all other values up one index.



```
1 import java.util.*; // import all classes in this package.
2 public class Test
3 {
4     public static void main(String[] args)
5     {
6         List<String> nameList = new ArrayList<String>();
7         nameList.add("Diego");
8         System.out.println(nameList);
9         nameList.add("Grace");
10        System.out.println(nameList);
11        nameList.add("Diego");
12        System.out.println(nameList);
13        System.out.println(nameList.size());
14    }
15 }
16
```

[Diego]
[Diego, Grace]
[Diego, Grace, Diego]
3

Activity: 1 -- ActiveCode (listAdd1)

Note

Lists can only hold objects, not primitive values. This means that `int` values must be wrapped into `Integer` objects to be stored in a list. You can do this using `new Integer(value)` as shown above. You can also just put an `int` value in a list and it will be changed into an `Integer` object automatically. This is called **autoboxing**. When you pull an `int` value out of a list of `Integers` that is called **unboxing**.

9.11 Check your Understanding

8-5-1: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();  
list1.add(new Integer(1));  
list1.add(new Integer(2));  
list1.add(new Integer(3));  
list1.add(2, new Integer(4));  
list1.add(new Integer(5));  
System.out.println(list1);
```

- ☐ A. [1, 2, 3, 4, 5]
☐ B. [1, 4, 2, 3, 5]
☒ C. [1, 2, 4, 3, 5]
☐ D. [1, 2, 4, 5]

Check Me

Compare me

✓ The `add(2, new Integer(4))` will put the 4 at index 2, but first move the 3 to index 3.

Activity: 4 -- Multiple Choice (qalAdd1)

8-5-2: What will print when the following code executes?

```
List<String> list1 = new ArrayList<String>();  
list1.add("Anaya");  
list1.add("Layla");  
list1.add("Sharrie");  
list1.add(1, "Sarah");  
System.out.println(list1);
```

- ☒ A. ["Anaya", "Sarah", "Layla", "Sharrie"]
☐ B. ["Anaya", "Layla", "Sharrie", "Sarah"]
☐ C. ["Sarah", "Anaya", "Layla", "Sharrie"]
☐ D. ["Anaya", "Layla", "Sarah", "Sharrie"]

Check Me

Compare me

✓ The `add(1, "Sarah")` will move any current items to the right and then put "Sarah" at index 1.

Activity: 5 -- Multiple Choice (qalAdd2)

8-5-3: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();  
list1.add(5);  
list1.add(4);  
list1.add(3);  
list1.add(1, 2);  
System.out.println(list1);
```

- ☐ A. [5, 4, 3, 2]
☐ B. [5, 4, 1, 3]
☐ C. [2, 5, 4, 3]
☒ D. [5, 2, 4, 3]

Check Me

Compare me

✓ This adds the 2 to index 1, but first moves all other values past that index to the right.

Activity: 6 -- Multiple Choice (qalAdd3)

8-5-4: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();  
list1.add(1);  
list1.add(3);  
list1.add(2);  
list1.add(1);  
System.out.println(list1);
```

- ☐ A. [1, 3, 2]
☒ B. [1, 3, 2, 1]
☐ C. [1, 1, 2, 3]
☐ D. [1, 2, 3]

Check Me

Compare me

✓ The add method adds each object to the end of the list and lists can hold duplicate objects.

Activity: 7 -- Multiple Choice (qalAdd4)

9.12 Getting and Setting Values in a List

You can retrieve an object at an index using `obj = listName.get(index)` and you can set an object to an index using `listName.set(index, obj)`. This is different from an array because lists are an object, so you have to use the method `.get()` or `.set()` instead of just doing `arrayName[]`

Run

Original - 1 of 1

```
1 import java.util.*; // import all classes in this package.
2 public class Test
3 {
4     public static void main(String[] args)
5     {
6         List<String> nameList = new ArrayList<String>();
7         nameList.add("Diego");
8         nameList.add("Grace");
9         nameList.add("Deja");
10        System.out.println(nameList);
11        System.out.println(nameList.get(0));
12        System.out.println(nameList.get(1));
13        System.out.println(nameList.get(2));
14        nameList.set(1, "John");
15        System.out.println(nameList);
16    }
17 }
18 }
19 }
```

[Diego, Grace, Deja]
Diego
Grace
Deja
[Diego, John, Deja]

Activity: 1 -- ActiveCode (listGetSet)

9.13 Removing an Object at an Index

You can also remove an object using `remove(index)` which returns the object and then shifts all other items back -1 index.

9.12 and 9.13 Check your Understanding

8-6-1: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();
list1.add(new Integer(1));
list1.add(new Integer(2));
list1.add(new Integer(3));
list1.set(2, new Integer(4));
list1.add(2, new Integer(5));
list1.add(new Integer(6));
System.out.println(list1);
```

- ☐ A. [1, 2, 3, 4, 5]
☐ B. [1, 2, 4, 5, 6]
☒ C. [1, 2, 5, 4, 6]
☐ D. [1, 5, 2, 4, 6]

Check Me

Compare me

✓ The `set` will change the item at index 2 to 4. The add of 5 at index 2 will move everything else to the right and insert 5. The last `add` will be at the end of the list.

8-6-2: What will print when the following code executes?

```
List<String> list1 = new ArrayList<String>();  
list1.add("Anaya");  
list1.add("Layla");  
list1.add("Sharrie");  
list1.set(1, "Destini");  
list1.add(1, "Sarah");  
System.out.println(list1);
```

- ☐ A. ["Sarah", "Destini", "Layla", "Sharrie"]
☐ B. ["Sarah", "Destini", "Anaya", "Layla", "Sharrie"]
☐ C. ["Anaya", "Sarah", "Sharrie"]
☒ D. ["Anaya", "Sarah", "Destini", "Sharrie"]

Check Me

Compare me

✓ The list is first ["Anaya", "Layla", "Sharrie"] and then changes to ["Anaya", "Destini", "Sharrie"] and then to ["Anaya", "Sarah", "Destini", "Sharrie"]

Activity: 4 -- Multiple Choice (qListRem2)

8-6-3: What will print when the following code executes?

```
List<Integer> numList = new ArrayList<Integer>();  
numList.add(new Integer(1));  
numList.add(new Integer(2));  
numList.add(new Integer(3));  
numList.set(2, new Integer(4));  
numList.add(1, new Integer(5));  
numList.add(new Integer(6));  
System.out.println(numList);
```

- ☐ A. [1, 2, 3, 4, 5]
☐ B. [1, 2, 4, 5, 6]
☐ C. [1, 2, 5, 4, 6]
☒ D. [1, 5, 2, 4, 6]

Check Me

Compare me

✓ **add** without an index adds at the end, **set** will replace the item at that index, **add** with an index will move all current values at that index or beyond to the right.

Activity: 5 -- Multiple Choice (qListRem3)

8-6-4: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();  
list1.add(new Integer(1));  
list1.add(new Integer(2));  
list1.add(new Integer(3));  
list1.remove(1);  
System.out.println(list1);
```

- ☐ A. [2, 3]
☐ B. [1, 2, 3]
☐ C. [1, 2]
☒ D. [1, 3]

Check Me

Compare me

✓ The item at index 1 is removed and the 3 is moved left.

Activity: 6 -- Multiple Choice (qListRem4)

8-6-5: What will print when the following code executes?

```
List<Integer> list1 = new ArrayList<Integer>();  
list1.add(new Integer(1));  
list1.add(new Integer(2));  
list1.add(new Integer(3));  
list1.remove(2);  
System.out.println(list1);
```

- ☐ A. [2, 3]
☐ B. [1, 2, 3]
☒ C. [1, 2]
☐ D. [1, 3]

Check Me

Compare me

✓ The 3 (at index 2) is removed

9.14 Looping Through a List

For-Each loops can be used on lists just as they can on arrays. While and For loops can be used as well, but be careful if you are adding or removing items, as it shifts the index ± 1 of elements in the index.

Run

Original - 1 of 1

```
1 import java.util.*; // import all classes in this package.
2 public class Test
3 {
4     public static void main(String[] args)
5     {
6         List<Integer> myList = new ArrayList<Integer>();
7         myList.add(50);
8         myList.add(30);
9         myList.add(20);
10        int total = 0;
11        for (Integer value: myList)
12        {
13            total = total + value;
14        }
15        System.out.println(total);
16    }
17 }
18
```

100

Activity: 1 -- ActiveCode (listForEachLoop)

Run

Original - 1 of 1

```
1 import java.util.*; // import all classes in this package.
2 public class ListWorker
3 {
4     private List<String> nameList;
5
6     public ListWorker(List<String> theNames)
7     {
8         nameList = theNames;
9     }
10
11    public boolean removeName(String name)
12    {
13        boolean found = false;
14        int index = 0;
15        while (index < nameList.size())
16        {
17            if (name.equals(nameList.get(index)))
18            {
19                nameList.remove(index);
20                found = true;
21            }
22            else index++;
23        }
24        return found;
25    }
26 }
```

[Amun, Ethan, Donnie, Ethan]
[Amun, Donnie]

Activity: 2 -- ActiveCode (listForEachLoopObj)

9.14 Check your Understanding

8-7-1: Assume that `nums` has been created as an `ArrayList` object and it initially contains the following `Integer` values [0, 0, 4, 2, 5, 0, 3, 0]. What will `nums` contain as a result of executing `numQuest` ?

```
List<Integer> list1 = new ArrayList<Integer>();
private List<Integer> nums;

// precondition: nums.size() > 0;
// nums contains Integer objects
public void numQuest()
{
    int k = 0;
    Integer zero = new Integer(0);
    while (k < nums.size())
    {
        if (nums.get(k).equals(zero))
            nums.remove(k);
        k++;
    }
}
```

- ☒ A. [0, 4, 2, 5, 3]
- ☐ B. [3, 5, 2, 4, 0, 0, 0, 0]
- ☐ C. [0, 0, 0, 0, 4, 2, 5, 3]
- ☐ D. [4, 2, 5, 3]

[Check Me](#)[Compare me](#)

✓ Incrementing the index each time through the loop will miss when there are two zeros in a row.

Activity: 3 -- Multiple Choice (qloopList_1)

8-7-2: Which of the following is a reason to use a list (assume an object of the class `ArrayList`) instead of an array?

- ☐ A. A list will always use less memory than an array.
- ☐ B. A list can store objects, but arrays can only store primitive types.
- ☐ C. A list has faster access to the last element than an array.
- ☒ D. A list resizes itself as necessary as items are added, but an array does not.

[Check Me](#)[Compare me](#)

✓ An `ArrayList` is really a dynamic array (one that can grow or shrink as needed).

Activity: 4 -- Multiple Choice (qloopList_2)