

Mind the Gap

ArrayList traversals



From array to list

PAUSE AND TRY

crew is now an `ArrayList<String>`. What two things must change?

THE ARRAY VERSION

```
String[] crew = {"Ada", "Bo", "Cy", "Di"};
for (int i = 0; i < crew.length; i++) {
    System.out.println(crew[i]);
}
```

Hint: how does a list report its length? How do you read element *i*?

By the end of class you can...

- 1 Traverse a list with an indexed for loop or a while loop: forward, backward, or by steps.
- 2 Keep the index between 0 and `size() - 1`, and fix a loop that crashes.
- 3 Traverse with an enhanced for loop, and know what it can and cannot change.
- 4 Remove elements during a traversal without skipping any.
- 5 Explain why changing the size inside an enhanced for loop is not allowed.

1

PART 1

Visiting by index

The indexed for loop

LISTLOOP.JAVA (MAIN)

```
ArrayList<String> crew =  
    new ArrayList<String>();  
crew.add("Ada");  
crew.add("Bo");  
crew.add("Cy");  
crew.add("Di");  
for (int i = 0; i < crew.size(); i++) {  
    System.out.println(i + ": " + crew.get(i));  
}
```



Traversing a list means using a loop to visit all of its elements, or an ordered run of them.

Array	ArrayList
crew.length	crew.size()
crew[i]	crew.get(i)
crew[i] = x;	crew.set(i, x);

OUTPUT

```
0: Ada  
1: Bo  
2: Cy  
3: Di
```

Backward, and every other one

PLANETS.JAVA (MAIN)

```
ArrayList<String> p =  
    new ArrayList<String>();  
p.add("Ana");  
p.add("Ben");  
p.add("Cy");  
p.add("Dev");  
p.add("Eve");  
for (int i = p.size() - 1; i >= 0; i--) {  
    System.out.print(p.get(i) + " ");  
}  
System.out.println();  
for (int i = 0; i < p.size(); i += 2) {  
    System.out.print(p.get(i) + " ");  
}  
System.out.println();
```

OUTPUT

```
Eve Dev Cy Ben Ana  
Ana Cy Eve
```

Loop	Visits
backward	4 3 2 1 0
i += 2	0 2 4

Backward starts at **size() - 1**, not size().

One step too far



An index outside **0** to **size() - 1** throws an **IndexOutOfBoundsException**.

```
for (int i = 0; i <= laps.size(); i++) {  
    System.out.println(laps.get(i));  
}
```

laps

61	58	60	3 ?
0	1	2	

OUTPUT, THEN THE CRASH

```
61  
58  
60  
Exception in thread "main"  
java.lang.IndexOutOfBoundsException:  
Index 3 out of bounds for length 3
```

A while loop walks too

PAUSE AND TRY

Write the exact output. Does it crash?

```
int i = 1;
while (i < bugs.size()) {
    System.out.println(i + " " + bugs.get(i));
    i += 2;
}
```

Track *i* on paper: what values does it take, and when does the test fail?

bugs holds [ant, bee, fly, moth, wasp].



2

PART 2

The enhanced for loop

Every element, no index

FOREACH.JAVA (MAIN)

```
ArrayList<String> crew =  
    new ArrayList<String>();  
crew.add("Ada");  
crew.add("Bo");  
crew.add("Cy");  
for (String name : crew) {  
    System.out.println("Hi, " + name);  
}
```

OUTPUT

```
Hi, Ada  
Hi, Bo  
Hi, Cy
```



for (String name : crew)

Use it when you only need to **read** each element in order. Use an indexed loop when you need the position, want to go backward or skip around, or want to replace elements.

The loop variable is a copy



Assigning to an enhanced for loop variable **never changes the list**. To replace elements, use **set(i, ...)** in an indexed loop.

SHOUT.JAVA (MAIN)

```
ArrayList<String> words =  
    new ArrayList<String>();  
words.add("hop");  
words.add("skip");  
for (String w : words) {  
    w = w + "!";  
}  
System.out.println(words);  
for (int i = 0; i < words.size(); i++) {  
    words.set(i, words.get(i) + "!");  
}  
System.out.println(words);
```

OUTPUT

```
[hop, skip]  
[hop!, skip!]
```

Line 1: the for-each loop changed only **w**.

Line 2: **set** changed the list.

Long snacks only

PAUSE AND TRY

Write the exact output.

```
for (String s : snacks) {  
    if (s.length() > 4) {  
        System.out.println(s);  
    }  
}  
System.out.println(snacks.size());
```

snacks holds [chips, nuts, pretzels, figs].

A white rounded square with a teal number '3' inside, tilted slightly to the right.

3

PART 3

Removing while you walk

A forward loop skips

REMOVE THE NEGATIVES

```
for (int i = 0; i < vals.size(); i++) {  
    if (vals.get(i) < 0) {  
        vals.remove(i);  
    }  
}  
System.out.println(vals);
```

OUTPUT

[5, -7, 4]



After **remove(i)**, the next element slides into index i. Then **i++** jumps past it.

i	List before the pass	get(i)	Action
0	[5, -2, -7, 4]	5	keep
1	[5, -2, -7, 4]	-2	remove
2	[5, -7, 4]	4	keep (-7 skipped!)
3	[5, -7, 4]		3 < 3 is false

Two ways not to skip

FIX 1: WALK BACKWARD

```
for (int i = vals.size() - 1; i >= 0; i--) {  
    if (vals.get(i) < 0) {  
        vals.remove(i);  
    }  
}  
System.out.println(vals);
```

OUTPUT

[5, 4]

Removing during a traversal needs a **special technique** so nothing is skipped.

FIX 2: MOVE ON ONLY IF KEPT

```
int i = 0;  
while (i < vals.size()) {  
    if (vals.get(i) < 0) {  
        vals.remove(i);  
    } else {  
        i++;  
    }  
}  
System.out.println(vals);
```

OUTPUT

[5, 4]

X marks the skip

PAUSE AND TRY

The loop should remove every "x". What does it print instead?

```
for (int i = 0; i < marks.size(); i++) {  
    if (marks.get(i).equals("x")) {  
        marks.remove(i);  
    }  
}  
System.out.println(marks);
```

Then: change the loop header so every x is removed.

marks holds [ok, x, x, ok, x].

A large, white, rounded square containing a bold, dark teal number '4'.

4

PART 4

Hands off the size

Changing the size in a for-each loop



Adding or removing inside an enhanced for loop can throw a **ConcurrentModificationException**.

CRASHES

```
for (String t : tasks) {  
    System.out.println("Done: " + t);  
    tasks.remove(0);  
}  
System.out.println("All done");
```

OUTPUT, THEN THE CRASH

```
Done: dishes  
Exception in thread "main"  
java.util.ConcurrentModificationException
```

WORKS

```
while (tasks.size() > 0) {  
    String t = tasks.remove(0);  
    System.out.println("Done: " + t);  
}  
System.out.println("All done");
```

OUTPUT

```
Done: dishes  
Done: trash  
Done: laundry  
All done
```

Which loop for the job?

The job	Use	Why
Read every element, in order	enhanced for	simplest; can't go out of bounds
Replace elements	indexed for + set	the loop variable is only a copy
Go backward or skip around	indexed for or while	you control the index
Remove some elements	backward for, or while	a forward for loop skips
Add or remove anything	never enhanced for	ConcurrentModificationException

Words to know

Traverse

Use a loop to visit all of a list's elements, or an ordered run of them.

Enhanced for loop

for (String s : list): each element in order, no index.

Last index

size() - 1. The first is 0.

IndexOutOfBoundsException

Thrown when get, set, or remove gets an index outside 0 to size() - 1.

Skipping

A forward loop misses the element that slides left after a remove.

ConcurrentModificationException

Can be thrown when a list changes size during an enhanced for loop.

Mind the gap



Explain to your partner why a backward loop can remove elements without skipping any.

SENTENCE STARTER

When I remove at index i , only the elements _____ i move, and going backward I have _____.

Next: the Mind the Gap puzzle worksheet.