

List Surgery

Implementing ArrayList algorithms



Same algorithm, new container

PAUSE AND TRY

Rewrite this for an `ArrayList<Integer>` named `scores`. Which pieces change?

TOPIC 4.5: MAXIMUM OF AN ARRAY

```
int[] scores = {82, 95, 77, 90};
int best = scores[0];
for (int i = 1; i < scores.length; i++) {
    if (scores[i] > best) {
        best = scores[i];
    }
}
```

Circle every piece that has to change. Underline the pieces that stay exactly the same.

By the end of class you can...

1

Find a sum, average, minimum, or maximum of an ArrayList, and count, check "any", or check "all".

2

Visit consecutive pairs without going out of bounds, and check a list for duplicates.

3

Insert while traversing without looping forever.

4

Delete while traversing without skipping an element.

5

Shift, rotate, reverse, and filter a list, and walk two collections side by side.

A large, white, rounded square containing the number '1' in a bold, black, sans-serif font, tilted slightly to the right.

1

PART 1

Old algorithms, new list

Sum, average, and maximum

SCORES.JAVA

```
ArrayList<Integer> scores = new ArrayList<Integer>();
scores.add(82); scores.add(95);
scores.add(77); scores.add(90);
int total = 0;
int best = scores.get(0);
for (int i = 0; i < scores.size(); i++) {
    total += scores.get(i);
    if (scores.get(i) > best) {
        best = scores.get(i);
    }
}
double avg = (double) total / scores.size();
System.out.println("Average: " + avg);
System.out.println("Best: " + best);
```

Array	ArrayList
arr.length	list.size()
arr[i]	list.get(i)
arr[i] = v;	list.set(i, v);

OUTPUT

```
Average: 86.0
Best: 95
```

An average divides by **size()**, so the list must not be empty.

Start a maximum at a real element



A made-up start like `0` works only by luck. Start at `list.get(0)` and compare the rest.

FREEZER.JAVA

```
ArrayList<Integer> cold = new ArrayList<Integer>();  
cold.add(-18); cold.add(-21);  
cold.add(-16);  
int highest = 0;  
for (int t : cold) {  
    if (t > highest) {  
        highest = t;  
    }  
}  
System.out.println(highest);
```

OUTPUT

0

0 is not even in the list.

THE FIX

```
int highest = cold.get(0);
```

OUTPUT

-16

Any, all, and how many

RIDES.JAVA

```
ArrayList<Integer> kids = new ArrayList<Integer>();
kids.add(52); kids.add(47);
kids.add(60); kids.add(44);
boolean anyShort = false;
boolean allTall = true;
int count = 0;
for (int h : kids) {
    if (h < 48) { anyShort = true; }
    if (h < 42) { allTall = false; }
    if (h >= 50) { count++; }
}
System.out.println(anyShort);
System.out.println(allTall);
System.out.println(count);
```

Question	Starts	Changes when
Any?	false	one HAS it
All?	true	one LACKS it
How many?	0	+1 per match

OUTPUT

```
true
true
2
```

A flag switches **once** and never switches back.

Is every battery charged?

PAUSE AND TRY

What does this print? Is that the right answer? Fix it.

```
ArrayList<Integer> cells = new ArrayList<Integer>();  
cells.add(80); cells.add(15); cells.add(90);  
boolean allCharged = true;  
for (int b : cells) {  
    if (b < 20) {  
        allCharged = false;  
    } else {  
        allCharged = true;  
    }  
}  
System.out.println(allCharged);
```

Hint: trace allCharged after each of the three elements.

A large, white, rounded square containing the number '2' in a bold, teal, sans-serif font.

2

PART 2

Two elements at a time

Consecutive pairs stop one early

HIKE.JAVA

```
ArrayList<Integer> hike = new ArrayList<Integer>();  
hike.add(120); hike.add(135); hike.add(130);  
hike.add(150); hike.add(145);  
int drops = 0;  
for (int i = 0; i < hike.size() - 1; i++) {  
    if (hike.get(i + 1) < hike.get(i)) {  
        drops++;  
    }  
}  
System.out.println("Drops: " + drops);
```

OUTPUT

Drops: 2

PAIRS IN A LIST OF 5



4 pairs: i = 0, 1, 2, 3



Use **i < list.size() - 1** so **get(i + 1)** stays in range. Or start at 1 and look back at **get(i - 1)**.

Is any username repeated?

USERS.JAVA

```
ArrayList<String> users = new ArrayList<String>();
users.add("kai"); users.add("rio");
users.add("zoe"); users.add("rio");
for (int i = 0; i < users.size(); i++) {
    for (int j = i + 1; j < users.size(); j++) {
        if (users.get(i).equals(users.get(j))) {
            System.out.println(users.get(i));
        }
    }
}
```

OUTPUT

rio



Inner loop starts at **j = i + 1**: never compare an element with itself, and check each pair once.



Strings: use **equals**. Integer elements: **==** compares references, so unbox first: **int a = list.get(i);**

A large, white, rounded square containing the number '3' in a bold, teal, sans-serif font.

3

PART 3

Changing the list as you go

Inserting: step past what you added

WORKOUT.JAVA

```
ArrayList<String> plan = new ArrayList<String>();
plan.add("run"); plan.add("swim");
plan.add("run");
for (int i = 0; i < plan.size(); i++) {
    if (plan.get(i).equals("run")) {
        plan.add(i, "stretch");
        i++;
    }
}
System.out.println(plan);
```

OUTPUT

```
[stretch, run, swim, stretch, run]
```



add(i, value) pushes the element at i, and everything after it, one place right.



Without the extra **i++**, the loop finds the same "run" again and **never ends**.

Deleting forward skips an element

CLEANUP.JAVA

```
ArrayList<Integer> data = new ArrayList<Integer>();
data.add(0); data.add(0);
data.add(5); data.add(0);
for (int i = 0; i < data.size(); i++) {
    if (data.get(i) == 0) {
        data.remove(i);
    }
}
System.out.println(data);
```

OUTPUT

[0, 5]

i	get(i)	List after
0	0	[0, 5, 0]
1	5	[0, 5, 0]
2	0	[0, 5]



After **remove(i)**, the next element slides into index i. The loop's i++ jumps over it.

Two ways to delete safely

FIX 1: WALK BACKWARD

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

OUTPUT

[5]



Not in an enhanced for loop: changing the size there throws an exception.

FIX 2: I++ ONLY IF KEPT

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

OUTPUT

[5]

Trace a backward delete

PAUSE AND TRY

Write the list after each removal. What prints?

```
ArrayList<Integer> nums = new ArrayList<Integer>();  
nums.add(1); nums.add(3); nums.add(4);  
nums.add(5); nums.add(6);  
for (int i = nums.size() - 1; i >= 0; i--) {  
    if (nums.get(i) % 2 == 1) {  
        nums.remove(i);  
    }  
}  
System.out.println(nums);
```

i	List after
4	
3	
2	
1	
0	

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

4

PART 4

Move, flip, and filter

Shift and rotate with remove and add

LINE.JAVA

```
ArrayList<String> line = new ArrayList<String>();  
line.add("Mo"); line.add("Lu");  
line.add("Ty"); line.add("Jo");  
line.add(line.remove(0));  
System.out.println(line);  
line.remove(line.size() - 1);  
line.add(0, "-");  
System.out.println(line);
```

OUTPUT

```
[Lu, Ty, Jo, Mo]  
[-, Lu, Ty, Jo]
```



Rotate: the element that leaves one end comes back at the other.



Shift: one element falls off, and a new value fills the empty end.

Reverse: swap pairs, stop halfway

FLIP.JAVA

```
ArrayList<Integer> nums = new ArrayList<Integer>();  
for (int k = 1; k <= 6; k++) {  
    nums.add(k);  
}  
int last = nums.size() - 1;  
for (int i = 0; i < nums.size() / 2; i++) {  
    int temp = nums.get(i);  
    nums.set(i, nums.get(last - i));  
    nums.set(last - i, temp);  
}  
System.out.println(nums);
```

OUTPUT

```
[6, 5, 4, 3, 2, 1]
```



The partner of index **i** is **last - i**. Save one value in **temp** before the swap.



Stop at **size() / 2**. Going all the way swaps everything back.

Build a new list from two collections

SUPPLIES.JAVA

```
String[] items = {"pen", "pad", "ink", "tape"};
ArrayList<Double> prices = new ArrayList<Double>();
prices.add(1.5); prices.add(3.25);
prices.add(0.75);
ArrayList<String> cheap = new ArrayList<String>();
for (int i = 0; i < prices.size(); i++) {
    if (prices.get(i) < 2.0) {
        cheap.add(items[i]);
    }
}
System.out.println(cheap);
```

OUTPUT

```
[pen, ink]
```



Filtered list: add matches to a new list. The original stays whole.



Side by side: one index for both, bounded by the **shorter** collection.

Words to know

Consecutive pair

Two elements next to each other in a list.

Insert

`add(i, value)`: later elements move one place right.

Delete

`remove(i)`: later elements move one place left.

Shift / rotate

Move every element one place; a rotate wraps the end around.

Reverse

Swap pairs from the outside in, stopping halfway.

Filtered list

A new list of only the elements that pass a test.

Insert or delete?



Explain to your partner why inserting and deleting during a loop need opposite fixes.

SENTENCE STARTER

add(i, value) moves elements _____, so I _____; remove(i) moves them _____, so I _____.

Next: the List Surgery puzzle worksheet.