

Sort It Out

Sorting algorithms



Sort five cards

PAUSE AND TRY

Put these in order, smallest first. Then describe exactly what you did.

34

19

8

27

15

Did you look for the smallest first?

Or take one card at a time and slide it into place?

By the end of class you can...

- 1 Say what it means for a list to be sorted, and check it in code.
- 2 Trace selection sort pass by pass: select the smallest, swap it into place.
- 3 Trace insertion sort pass by pass: shift larger values right, insert the saved one.
- 4 Read the Java for both sorts on an array or an ArrayList, including Strings.
- 5 Tell the two sorts apart from an in-between state, and compare their work.

A large, white, rounded square containing a large, bold, teal number '1'.

1

PART 1

What sorted means

Ascending: each element $\leq$ the next

```
int[] a = {3, 5, 5, 8};
boolean sorted = true;
for (int i = 0; i < a.length - 1; i++) {
    if (a[i] > a[i + 1]) {
        sorted = false;
    }
}
System.out.println("Sorted: " + sorted);
```

OUTPUT

Sorted: true

3

5

5

8

Repeats are fine.



Ascending order: smallest first, each element $\leq$ the next. **Descending** order: largest first.

2

PART 2

Selection sort

Selection sort: select the smallest, swap it in

Start	34	19	8	27	15
After pass 1	8	19	34	27	15
After pass 2	8	15	34	27	19
After pass 3	8	15	19	27	34
After pass 4	8	15	19	27	34



Each pass: find the **smallest** value in the unsorted part, then **swap** it with the first unsorted element.

Shaded values are in their **final** positions. Orange outlines mark the next swap.

Selection sort in Java

```
public static void selectionSort(int[] arr) {
    for (int i = 0; i < arr.length - 1; i++) {
        int minIndex = i;
        for (int j = i + 1; j < arr.length; j++) {
            if (arr[j] < arr[minIndex]) {
                minIndex = j;
            }
        }
        int temp = arr[i];
        arr[i] = arr[minIndex];
        arr[minIndex] = temp;
    }
}

// in main
int[] nums = {34, 19, 8, 27, 15};
selectionSort(nums);
for (int v : nums) { System.out.print(v + " "); }
```

OUTPUT

8 15 19 27 34



Reset **minIndex = i** every pass. The inner loop finds the smallest; the swap happens **once**, after the inner loop.

Selection sort swaps two values; it never slides



After one pass, exactly **two** elements have changed places. Everything else stays put.

Start



Right: swap



Only positions 0 and 2 changed.

Wrong: slide



34 and 19 slid right: that is not selection sort.

Your turn: selection sort

PAUSE AND TRY

Write the array after each of the first three passes.

40	25	10	35	20
----	----	----	----	----

After pass 1: _____

After pass 2: _____

After pass 3: _____

3

PART 3

Insertion sort

Insertion sort: shift right, insert

Start	34	19	8	27	15
After pass 1	19	34	8	27	15
After pass 2	8	19	34	27	15
After pass 3	8	19	27	34	15
After pass 4	8	15	19	27	34



Each pass takes the next value (orange), **shifts** larger sorted values one spot right, and **inserts** it in the gap.

The shaded part is sorted, but **not final**: a later, smaller value can push in front.

Insertion sort in Java

```
public static void insertionSort(int[] arr) {
    for (int i = 1; i < arr.length; i++) {
        int current = arr[i];
        int j = i - 1;
        while (j >= 0 && arr[j] > current) {
            arr[j + 1] = arr[j];
            j--;
        }
        arr[j + 1] = current;
    }
}

// in main
int[] nums = {34, 19, 8, 27, 15};
insertionSort(nums);
for (int v : nums) { System.out.print(v + " "); }
```

OUTPUT

8 15 19 27 34



Save **current** first. Check **j >= 0** before **arr[j]**.
Insert at **j + 1**.

Sorted so far is not the same as final



Insertion sort places each value in its correct spot **so far**, not its final spot.

Selection, after 2 passes



8 and 15 are the two smallest of all. Final.

Insertion, after 1 pass



19 and 34 are in order, but 8 will push in front of them on the next pass.

Your turn: insertion sort

PAUSE AND TRY

Write the array after passes 1, 2, and 3. Count the shifts.

35	15	25	5	45
----	----	----	---	----

After pass 1: _____

After pass 2: _____

After pass 3: _____

A large, bold, teal-colored number '4' is centered within a white rounded square with a slight drop shadow.

PART 4

ArrayLists and telling them apart

Insertion sort on an ArrayList of Strings

```
public static void sort(ArrayList<String> list) {  
    for (int i = 1; i < list.size(); i++) {  
        String current = list.get(i);  
        int j = i - 1;  
        while (j >= 0 && list.get(j).compareTo(current) > 0) {  
            list.set(j + 1, list.get(j));  
            j--;  
        }  
        list.set(j + 1, current);  
    }  
}
```

```
ArrayList<String> names = new ArrayList<String>();  
names.add("Pia"); names.add("Ben");  
names.add("Zara"); names.add("Cal");  
sort(names);  
System.out.println(names);
```

OUTPUT

```
[Ben, Cal, Pia,  
Zara]
```

Two sorts, side by side

	Selection sort	Insertion sort
Each pass	Find the smallest left, swap it in	Take the next value, shift, insert
What moves	Exactly two elements trade places	A block slides right one spot
Front part	Smallest of the whole list; final	Original front values, in order; not final
Back part	Can be rearranged by swaps	Untouched until reached
8 values, already sorted	28 comparisons	0 shifts

Which sort made this?

PAUSE AND TRY

Both started as 9 3 7 1 8 and ran 2 passes. Which sort made each?

A**B**

Words to know

Ascending

Smallest first: each element $\leq$ the next.

Pass

One run of a sort's outer loop.

Selection sort

Select the smallest left and swap it into its final spot.

Insertion sort

Shift larger sorted values right and insert the next value.

Swap

Two elements trade places, using a temporary variable.

Shift

Copy an element one spot right to open a gap.

Swap or shift?



Explain to your partner how one pass of selection sort differs from one pass of insertion sort.

SENTENCE STARTER

Selection sort ____ the smallest value into a ____ spot, but insertion sort ____ values right to make room.

Next: the Sort It Out puzzle worksheet.