

Work the Grid

Implementing 2D array algorithms



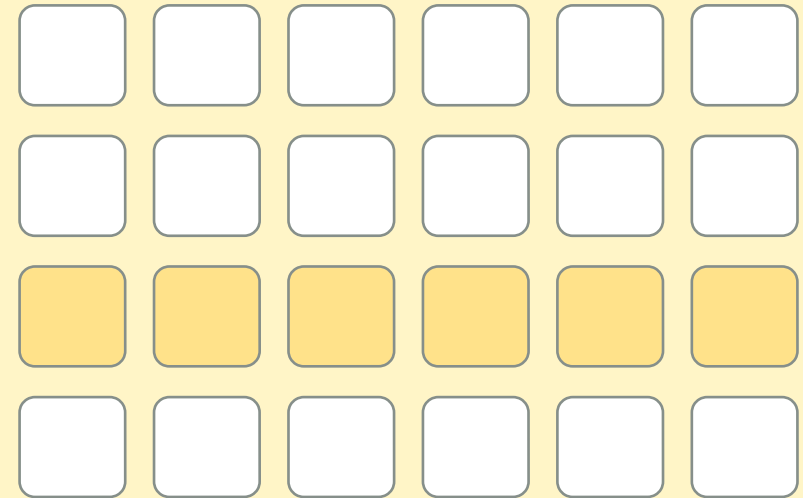
Questions about a grid

PAUSE AND TRY

Which seats would you need to look at to answer each question?

A THEATER: 4 ROWS OF 6 SEATS

1. How many tickets were sold in row 2?
2. Is any seat in the first column empty?
3. Is the whole theater sold out?



row 2

By the end of class you can...

- 1 Total or average a whole 2D array, one row, one column, or a smaller region.
- 2 Find the largest or smallest value and remember where it was.
- 3 Use a flag for "any?" and "all?" questions and a counter for "how many?".
- 4 Compare neighbors and detect duplicates in a row or column without going out of bounds.
- 5 Shift, rotate, and reverse one row or column, and trace each step.

A large white square with rounded corners, tilted slightly to the right, containing a large, bold, dark teal number '1'.

PART 1

Aim the loops

Fix one index, loop the other

Region	What stays fixed	Loop runs	Element
Whole array	nothing	<code>r < grid.length</code> <code>c < grid[r].length</code>	<code>grid[r][c]</code>
One row	the row index	<code>c < grid[row].length</code>	<code>grid[row][c]</code>
One column	the column index	<code>r < grid.length</code>	<code>grid[r][col]</code>
Main diagonal (square)	row equals column	<code>i < grid.length</code>	<code>grid[i][i]</code>



Changing where a loop **starts** and where it **stops** aims an algorithm at any rectangle of the array.

One day's total, one item's average

```
int[][] sales = { {12, 8, 15, 6},
                  {9, 14, 11, 7},
                  {10, 5, 13, 9} };

int day = 2;
int dayTotal = 0;
for (int c = 0; c < sales[day].length; c++) {
    dayTotal += sales[day][c];
}

int item = 1;
int itemTotal = 0;
for (int r = 0; r < sales.length; r++) {
    itemTotal += sales[r][item];
}

double avg = (double) itemTotal / sales.length;
System.out.println("Day 2 total: " + dayTotal);
System.out.println("Item 1 average: " + avg);
```

OUTPUT

Day 2 total: 37
Item 1 average: 9.0



Row 2 loop: **c** runs across, up to **sales[day].length**.

Column 1 loop: **r** runs down, up to **sales.length**. Cast to **double** before dividing.

A column has one element per row



Walking **down a column** with the number of columns as the bound runs off the bottom of the array.

```
int[][] sales = { {12, 8, 15, 6},  
                  {9, 14, 11, 7},  
                  {10, 5, 13, 9} };  
  
int item = 1;  
int itemTotal = 0;  
for (int r = 0; r < sales[0].length; r++) {  
    itemTotal += sales[r][item];  
}  
System.out.println("Item 1 total: " + itemTotal);
```

Fix: `r < sales.length`

Nothing prints: the crash comes before the println.

ERROR

```
java.lang.ArrayIndexOutOfBoundsException: Index 3 out of bounds for length 3
```

A large white square with rounded corners, tilted slightly to the right, containing the number '2' in a bold, dark teal font.

2

PART 2

Best, any, all, how many

The best so far, and where it was

```
int[][] scores = { {340, 560, 410},  
                  {720, 390, 650},  
                  {480, 810, 300} };  
  
int best = scores[0][0];  
int bestR = 0;  
int bestC = 0;  
for (int r = 0; r < scores.length; r++) {  
    for (int c = 0; c < scores[r].length; c++) {  
        if (scores[r][c] > best) {  
            best = scores[r][c];  
            bestR = r;  
            bestC = c;  
        }  
    }  
}  
  
System.out.println("Best: " + best);  
System.out.println("Row " + bestR + ", col " + bestC);
```

OUTPUT

Best: 810
Row 2, col 1



Start the best-so-far at an element **of the region** you search.

Save **r** and **c** at the same moment you save the value.

Flags answer yes or no; counters answer how many

Question	Variable	Starts	Changes when an element...
Does at least one?	boolean flag	false	has the property (set to true)
Do all of them?	boolean flag	true	does not have it (set to false)
How many?	int counter	0	has the property (count++)



Set it to its start **before all the loops** for one answer about the whole region. Set it **at the top of the outer loop** for a separate answer for each row.

Reading days over 30 minutes, week by week

READING LOG

```
int[][] minutes = { {20, 45, 35, 10},  
                    {50, 40, 30, 60},  
                    {15, 25, 5, 20} };  
  
for (int r = 0; r < minutes.length; r++) {  
    int count = 0;  
    for (int c = 0; c < minutes[r].length; c++) {  
        if (minutes[r][c] > 30) {  
            count++;  
        }  
    }  
    System.out.println("Week " + r + ": " + count);  
}
```

OUTPUT

```
Week 0: 2  
Week 1: 3  
Week 2: 0
```



A fresh **count** for every row, printed at the end of that row.

Try it: what prints if the counter moves above the outer loop?

Shut or open?

PAUSE AND TRY

What does this print? Which question does each flag answer?

```
int[][] lockers = { {1, 0, 0}, {0, 0, 1} }; // 1 = open
boolean colShut = true;
for (int r = 0; r < lockers.length; r++) {
    if (lockers[r][1] == 1) { colShut = false; }
}
boolean rowOpen = false;
for (int c = 0; c < lockers[1].length; c++) {
    if (lockers[1][c] == 1) { rowOpen = true; }
}
System.out.println(colShut + " " + rowOpen);
```

One line of output. Write both values.

Resetting the flag wipes out the answer



For one answer about the whole array, set the flag **once**, before both loops. A reset inside the outer loop keeps only the last row's answer.

IS ANY LOCKER OPEN?

```
String[][] lockers = { {"shut", "open", "shut"},  
                       {"shut", "shut", "shut"} };  
  
boolean anyOpen = false;  
for (String[] row : lockers) {  
    anyOpen = false;  
    for (String s : row) {  
        if (s.equals("open")) { anyOpen = true; }  
    }  
}  
  
System.out.println("Any open: " + anyOpen);
```

OUTPUT

Any open: false

Fix: delete the line `anyOpen = false;` inside the outer loop.

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

3

PART 3

Neighbors and repeats

Neighbor pairs stop one early

TEMPERATURE DROPS

```
int[][] temps = { {61, 70},
                  {64, 68},
                  {59, 73},
                  {66, 71} };

int col = 0;
int drops = 0;
for (int r = 0; r < temps.length - 1; r++) {
    if (temps[r + 1][col] < temps[r][col]) {
        drops++;
    }
}
System.out.println("Drops: " + drops);
```

OUTPUT

Drops: 1

Pairs run...	Bound
across a row	$c < \text{grid}[r].\text{length} - 1$
down a column	$r < \text{grid}.\text{length} - 1$

4 rows have only 3 neighbor pairs.

Duplicates: compare with every later element

REPEATED PICK?

```
int[][] picks = { {7, 3, 9, 3},  
                  {1, 2, 5, 8} };  
  
int row = 0;  
boolean dup = false;  
for (int i = 0; i < picks[row].length; i++) {  
    for (int j = i + 1; j < picks[row].length; j++) {  
        if (picks[row][i] == picks[row][j]) {  
            dup = true;  
        }  
    }  
}  
  
System.out.println("Row 0 repeats: " + dup);
```

OUTPUT

Row 0 repeats: true



Start $j = i + 1$. With $j = 0$, each element is compared with itself, and the answer is always true.

Count before you code

PAUSE AND TRY

One row of a 2D array has 6 elements.

A How many neighbor pairs does a pairs loop visit?

B How many comparisons does the duplicate check (j starts at $i + 1$) make?

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

4

PART 4

Move things around

Shift loses the end; rotate wraps it

Move left	Before	After
Shift	1 2 3 4	2 3 4 0
Rotate	1 2 3 4	2 3 4 1



1. Save the element that falls off (rotate).
2. Copy, starting at the end things move toward.
3. Fill the open spot.

```
int[][] lanes = { {3, 8, 1, 6, 4},  
                  {2, 7, 5, 9, 0} };  
  
int row = 0;  
int len = lanes[row].length;  
int first = lanes[row][0];  
for (int c = 0; c < len - 1; c++) {  
    lanes[row][c] = lanes[row][c + 1];  
}  
lanes[row][len - 1] = first;  
for (int v : lanes[row]) {  
    System.out.print(v + " ");  
}
```

OUTPUT

8 1 6 4 3

Reverse a column: swap the first half

REVERSE COLUMN 1

```
int[][] stack = { {1, 9},
                  {2, 8},
                  {3, 7},
                  {4, 6} };

int col = 1;
int n = stack.length;
for (int r = 0; r < n / 2; r++) {
    int temp = stack[r][col];
    stack[r][col] = stack[n - 1 - r][col];
    stack[n - 1 - r][col] = temp;
}
for (int[] row : stack) {
    System.out.println(row[0] + " " + row[1]);
}
```

OUTPUT

```
1 6
2 7
3 8
4 9
```



Stop at **$n / 2$** . Going all the way swaps every pair **twice**, and two swaps cancel out.

Words to know

Accumulator

Collects a result, like a running total, across loop passes.

Flag

A boolean set before the loops and switched when one element decides.

Counter

An int that starts at 0 and goes up by 1 for each match.

Consecutive

Side by side: c and $c + 1$ in a row, or r and $r + 1$ in a column.

Shift / rotate

Move every element one spot; shift loses the end, rotate wraps it.

Swap

Trade two elements, saving one in a temporary variable first.

Which elements, then what?



A 2D algorithm makes two choices. Explain both to your partner, using the theater from the warm-up.

SENTENCE STARTER

First I pick ____ to visit, so my loops _____. Then I pick ____ to do with each one.

Next: the Work the Grid puzzle worksheet.