

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.1 2D Array Basics

Solutions

8.1.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
int[][] grid = new int[2][3];
grid[1][0] = 7;
System.out.println(grid[0][1] + grid[1][0]);
```

Output

7

All elements begin at 0; only row 1, column 0 is assigned 7.

8.1.2 - Write code

Create a 3 by 3 int array with 1 on its main diagonal and 0 elsewhere. Print each row with Arrays.toString.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
int[][] identity = new int[3][3];
for (int i = 0; i < identity.length; i++) identity[i][i] = 1;
for (int[] row : identity) System.out.println(Arrays.toString(row));
```

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.2 2D Array Traversal

Solutions

8.2.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
int[][] grid = {{1,2},{3,4}};
for (int c = 0; c < grid[0].length; c++) {
    for (int r = 0; r < grid.length; r++) System.out.print(grid[r][c]);
}
System.out.println();
```

Output

1324

The outer column loop makes this column-major order.

8.2.2 - Write code

Write public static int rowSum(int[][] grid, int row), assuming row is valid. Sum that row without changing the grid.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static int rowSum(int[][] grid, int row) {
    int sum = 0;
    for (int c = 0; c < grid[row].length; c++) sum += grid[row][c];
    return sum;
}
```

Check your code

```
int[][] grid = {{2,3,4},{7,8,9}};
System.out.println(rowSum(grid,0));
System.out.println(rowSum(grid,1));
```

Expected output

9
24

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.3 Linear and Binary Search

Solutions

8.3.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
int[] sorted = {2,5,8,11,14};
int low = 0;
int high = 4;
int mid = (low + high) / 2;
if (sorted[mid] < 11) low = mid + 1;
System.out.println(low + ":" + high);
```

Output

3:4

Because the midpoint value is below the target, discard it and every smaller index.

8.3.2 - Write code

Write public static int binarySearch(int[] values, int target) for an ascending sorted array. Return a matching index or -1. Use an iterative loop.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static int binarySearch(int[] values, int target) {
    int low = 0;
    int high = values.length - 1;
    while (low <= high) {
        int mid = low + (high - low) / 2;
        if (values[mid] == target) return mid;
        if (values[mid] < target) low = mid + 1;
        else high = mid - 1;
    }
    return -1;
}
```

Check your code

```
System.out.println(binarySearch(new int[]{2,5,8,11,14},14));
System.out.println(binarySearch(new int[]{2,5,8},4));
System.out.println(binarySearch(new int[] {},4));
```

Expected output

4
-1
-1

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.4 Sorting and Efficiency

Solutions

8.4.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
int[] a = {2,5,4};
int key = a[2];
int j = 1;
while (j >= 0 && a[j] > key) { a[j + 1] = a[j]; j--; }
a[j + 1] = key;
System.out.println(Arrays.toString(a));
```

Output

[2, 4, 5]

The insertion shifts 5 right, then places 4 after 2.

8.4.2 - Write code

Write public static void selectionSort(int[] values). Sort ascending in place. Explain why doubling a large input size produces approximately four times as many comparisons.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static void selectionSort(int[] values) {
    for (int start = 0; start < values.length - 1; start++) {
        int min = start;
        for (int i = start + 1; i < values.length; i++) {
            if (values[i] < values[min]) min = i;
        }
        int temp = values[start];
        values[start] = values[min];
        values[min] = temp;
    }
}
// Comparisons:  $n * (n - 1) / 2$ ; the leading term is proportional to  $n$  squared.
```

Check your code

```
int[] a = {5,2,5,-1}; selectionSort(a); System.out.println(Arrays.toString(a));
int[] b = {}; selectionSort(b); System.out.println(Arrays.toString(b));
```

Expected output

[-1, 2, 5, 5]
[]

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.5 Text Files and Scanner

Solutions

8.5.1 - Trace

Determine the exact output. Explain the Java rule or state changes that produce it.

```
Scanner input = new Scanner("Ada 17");
String name = input.next();
int age = input.nextInt();
System.out.println(name + ":" + (age + 1));
input.close();
```

Output

Ada:18

The first token is text, and the second is parsed as an integer before addition.

8.5.2 - Write code

Write public static int sumTokens(Scanner input). Assume every remaining token is an integer. Consume them all and return the sum. The caller owns and closes the scanner. For a text file, the same method can receive a Scanner connected to that file.

One valid solution is shown below. Other implementations satisfying the contract are also valid.

```
public static int sumTokens(Scanner input) {
    int sum = 0;
    while (input.hasNext()) sum += input.nextInt();
    return sum;
}
```

Check your code

```
Scanner a = new Scanner("5 -2 8"); System.out.println(sumTokens(a)); a.close();
Scanner b = new Scanner(""); System.out.println(sumTokens(b)); b.close();
```

Expected output

```
11
0
```

Teaching Unit 8: 2D Arrays, Algorithms, and Data

8.6 Data and Its Implications

Solutions

8.6.1 - Explain

A school surveys only robotics-club members to decide which programming class all students want. Identify a sampling problem and propose a better collection method.

Robotics-club members are likely to have more programming interest and experience than the full student population. Survey a representative sample across grades and experience levels, or invite the full student body and evaluate nonresponse.

8.6.2 - Design

A practice app stores student names, birth dates, full location histories, and topic scores forever. It needs only to recommend review topics. Propose three concrete changes and explain their benefits.

Keep only information needed for recommendations, such as a student identifier and recent topic scores. Remove birth dates and location histories unless a separate necessary purpose justifies them. Restrict access and use a short retention/deletion policy. These changes reduce exposure while preserving the recommendation function.