

# KIT101 Programming Fundamentals

## PP Task 5.1 Tracing Code with Arrays

---

### Overview

---




**Purpose:** Learn about how arrays work in the computer.

**Task:** Trace the execution of the provided code snippets to demonstrate key aspects of working with arrays.

**Learning Outcomes:** 1  2 

**Time:** Aim to complete this task before the start of Week 6.

**Resources:**

- Introductory Programming Notes:
  - 10 Managing Collections with Arrays
- Online Mini-lectures:
  - Arrays:
    -  Introduction and key syntax
    -  Tracing array code
    -  Writing methods to work with arrays

### Submission Details

Upload the following to the MyLO submission folder for this task:

- **Tracing tables** for the methods and code samples you execute. These can be *either* photos/scans of hand-written tables or a Word/PDF document. You may find hand-written is easier.

### Assessment Criteria

A  Completed submission will:

- Show how the code is executed by tracing the changes to variable values and recording the results for different inputs
- Provide meaningful names (in the images/document) for each of the two mystery methods traced

### Instructions

---

Arrays are incredibly useful, and really important to understand. Tracing code that involves their use can really help you understand these more fully.

Part 10 of the *Introductory Programming Notes* suggests two alternative ways of tracing code involving arrays, depending on whether the code is predominantly *reading* values from the array or *modifying* the array's contents.

In the first case draw a tracing table as usual, but with the array depicted off to the side, such as:

	0	1	2	3	4
data	2	6	-4	8	3

In the second case you can incorporate the array's elements into the table, with one column per element, so the columns may resemble:

		data				
Line	other variables	0	1	2	3	4

Only the last sample below modifies an array after initialising it, but use whichever approach you find best.

**Tip:** Feel free to put additional annotations on your table to document what is happening in the code. These could include noting the outcome of a loop's test or indicating if a return statement has been reached. The only requirement is that we can see you have followed the code's execution.

Your task is to trace the execution of two methods that work with arrays (and to suggest a meaningful name for each) and also to trace two other samples of array-related code.

## Mystery Method 1

This method performs a useful task with a collection of integers, but we've hidden its name so you have to work out what that is. What does it do?

```
1 public static int ???(int[] data) { //Note: ??? isn't a valid name
2     int result = -1000;
3
4     for (int i = 0; i < data.length; i++) {
```

```

5         if (data[i] > result) {
6             result = data[i];
7         }
8     }
9
10    return result;
11 }

```

**Trace the execution** of the method if it were called with the following parameter values (that is, create two tracing tables, one for each run) to see if you can work out what it does. Include the following summary table after your traces to record the outcomes.

data	Value returned
{ 1, 4, -4, 3, 8 }	
{ 15, 20, 50, 35 }	

**Provide a name** for this method with your two tracing tables.

**Note:** There is no array literal syntax in Java *except* when initialising an array, as in:

```
int[] a = { 1, 2, 3 };
```

If you want to implement the above function in a program and then call it with a predefined array value then instead of declaring and initialising an array variable then prefix the list of values with `new int[]`, as in `new int[]{1, 2, 3, 4, 5}`. This allows the compiler to confirm that {1, 2, 3, 4, 5} represents an array of `ints`. And if you do choose to implement it to confirm your code traces then remember to give it a valid name (??? is not a valid Java identifier).

## Mystery Method 2

The method below performs another useful task with an array of `int` values. Demonstrate how it is executed in the computer by creating two tracing tables, one for each sample input, then suggest a valid method name that describes its purpose.

```

1    public static boolean ???(int[] data, int val) {
2        boolean result = false;
3        int i = 0;

```

```

4
5     while (i < data.length && !result) {
6         if (data[i] == val) {
7             result = true;
8         }
9         i++;
10    }
11
12    return result;
13 }

```

**Trace the execution** of the method if it were called with the following parameter values (that is, create two tracing tables, one for each run) to see if you can work out what it does. Include the following summary table after your traces to record the outcomes.

data	val	Value returned
{ 2, 6, -3, 7, 3 }	7	
{ -1, 7, 2, -4, 9 }	5	

**Provide a name** for this method with your two tracing tables.

### Code Sample 3

Create a tracing table for the following code:

```

1    String[] spacecraft = { "Nostromo", "Sulaco", "Prometheus" };
2
3    System.out.println("Here are " + spacecraft.length + " of many fictional spacecraft:");
4    for (int i = 0; i < spacecraft.length; i++) {
5        System.out.println(spacecraft[i]);
6    }

```

### Code Sample 4

Create a tracing table for the following code. It may look confusing as a whole, so work through it methodically line by line, paying careful attention to the loop initialisation:

```
1  int[] values = { 5, 6, 7 };
2  int temp;
3
4  temp = values[0];
5  for (int i = 1; i < values.length; i++) {
6      values[i - 1] = values[i];
7  }
8  values[ values.length - 1 ] = temp;
```

At the end, sketch the array's contents [as illustrated above](#).

---