

Array Parameters and Array Returns

A byte size lesson in Java programming.

Recall #1 Array

- From previous lessons, we know that an **array** in Java is a group of memory locations that can store a number of variables of the **same data type**.
- Below is an array that can store 7 temperatures of type double.

```
double[] temperatures = new double[7];
```

Recall #2 Parameter and Return

- When we learnt about Java **methods** in previous lessons, we have come across parameters and the return keyword.
- Below is a method called **calcAreaSquare** that accepts a **parameter** “w” and **returns** a value of type double.

```
public static double calcAreaSquare(double w) {  
    double area = 0.0;  
    area = w*w;  
    return area;  
}
```


Mix and Match

- An array that stores days of the week
- A method without parameters but returns a String.
- A method that accepts a parameter.
- A method that accepts a parameter and returns a value.

(3)

```
1. int factorial(int n) { //... }  
2. void square(double w) { //... }  
3. String[] daysOfWeek = new double[7];  
4. String getMessage() { //... }
```

Passing arrays as parameters

- A method in Java is able to accept any kind of parameter.
- Even something as complex as an array.
- Below is an example of a method that **accepts an array of integers**.

```
void add(int[] numbers) {  
    // code to add all numbers in an array  
}
```

How to use array parameters

- We pass in information in an array **so that the method can use it**. In the example below the method is looping through the numbers array. Do you notice something different here?

```
int add(int[] numbers) {  
    int addition = 0;  
    for (int i = 0; i < numbers.length; i++) {  
        addition = addition + numbers[i];  
    }  
    return addition;  
}
```


Array.length

- When arrays are passed as parameters it is important for the method to “know” how large the array is to avoid mistakes.
- For this reason, Java exposes a **property, length**, that we can use from the array.
- You will understand what properties are when we do objects very soon.

```
int[] numbers = new int[10];  
int numbersLength = numbers.length; // should give us 10
```

An easy warm-up first...

- For the following array, what will .length give back as a value?

```
double[] temperatures = new double[7];  
int temperaturesLength = temperatures.length; // should give 
```

```
String[] names = new String[100];  
int namesLength = names.length; // should give 
```


Let's test your understanding!

- Fill in the blanks:

```
double average(double[] numbers) {  
    int average = 0;  
    for (int i = 0; i <  ; i++) {  
        average = numbers + numbers[i];  
    }  
    return  /  ;  
}
```

Calling methods that accept array parameters

```
int add(int[] numbers) {  
    int addition = 0;  
    for (int i = 0; i < numbers.length; i++) {  
        addition = addition + numbers[i];  
    }  
    return addition;  
}
```

Here's how we call the above method:

```
int[] myNumbers = {1, 2, 3, 4, 5};  
Int myAddition = add(myNumbers);
```

Let's test your understanding!

Complete the following method definition:

```
int sub(int[] numbers) {  
    int subtraction = 0;  
    for (int i = 0; i < numbers.length; i++) {  
        subtraction = subtraction - numbers[i];  
    }  
    return subtraction;  
}
```

And this call to subtract all numbers in an array called **myNumbers**:

```
int mySubtraction =  ;
```


Methods that return an array as a value

One final note before our practice session. Here is a method that returns an array!

```
int[] sort(int[] numbers){  
    // code to sort numbers  
    return numbers;  
}
```