

Arrays as parameters

Swapping values

```
public static void main(String[] args) {  
    int a = 7;  
    int b = 35;  
  
    // swap a with b?  
    a = b;  
    b = a;  
  
    System.out.println(a + " " + b);  
}
```

– What is wrong with this code? What is its output?

- The red code should be replaced with:

```
int temp = a;  
a = b;  
b = temp;
```

Array reversal question

- Write code that reverses the elements of an array.
 - For example, if the array initially stores:
`[11, 42, -5, 27, 0, 89]`
 - Then after your reversal code, it should store:
`[89, 0, 27, -5, 42, 11]`
 - The code should work for an array of any size.
 - Hint: think about swapping various elements...

Algorithm idea

- Swap pairs of elements from the edges; work inwards:

<i>index</i>	0	1	2	3	4	5
<i>value</i>	89	0	27	-5	42	11
	↑	↑	↑	↑	↑	↑

Flawed algorithm

- What's wrong with this code?

```
int[] numbers = [11, 42, -5, 27, 0, 89];  
// reverse the array  
for (int i = 0; i < numbers.length; i++) {  
    int temp = numbers[i];  
    numbers[i] = numbers[numbers.length - 1 - i];  
    numbers[numbers.length - 1 - i] = temp;  
}
```

- The loop goes too far and un-reverses the array! Fixed version:

```
for (int i = 0; i < numbers.length / 2; i++) {  
    int temp = numbers[i];  
    numbers[i] = numbers[numbers.length - 1 - i];  
    numbers[numbers.length - 1 - i] = temp;  
}
```

Array reverse question 2

- Turn your array reversal code into a `reverse` method.
 - Accept the array of integers to reverse as a parameter.

```
int[] numbers = {11, 42, -5, 27, 0, 89};  
reverse(numbers);
```

- How do we write methods that accept arrays as parameters?
- Will we need to return the new array contents after reversal?
- ...

Array parameter (declare)

```
public static type methodName(type[] name) {
```

- Example:

```
// Returns the average of the given array of numbers.
```

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

- You don't specify the array's length (but you can examine it).

Array parameter (call)

methodName (**arrayName**) ;

- Example:

```
public class MyProgram {  
    public static void main(String[] args) {  
        // figure out the average TA IQ  
        int[] iq = {126, 84, 149, 167, 95};  
        double avg = average(iq);  
        System.out.println("Average IQ = " + avg);  
    }  
    ...  
}
```

- Notice that you don't write the [] when passing the array.

Array return (declare)

```
public static type[] methodName(parameters) {
```

- Example:

```
// Returns a new array with two copies of each value.
```

```
// Example: [1, 4, 0, 7] -> [1, 1, 4, 4, 0, 0, 7, 7]
```

```
public static int[] stutter(int[] numbers) {  
    int[] result = new int[2 * numbers.length];  
    for (int i = 0; i < numbers.length; i++) {  
        result[2 * i] = numbers[i];  
        result[2 * i + 1] = numbers[i];  
    }  
    return result;  
}
```

Array return (call)

type[] name = methodName(parameters) ;

- Example:

```
public class MyProgram {  
    public static void main(String[] args) {  
        int[] iq = {126, 84, 149, 167, 95};  
        int[] stuttered = stutter(iq);  
        System.out.println(Arrays.toString(stuttered));  
    }  
    ...  
}
```

- Output:

[126, 126, 84, 84, 149, 149, 167, 167, 95, 95]