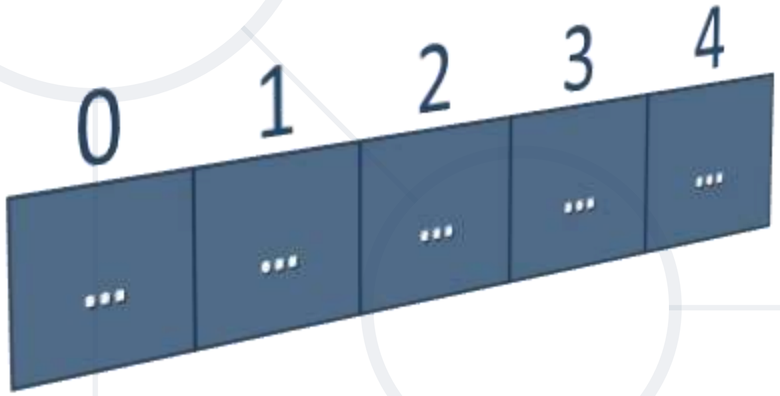
Java Foundations

Arrays in Java

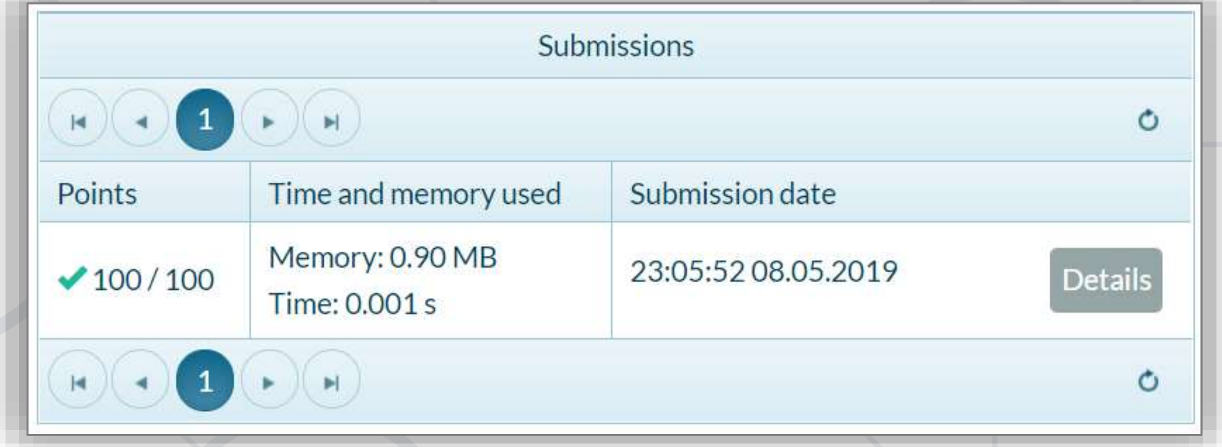


Your Course Instructors

George Georgiev

Svetlin Nakov

The Judge System



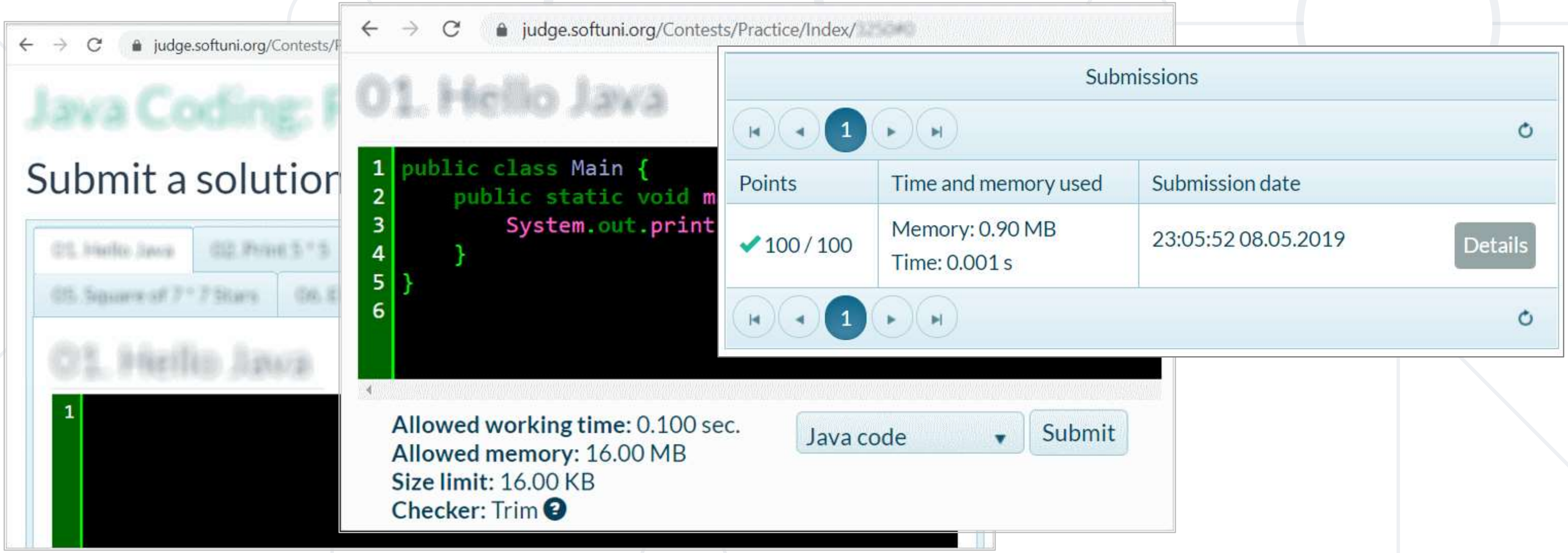
Submissions

Points	Time and memory used	Submission date	
✓ 100 / 100	Memory: 0.90 MB Time: 0.001 s	23:05:52 08.05.2019	Details

Sending your Solutions for Automated Evaluation

Testing Your Code in the Judge System

- Test your code online in the SoftUni **Judge system**: https://judge.softuni.org/Contests/3294



judge.softuni.org/Contests/Practice/Index/

Submit a solution

```java
public class Main {
    public static void m
        System.out.print
    }
}
```

Allowed working time: 0.100 sec.
Allowed memory: 16.00 MB
Size limit: 16.00 KB
Checker: Trim

Submissions

Points	Time and memory used	Submission date	
✓ 100 / 100	Memory: 0.90 MB Time: 0.001 s	23:05:52 08.05.2019	Details

Arrays

Fixed-Size Sequences of Numbered Elements

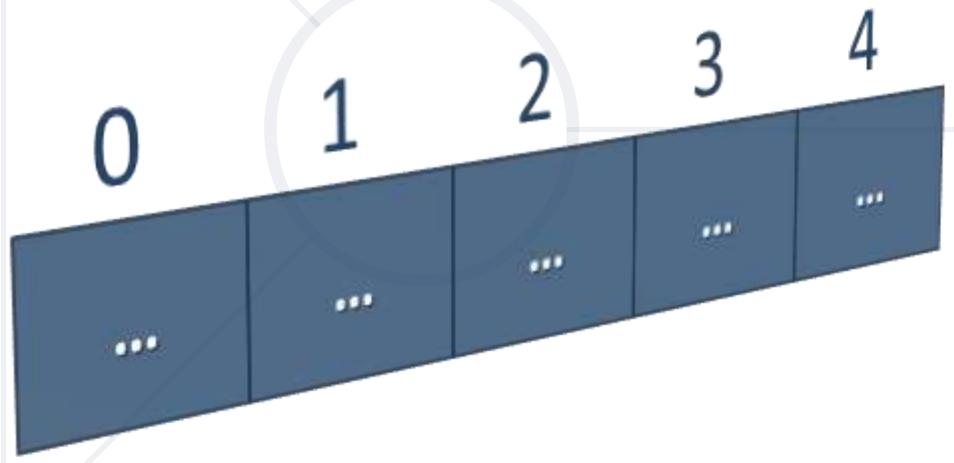
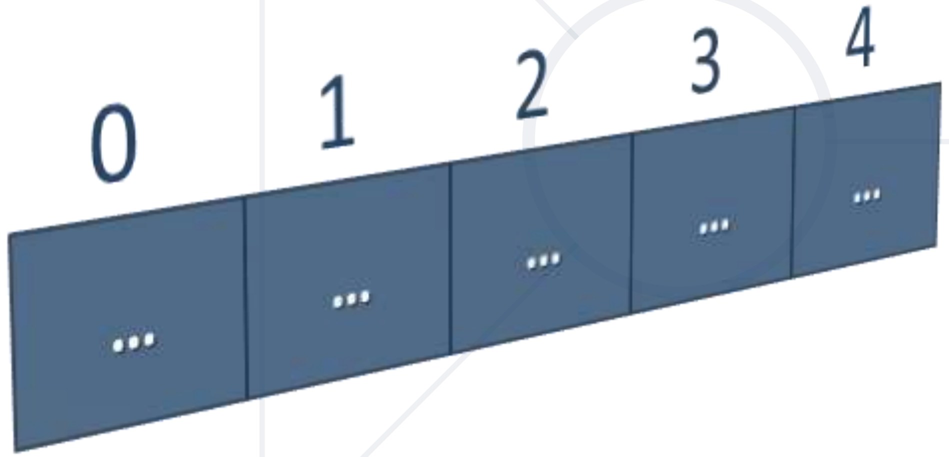
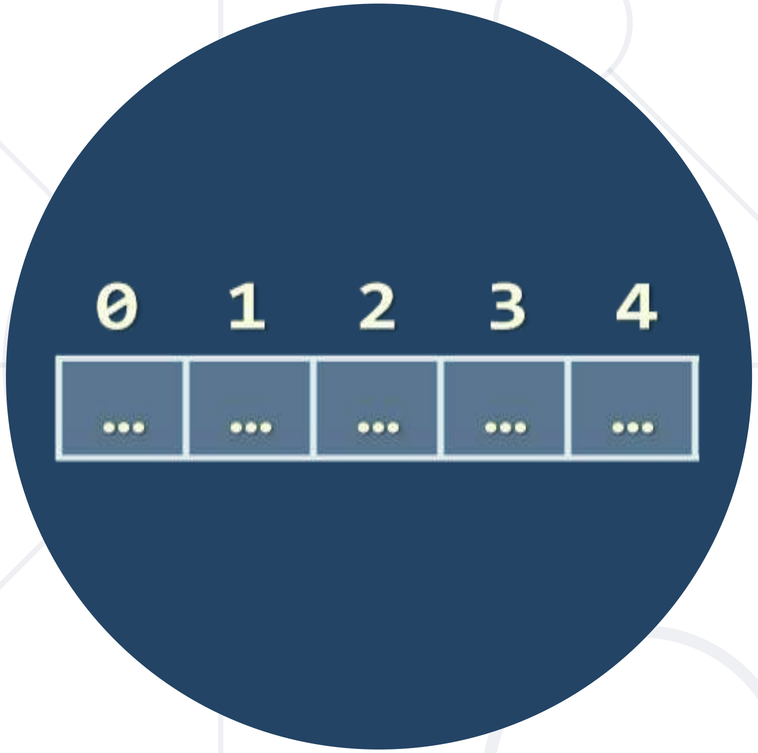


Table of Contents

1. Arrays
2. Array Operations
3. Reading Arrays from the Console
4. For-each Loop



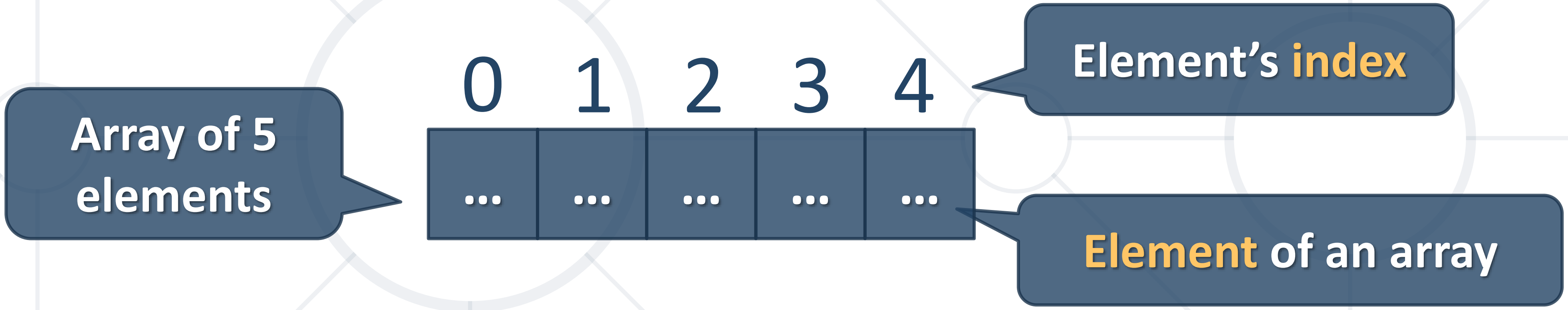


Arrays in Java

Working with Arrays of Elements

What are Arrays?

- In programming, an **array** is a **sequence of elements**



- Arrays have **fixed size** (**array.length**) cannot be resized
- Elements are of the **same type** (e.g. integers)
- Elements are numbered from **0** to **length-1**

Working with Arrays

SoftUni

- **Allocating** an array of 10 integers:

```java
int[] numbers = new int[10];
```

All elements are initially == 0

- **Assigning values** to the array elements:

```java
for (int i = 0; i < numbers.length; i++)
  numbers[i] = 1;
```

The length holds the number of array elements

- **Accessing** array elements by index:

```java
numbers[5] = numbers[2] + numbers[7];
numbers[10] = 1; // ArrayIndexOutOfBoundsException
```

The [] operator accesses elements by index

Days of Week – Example

- The days of a week can be stored in an array of strings:

```
String[] days = {
   "Monday",
   "Tuesday",
   "Wednesday",
   "Thursday",
   "Friday",
   "Saturday",
   "Sunday"
};
```

Operator	Value
days[0]	Monday
days[1]	Tuesday
days[2]	Wednesday
days[3]	Thursday
days[4]	Friday
days[5]	Saturday
days[6]	Sunday

Problem: Day of Week

- Enter a **day number** [1…7] and print the **day name** (in English) or "Invalid day!"

```
String[] days = { "Monday", "Tuesday", "Wednesday",
"Thursday", "Friday", "Saturday", "Sunday" };
int day = Integer.parseInt(sc.nextLine());
if (day >= 1 && day <= 7)
  System.out.println(days[day - 1]);
else
  System.out.println("Invalid day!");
```

The first day in our array is on index 0, not 1.

Reading Array

Using a `for` Loop or `String.split()`

Reading Arrays From the Console

- First, read the array **length** from the console :

```java
int n = Integer.parseInt(sc.nextLine());
```

- Next, create an array of given size **n** and read its **elements**:

```java
int[] arr = new int[n];
for (int i = 0; i < n; i++) {
  arr[i] = Integer.parseInt(sc.nextLine());
}
```

Reading Array Values from a Single Line

- Arrays can be read from a **single line** of **separated values**

```
2 8 30 25 40 72 -2 44 56
```

```
String values = sc.nextLine();
String[] items = values.split(" ");
int[] arr = new int[items.length];

for (int i = 0; i < items.length; i++)
  arr[i] = Integer.parseInt(items[i]);
```

Shorter: Reading Array from a Single Line

- Read an array of integers using functional programming:

```
String inputLine = sc.nextLine();
String[] items = inputLine.split(" ");
int[] arr = Arrays.stream(items)
  .mapToInt(e -> Integer.parseInt(e)).toArray();
```

import java.util.Arrays;

```
int[] arr = Arrays
  .stream(sc.nextLine().split(" "))
  .mapToInt(e -> Integer.parseInt(e)).toArray();
```

You can chain methods

Printing Arrays on the Console

- To print all array elements, a for-loop can be used
  - Separate elements with white space or a new line

```
String[] arr = {"one", "two"};
// == new String [] {"one", "two"};
// Process all array elements
for (int i = 0; i < arr.length; i++) {
   System.out.printf("arr[%d] = %s%n", i, arr[i]);
}
```

Problem: Reverse an Array of Integers

- Read an array of integers (**n** lines of integers), **reverse** it and print its elements on a single line, space-separated:

```
3
10
20
30
```

$\rightarrow$

```
30 20 10
```

```
4
-1
20
99
5
```

$\rightarrow$

```
5 99 20 -1
```

Solution: Reverse an Array of Integers

```
// Read the array (n lines of integers)
int n = Integer.parseInt(sc.nextLine());
int[] arr = new int[n];
for (int i = 0; i < n; i++)
  arr[i] = Integer.parseInt(sc.nextLine());
// Print the elements from the last to the first
for (int i = n - 1; i >= 0; i--)
  System.out.print(arr[i] + " ");
System.out.println();
```

Printing Arrays with for / String.join(…)

- Use for-loop:

```
String[] arr = {"one", "two"};
for (int i = 0; i < arr.length; i++)
   System.out.println(arr[i]);
```

- Use **String.join(separator, array)**:

```
String[] strings = { "one", "two" };
System.out.println(String.join(" ", strings)); // one two
int[] arr = { 1, 2, 3 };
System.out.println(String.join(" ", arr)); // Compile error
```

Works only with strings

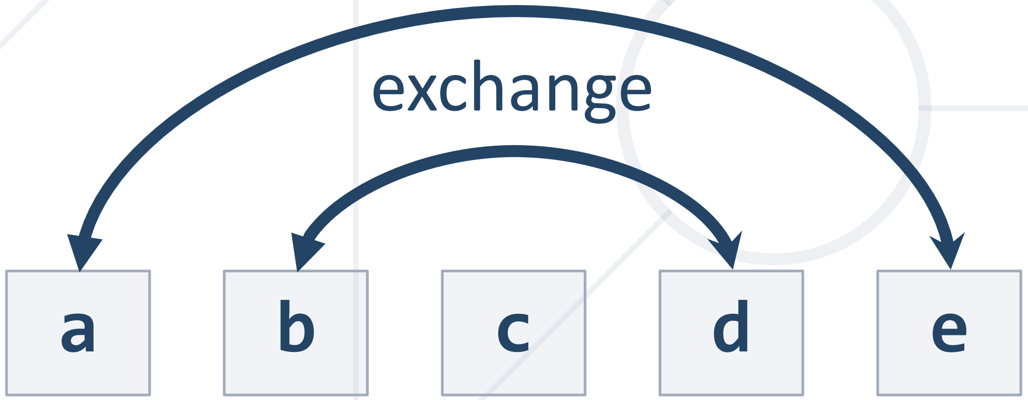
Problem: Reverse Array of Strings

- Read an array of strings (space separated values), reverse it and print its elements:

```
a b c d e  ➡  e d c b a
```

```
-1 hi ho w  ➡  w ho hi -1
```

- Reversing array elements:



Solution: Reverse Array of Strings

```java
String[] elements = sc.nextLine().split(" ");
for (int i = 0; i < elements.length / 2; i++) {
  String oldElement = elements[i];
  elements[i] = elements[elements.length - 1 - i];
  elements[elements.length - 1 - i] = oldElement;
}
System.out.println(String.join(" ", elements));
```

For-each Loop

Iterate through Collections

For-each Loop

- Iterates through all elements in a collection
- Cannot access the current index
- **Read-only**

```
for (var item : collection) {
    // Process the value here
}
```

Print an Array with Foreach

```
int[] numbers = { 1, 2, 3, 4, 5 };
for (int number : numbers) {
   System.out.println(number + " ");
}
```

```
1 2 3 4 5
```

Problem: Even and Odd Subtraction

- Read an array of integers
- Sum all even and odd numbers
- Find the difference
- Examples:

`1 2 3 4 5 6`	`3`
`2 4 6 8 10`	`30`
`3 5 7 9 11`	`-35`
`2 2 2 2 2 2`	`12`

Solution: Even and Odd Subtraction

```java
int[] arr = Arrays.stream(sc.nextLine().split(" "))
  .mapToInt(e -> Integer.parseInt(e)).toArray();
int evenSum = 0;
int oddSum = 0;
for (int num : arr) {
  if (num % 2 == 0) evenSum += num;
  else oddSum += num;
}
// TODO: Find the difference and print it
```

Live Exercises

Summary

- Arrays hold a **sequence** of elements
  - Elements are numbered from **`0`** to **`length – 1`**
- Creating (allocating) an array
- Accessing array elements by **index**
- Printing array elements

Next Steps

- Join the SoftUni "Learn To Code" Community

https://softuni.org

- Access the Free Coding Lessons
- Get Help from the Mentors
- Meet the Other Learners