java from scratch Knowlage Base

- Welcome
- Mastering Agile and Scrum: Video Training Series
- Program
- Introduction
- The architecture of an operating system
- The structure of files and directories
- Navigating through directories
- Environment variables
- Extracting archives
- Installing the software
- Monitoring the usage of system resources
- Ending – control questions
- Software Installations
- IntelliJ EduTools – installation
- Introduction
- A brief history of Java
- First program
- Types of data
- Operators
- Conditional statements
- Loops

Arrays

An array is a data structure that allows you to store multiple elements of the same type. Instead of creating several, a dozen or several dozen variables that are somehow related to each other, you can group them to be stored in one array. You can access each element of an array through its *index* (position in the array), which is a number of the `int` type.

When creating an array, you must specify its size. The size of an array **cannot be modified**. If it turns out that you need a larger array, you must create a new (larger) one and copy the data from the smaller array to it.

Indexes of an array are numbered from `0`. Thus, the first element is the element with *index* `0` , and the last one is the element with index `n-1` , where `n` is the size of the array.

Elements of the array can be both primitive and object-oriented types. An array (of whatever type) **is an object**! Arrays can be single- or multidimensional.

Creating arrays

Pattern of array declaration

```
type[] variable;
```

Example of array declaration

```
int[] numbers;
boolean[] switches;
String[] names;
Double[] values;
```

Tip

Sometimes you can encounter alternative

- Arrays
- Object-oriented programming
- Conclusion
- Assignments
- Basics of GIT – video training
- HTTP basics – video training
- Design patterns and good practices video course
- Prework Primer: Essential Concepts in Programming
- Cybersecurity Essentials: Must-Watch Training Materials
- Java Developer – introduction
- Java Fundamentals – coursebook
- Java fundamentals slides
- Java fundamentals tasks
- Test 1st attempt | after the block: Java fundamentals
- Test 2nd attempt | after the block: Java fundamentals
- GIT version control system coursebook
- Java – Fundamentals: Coding slides
- Java fundamentals tasks
- Software Testing slides
- Software Testing Coursebook
- Software Testing tasks
- Test 1st attempt | after the block: Software testing
- Test 2nd attempt | after the block: Software testing
- Java – Advanced Features coursebook

declarations of array variables. Equivalent declarations of array variables.

Equivalent declarations of array variables

```
int[] a;
int []b;
int c[];
```

The recommended record is `int[] a`.

Pattern of array declaration and initialization

```
type[] variable = new type[N];
type[] variable = new type[]{value1, value2, ...,
valueN};
type[] variable = {value1, value2, ..., valueN};
```

- creating an N-element array without filling it with data; each element now has a default value relevant for the array type
- creating an array and filling it in with data; based on their quantity, Java knows the size of the array
- shortened (recommended) entry of the above declaration

Examples of array declaration and initialization

```
int[] numbers = new int[10];
String[] labels = new String[2];
String[] names = new String[]{"Sandra", "Tom", "Kate",
"WoBarttek"};
Double[] values = {3.1415D, 2.7182D};
```

- we have created a `10` – element array of the `int` type; we have not filled it with values, therefore it has been filled with default values for the `int` type, that is `0`
- we have created a `2` – element array of the `String` type; we have not filled it with values, therefore it has been filled with default values for the `String` type, that is, `null`
- we have created a `4` – element array of the `String` type and provided 4 elements

- Java – Advanced Features slides
- Java – Advanced Features tasks
- Test 1st attempt | after the block: Java Advanced Features
- Test 2nd attempt | after the block: Java Advanced Features
- Java – Advanced Features: Coding slides
- Java – Advanced Features: Coding tasks
- Test 1st attempt | after the block: Java Advanced Features coding
- Test 2nd attempt | after the block: Java Advanced Features coding
- Data bases SQL coursebook
- Databases SQL slides
- Databases – SQL tasks
- Coursebook: JDBC i Hiberate
- Excercises: JDBC & Hibernate
- Test 1st attempt | after the block: JDBC
- Test 2nd attempt | after the block: JDBC
- Design patterns and good practices
- Design patterns and good practices slides
- Design Patterns & Good Practices tasks
- Practical project coursebook
- Practical project slides
- HTML, CSS, JAVASCIRPT Coursebook
- HTML, CSS, JAVASCRIPT slides
- HTML, CSS, JavaScript tasks

- we have created a 2 – element array of the `Double` type and provided 2 values

- If at the time of creating the array you **know how many elements you want to store** (you know the array size, but you **do not know the values** yet), then use the following pattern:`type[] variable = new type[N];` np.`int[] numbersFromLottery = new int[7];`
- However, if at the time of creating the array you **know all the values** to be stored (their number is the size of the array), then use the following record:`type[] variable = {value1, value2, ..., valueN};` e.g.`String[] seasons = {"Spring", "Summer", "Fall", "Winter"};`

Operations on arrays

Let us try to first display the content of the created (declared and initialized) arrays.

Example of displaying the array contents

```
int[] numbers = new int[5];
int[] luckyNumbers = {1, 3, 7, 17, 21};
String[] names = new String[10];
String[] seasons = {"Spring", "Summer", "Fall",
"Winter"};

System.out.println(Arrays.toString(numbers));
System.out.println(Arrays.toString(luckyNumbers));
System.out.println(Arrays.toString(names));
System.out.println(Arrays.toString(seasons));

System.out.println(luckyNumbers);
System.out.println(seasons);
```

Result:

```
[0, 0, 0, 0, 0]
[1, 3, 7, 17, 21]
[null, null, null, null, null, null, null, null, null,
null]
[Spring, Summer, Fall, Winter]
[I@610455d6
[Ljava.lang.String;@511d50c0
```

Tip

- Test 1st attempt | after the block: HTML,CSS,JS
- Test 2nd attempt | after the block: HTML,CSS,JS
- Frontend Technologies coursebook
- Frontend technologies slides
- Frontend Technologies tasks
- Test 1st attempt | after the block: FRONTEND TECHNOLOGIES (ANGULAR)
- Test 2nd attempt | after Frontend technologies
- Spring coursebook
- Spring slides
- Spring tasks
- Test 1st attempt | after the block: spring
- Test 2nd attempt | after the block: spring
- Mockito
- PowerMock
- Testing exceptions
- Parametrized tests
- Final project coursebook
- Final project slides
- Class assignments

To display the contents of the entire array, we have used the `Arrays.toString()` method. If this method is underlined in red on your screens, add the following line above the class declaration:

```
import java.util.Arrays;
```

> It allows you to apply methods from the `Arrays` class in our code.
>
> If you try to display the arrays with the `System.out.println()` method, you will get an poorly legible result.

You can also download and modify individual elements. Access to an element is obtained through its *index*.

Pattern of access to the element of the array

```
variable[index]
```

Example of operations on individual elements of arrays

```
int[] numbers = new int[5];
String[] seasons = {"Spring", "Summer", "Fall",
"Winter"};

System.out.println(Arrays.toString(numbers));
System.out.println(Arrays.toString(seasons));

numbers[0] = 25;
System.out.println(Arrays.toString(numbers));
System.out.println(numbers[0]);
numbers[4] = -200;
System.out.println(Arrays.toString(numbers));

seasons[1] = "";
seasons[2] = null;
System.out.println(Arrays.toString(seasons));
System.out.println(seasons[3]);
```

Result:

```
[0, 0, 0, 0, 0]
[Spring, Summer, Fall, Winter]
[25, 0, 0, 0, 0]
25
[25, 0, 0, 0, -200]
[Spring, , null, Winter]
Winter
```

It is useful to know the size of the array (if, for example, you do not remember its size or you were only given an array variable). The size of the array is obtained by referring to a field called `length`.

Example of access to the array size

```
int[] numbers = new int[5];
String[] seasons = {"Spring", "Summer", "Fall",
"Winter"};

System.out.println(numbers.length);
System.out.println(seasons.length);
```

Result:

```
5
4
```

> **Tip**
>
> Thus, the first element of the array is the element with *index* `0`, and the last element is the one with *index* `variable`. **length** – 1

> **Important**
>
> If you try to refer to an *index* outside the range (`0`, `variable.length-1`), the program will end its operation and an error will appear.

Examples of incorrect references

Examples of incorrect references

```
String[] seasons = {"Spring",
"Summer", "Fall", "Winter"};
System.out.println(seasons[-1]);
System.out.println(seasons[5]);
```

Arrays and loops

Loops are very common when working with arrays.

An example of using loops with arrays

```
String[] seasons = {"Spring", "Summer", "Fall",
"Winter"};
int[] numbers = {34, 23, 89, 14, 63, 2, 55};

for (int i = 0; i < seasons.length; i++) {
    System.out.println(seasons[i]);
}

for (int i = numbers.length - 1; i >= 0; i--) {
    System.out.print(numbers[i] + " ");
}

System.out.println();

for (String season : seasons) {
    System.out.println(season);
}
```

Result:

```
Spring
Summer
Fall
Winter
55 2 63 14 89 23 34
Spring
Summer
Fall
Winter
```

Summary

What have you learned in this chapter?

What are arrays?

Arrays are data structures that allow you to store several elements of the same type. You can access individual elements of the array through the *index*.

What is an index in an array?

Index is the position of an element in an array. Indexes are numbered from `0`. This means that the first element of the array is the element with index 0. The last element of the n-element array is the element with index `n-1`.

Is it possible to change the size of the array?

No, an array once declared has a fixed size that cannot be changed. If you need a larger (or smaller) array, you must create a new array and move elements to it.

How should I display the array contents?

To display the contents of the array, you can use the `Arrays.toString(<arraya>)` function, e.g.

```
String[] seasons = {"Spring", "Summer", "Fall",
"Winter"};
System.out.println(Arrays.toString(seasons));
```

> **Important**
>
> Remember about the import declaration:
>
> ```
> import java.util.Arrays;
> ```

How should I get information about the size of the array?

Every array contains a field called `length` that defines its size.

Exercises

- There is an array of 10 elements of the `int` type:`int[] integers = {1, 3, 5, 2, 5, 6, 7, 4, 9, 7};` Using one of the loops, write a code fragment that displays:`- all numbers - first 6 numbers - last 6 numbers - all even`

`numbers – all digits at odd indexes – all numbers backwards – all numbers except 5 – all digits up to and including 7 – all digits divisible by 3 – sum of all digits – sum of all digits greater or equal 4 – the smallest and largest digit`

- There is a board storing the denominations of money:`int[] money = {1, 2, 5, 10, 20};` and a code example that randomizes one of the indexes of this array:`int idx = (int) (Math.random() * money.length + 1);` Next, there is a program that:
  - will loop randomly until it "accumulates" a value of 100
  - each downloaded value is to be displayed
  - each component sum is to be displayed
  - at the end, the number of withdrawals of money necessary to collect 100 and the amount collected (because it can exceed 100) should be displayed
  - try to modify this program so that it allows you to collect the exact amount (not allow it to be exceeded)

Complete Lesson