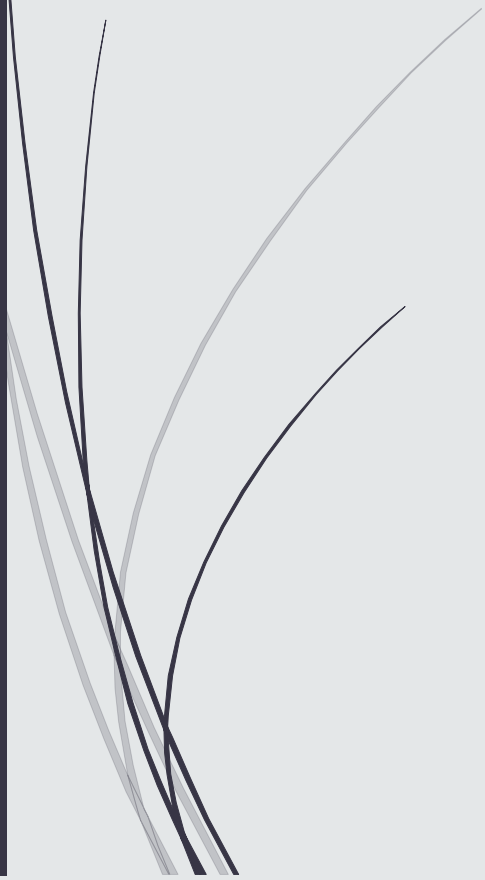
Java Lesson 2

Java From Scratch

Java Output / Print
Java Comments
Java Variables



Content

Java Output / Print

Print Text

You learned from the previous chapter that you can use the println() method to output values or print text in Java:

Example

```
System.out.println("Hello World!");
```

You can add as many println() methods as you want. Note that it will add a new line for each method:

Example

```
System.out.println("Hello World!");
System.out.println("I am learning Java.");
System.out.println("It is awesome!");
```

Double Quotes

When you are working with text, it must be wrapped inside double quotations marks "".

If you forget the double quotes, an error occurs:

Example

```
System.out.println("This sentence will work!");
System.out.println(This sentence will produce an error);
```

The Print() Method

There is also a `print()` method, which is similar to `println()`.

The only difference is that it does not insert a new line at the end of the output:

Example

```
System.out.print("Hello World! ");
System.out.print("I will print on the same line.");
```

Note that we add an extra space (after "Hello World!" in the example above), for better readability.

In this tutorial, we will only use println() as it makes it easier to read the output of code.

Java Output Numbers

Print Numbers

You can also use the println() method to print numbers.

However, unlike text, we don't put numbers inside double quotes:

Example

```
System.out.println(3);
System.out.println(358);
System.out.println(50000);
```

You can also perform mathematical calculations inside the println() method:

Example

```
System.out.println(3 + 3);
```

Example

```
System.out.println(2 * 5);
```

Java Comments

Comments can be used to explain Java code, and to make it more readable. It can also be used to prevent execution when testing alternative code.

Single-line Comments

Single-line comments start with two forward slashes (//).

Any text between // and the end of the line is ignored by Java (will not be executed).

This example uses a single-line comment before a line of code:

Example

```
// This is a comment
System.out.println("Hello World");
```

This example uses a single-line comment at the end of a line of code:

Example

```
System.out.println("Hello World"); // This is a comment
```

Java Multi-line Comments

Multi-line comments start with /* and ends with */.

Any text between /* and */ will be ignored by Java.

This example uses a multi-line comment (a comment block) to explain the code:

Example

```
/* The code below will print the words Hello World
to the screen, and it is amazing */
System.out.println("Hello World");
```

Single or multi-line comments?

It is up to you which you want to use. Normally, we use // for short comments, and /* */ for longer.

Exercise:

Insert the missing part to create two types of comments.

```
    This is a single-line comment
    This is a multi-line comment
```

Java Variables

Variables are containers for storing data values.

In Java, there are different **types** of variables, for example:

- String - stores text, such as "Hello". String values are surrounded by double quotes
- int - stores integers (whole numbers), without decimals, such as 123 or -123
- float - stores floating point numbers, with decimals, such as 19.99 or -19.99
- char - stores single characters, such as 'a' or 'B'. Char values are surrounded by single quotes
- boolean - stores values with two states: true or false

Declaring (Creating) Variables

To create a variable, you must specify the type and assign it a value:

Syntax

```
type variableName = value;
```

Where *type* is one of Java's types (such as int or String), and *variableName* is the name of the variable (such as **x** or **name**). The **equal sign** is used to assign values to the variable.

To create a variable that should store text, look at the following example:

Example

Create a variable called **name** of type String and assign it the value "**John**":

```
String name = "John";
System.out.println(name);
```

To create a variable that should store a number, look at the following

Example

Create a variable called **myNum** of type int and assign it the value 15

```
int myNum = 15;
System.out.println(myNum);
```

You can also declare a variable without assigning the value, and assign the value later:

Example

```
int myNum;
myNum = 15;
System.out.println(myNum);
```

Note that if you assign a new value to an existing variable, it will overwrite the previous value:

Example

Change the value of myNum from 15 to 20:

```
int myNum = 15;
myNum = 20;  // myNum is now 20
System.out.println(myNum);
```

Final Variables

If you don't want others (or yourself) to overwrite existing values, use the final keyword (this will declare the variable as "final" or "constant", which means unchangeable and read-only):

Example

```
final int myNum = 15;
myNum = 20;  // will generate an error: cannot assign a value to a final variable
```

Other Types

A demonstration of how to declare variables of other types:

Example

```
int myNum = 5;
float myFloatNum = 5.99f;
char myLetter = 'D';
boolean myBool = true;
String myText = "Hello";
```

Exercise:

Create a variable named carName and assign the value Volvo to it.

[ ] [ ] = [ ];

Java Print Variables

Display Variables

The println() method is often used to display variables.

To combine both text and a variable, use the + character:

Example

```
String name = "John";
System.out.println("Hello " + name);
```

You can also use the + character to add a variable to another variable:

Example

```
String firstName = "John ";
String lastName = "Doe";
String fullName = firstName + lastName;
System.out.println(fullName);
```

For numeric values, the + character works as a mathematical operator (notice that we use int (integer) variables here):

Example

```
int x = 5;
int y = 6;
System.out.println(x + y); // Print the value of x + y
```

From the example above, you can expect:

- x stores the value 5
- y stores the value 6
- Then we use the println() method to display the value of x + y, which is **11**

Java Declare Multiple Variables

Declare Many Variables

To declare more than one variable of the **same type**, you can use a comma-separated list:

Example

Instead of writing:

```
int x = 5;
int y = 6;
int z = 50;
System.out.println(x + y + z);
```

You can simply write:

```
int x = 5, y = 6, z = 50;
System.out.println(x + y + z);
```

One Value to Multiple Variables

You can also assign the **same value** to multiple variables in one line:

Example

```
int x, y, z;
x = y = z = 50;
System.out.println(x + y + z);
```

Exercise:

Fill in the missing parts to create three variables of the same type, using a comma-separated list:

```
    x = 5    y = 6    z = 50;
```

Java Identifiers

Identifiers

All Java **variables** must be **identified** with **unique names**.

These unique names are called **identifiers**.

Identifiers can be short names (like x and y) or more descriptive names (age, sum, totalVolume).

Note: It is recommended to use descriptive names in order to create understandable and maintainable code:

Example

```
// Good
int minutesPerHour = 60;

// OK, but not so easy to understand what m actually is
int m = 60;
```

The general rules for naming variables are:

- Names can contain letters, digits, underscores, and dollar signs
- Names must begin with a letter
- Names should start with a lowercase letter and it cannot contain whitespace
- Names can also begin with $ and _ (but we will not use it in this tutorial)
- Names are case sensitive ("myVar" and "myvar" are different variables)
- Reserved words (like Java keywords, such as int or boolean) cannot be used as names

Our Java Lessons

Java From Scratch

- Lesson 1 PDF (Java Getting Started)
- Lesson 2 PDF (Java Output, Comments, and Variables)
- Lesson 3 PDF (Java Data Types and Casting)
- Lesson 4 PDF (Java Operators and Strings)
- Lesson 5 PDF (Java Math and Booleans)
- Lesson 6 PDF (JAVA IF ELSE AND SWITCH)
- Lesson 7 PDF (Java While Loop and For Loop)
- Lesson 8 PDF (Java Arrays)