

JAVA

**THE BEGINNERS GUIDE FOR EVERY NON-PROGRAMMER WHICH
WILL ATTEND YOU TROUGH YOUR LEARNING PROCESS**

```
android.support.design.widget.Snackbar extends  
android.webkit.WebView, extends  
android.webkit.WebViewClient;  
@Override  
public void onCreate(Bundle savedInstanceState)  
super.onCreate(savedInstanceState)  
setContentView(R.layout.activity_main)  
public void onClick(View view)  
Intent i = new
```

EDWARD D. ALPHONS

Java

*The Beginners Guide for every non-programmer which
will attend you trough your learning process*

Introduction

I want to thank you and congratulate you for downloading the book, “*Java*”.

I want to start my book with a small motivation for you.

Why should we learn to code?

And why Java?

In my opinion coding is an art. And an Artist has artistic freedom. Everybody who codes is able to create something new. And it doesn't matter if you need your products or if anyone else needs them. Important is your personal development. A programmer will never finish his own development. You can be a student or a pensioner. That will not say anything about your programming skills.

Java is one of the most extensive programming languages and Java developers are very qualified in the IT industry. And it works across many platforms. That means Java applications should usually work on the most operating systems. The most android apps are written in java and the operating system of android devices are based on it. Furthermore, java is a hybrid programming language. It uses several programming paradigms. It is functional and object-oriented. We will learn what this means during the learning progress. Therefore Java is simple to learn

This book contains proven steps and strategies on how to

- Hello World is the traditional beginning of all programming learning processes. Here do you learn how to let you give an output from Java.
- Operators let you do mathematic calculation with given values.

- Loops (I) execute our commands as long as the condition, which we set is fulfilled.
- Primitive Data Types let us work more efficient with datas to write more complex programs. But at first we need only one.
- Loops II is a sequel to Loops I.
- If-clauses can let our program choose decisions after analyzing our current situation.
- And much more!

Thanks again for downloading this book, I hope you enjoy it!

Table of Contents

Introduction

Chapter 1: Hello World

Chapter 2: Operators

Chapter 3: Loops I

Chapter 4: Data Types

Chapter 5: Loop II

Chapter 6: If-clauses

Conclusion

This document is geared towards providing exact and reliable information in regards to the topic and issue covered. The publication is sold with the idea that the publisher is not required to render accounting, officially permitted, or otherwise, qualified services. If advice is necessary, legal or professional, a practiced individual in the profession should be ordered.

- From a Declaration of Principles which was accepted and approved equally by a Committee of the American Bar Association and a Committee of Publishers and Associations.

In no way is it legal to reproduce, duplicate, or transmit any part of this document in either electronic means or in printed format. Recording of this publication is strictly prohibited and any storage of this document is not allowed unless with written permission from the publisher. All rights reserved.

The information provided herein is stated to be truthful and consistent, in that any liability, in terms of inattention or otherwise, by any usage or abuse of any policies, processes, or directions contained within is the solitary and utter responsibility of the recipient reader. Under no circumstances will any legal responsibility or blame be held against the publisher for any reparation, damages, or monetary loss due to the information herein, either directly or indirectly.

Respective authors own all copyrights not held by the publisher.

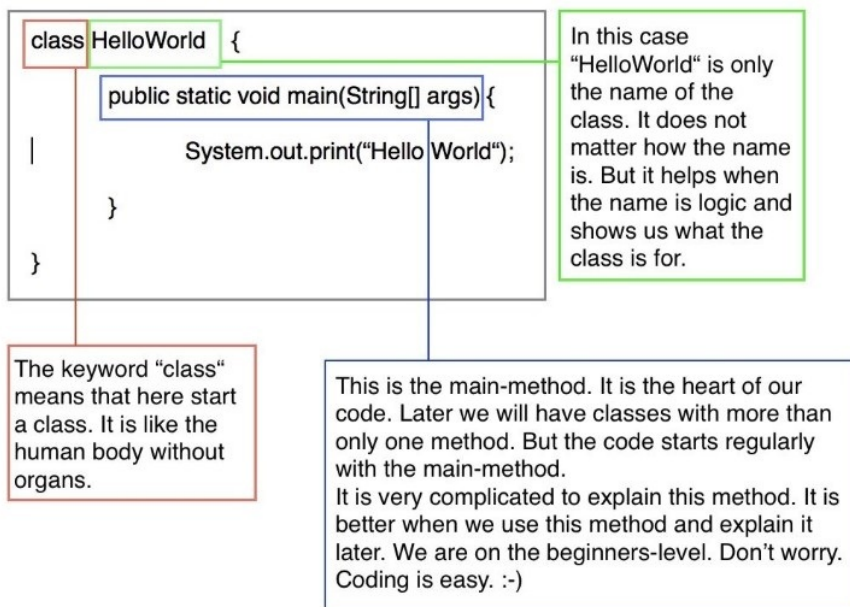
The information herein is offered for informational purposes solely, and is universal as so. The presentation of the information is without contract or any type of guarantee assurance.

The trademarks that are used are without any consent, and the publication of the trademark is without permission or backing by the trademark owner. All trademarks and brands within this book are for clarifying purposes only and are the owned by the owners themselves, not affiliated with this document.

Chapter 1: Hello World

Let's code:

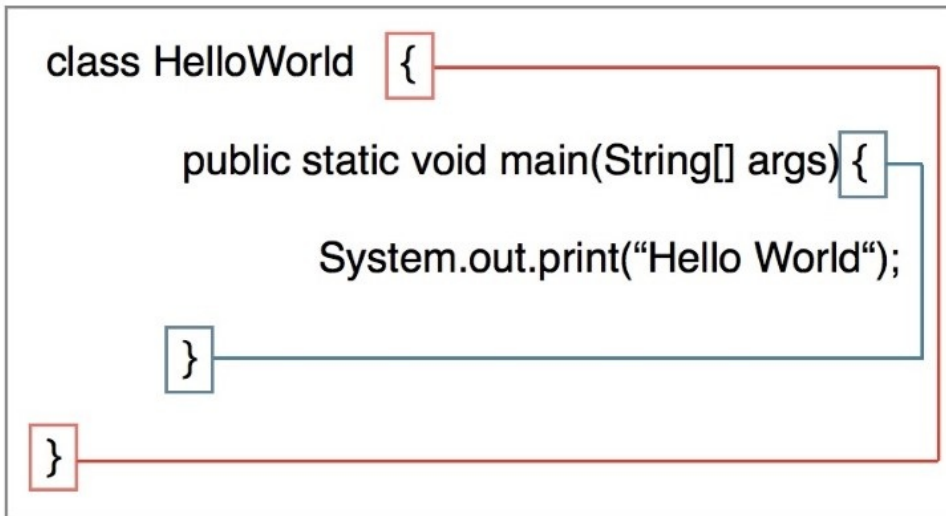
All you need is a computer with any operating system and a simple text editor. We don't need any special programs at first. But you must install the Java Development Kit and the Java Runtime Environment. In the end of the book, you will find some links. Now write the following code in your text editor:



`System.out.print("Hello World")` gives us an output. But which? All text what is written between the quotation marks. In this case: Hello World. The quotation marks must be surrounded by round brackets. Every clause must have a semicolon at the end. The figure shows us this standard.

In the following figure, we can see that we must open and close our classes and methods (for example the main method).

```
class HelloWorld {  
    public static void main(String[] args) {  
        System.out.print("Hello World");  
    }  
}
```



We open and close with curly brackets. They show where a class begins and where it ends. The same principle with the methods. And later we must use the brackets with loops and more.

Briefly: If we open something we must close it.

Now we have the code. But how can we see our output?

We must save our text file. But important is the file extension. it must be .java and not .txt.

A screenshot of a 'Save As' dialog box. The 'Save As:' field contains 'HelloWorld.txt'. The 'Where:' dropdown shows 'Documents — iCloud'. The 'Line endings:' dropdown is set to 'LF (recommended)'. The 'Encoding:' dropdown is set to 'Unicode – UTF-8'. There is an unchecked checkbox for 'Add byte order mark'. At the bottom are 'Cancel' and 'Save' buttons.

False

A screenshot of a 'Save As' dialog box. The 'Save As:' field contains 'HelloWorld.java'. The 'Where:' dropdown shows 'Documents — iCloud'. The 'Line endings:' dropdown is set to 'LF (recommended)'. The 'Encoding:' dropdown is set to 'Unicode – UTF-8'. There is an unchecked checkbox for 'Add byte order mark'. At the bottom are 'Cancel' and 'Save' buttons.

Correct

Now we have a java file in the file location. But how do we start it? The Answer is ,The command line‘. At first, we must open the terminal.

● Windows:

- Press the Windows key and R
- Now type in: “cmd” and then press Enter key

● Mac OS:

- Open the Spotlight search bar (Command + Spacebar)
- Now type in “Terminal” and press the Enter key.

As a beginner maybe you are overwhelmed when you see all that paths and version numbers. But all will be okay after a while. At first, you must know your data path. If you saved your file on the desktop you must type in (in the command line) “cd desktop” and then press the Enter key.

The terminal is in the directory “desktop” now.

We see that “cd” might mean change directory. And yes, this is true. We said to the terminal that it must change the directory to “desktop”.

Hint:

If you don't know what the path of your java file is, you can open the context menu of your file (right click on the file) and open the properties. There you can find the data path.

Now we must convert the java file into a class file. Otherwise, our computer can not execute our program code.

But how do we do that? Yes, with a command in the command line. Very simple. The command is: “`javac YourFileName.java`”

Our java file is a class file and now we can execute it with “`java YourFileName`”

And now we have our first output. We programmed that our command line says us “Hello World”. And it happened.

You officially wrote your first program. It was maybe a hard way but now it will be getting easier.

Chapter 2: Operators

Next, we will let our program calculate. We open our text editor again and code our class and the main method.

We can't simply type in "5 + 5". Maybe you don't know why. Don't worry. The most beginners don't know why java works as it works. We must code that we want an output.

```
class HelloWorld {  
    public static void main(String[] args) {  
  
    }  
}
```

System.out.print();

We say that java gives us an output. But which output? It can't read our minds. We must say it to it. For example "5 + 5". Or maybe without quotation marks? Let's try it!

Execute both opportunities on the next page and watch what happens. Don't read the following page previous to your small experiment.

```
class testing {  
    public static void main(String[] args) {  
        System.out.print(5 + 5);  
    }  
}
```

```
class testing {  
    public static void main(String[] args) {  
        System.out.print("5 + 5");  
    }  
}
```

The result is that the upper code gives us the sum of our operation.

The lower code gives us the text “5 + 5“. But why? The answer is simple. Everything that is written in the quotation marks will be displayed as a text. If we don’t use these marks, java wants to calculate or search. What is meant by search, you will read later in this book.

We can also combine our codes. It could look like that:

```
class testing {  
    public static void main(String[] args) {  
        System.out.print("The sum of 5 and 5 is " + (5 + 5));  
    }  
}
```

Important is the addition sign “+“ and the round brackets. The addition sign tells java that there is more than one output and it must write our text and our operation side by side. What will be our output?

At first, it will give out our text in the quotation marks and then it will calculate our operation in the brackets and write the result beside the text which is written before. That means:

The sum of 5 and 5 is 10.

A few examples for clarification:

```
System.out.print("The sum of 31 and 69 is " + (31 + 69) + "!" );
```

The sum of 31 and 69 is 100!

```
System.out.print("The difference between 100 and 69 is " + (100 - 69) + "!" );
```

The difference between 100 and 69 is 31!

We can simply write several prints consecutively like this:

Please execute your program and look at the output before you read the next sentences.

Your output is:

OneTwo

The problem is that java reads from up to down and from left to right and it does everything that it reads. It writes "One" then "Two". We didn't say that it should make a line break. We can solve the problem with two letters. Instead of `System.out.print();` we must type in

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

The “ln” means line and it ensures that java must enter a line break after our prints.

Chapter 3: Loops I

At next we will get to know loops. Loops are a very important element of every programming language. It provides that we mustn't code the same programming command over and over again.

There are miscellaneous types of loops in java. I will start with the first one. The so-called while-loop. This loop works with a condition. While the condition is not laid down it will carry out the programming commands which are typed in the loop. Like everything else in java, a loop must have a certain syntax. Sometimes examples explain it better than words:

```
class testing {  
    public static void main(String[] args) {  
        while(1 != 0){  
            System.out.print("ha");  
        }  
    }  
}
```

We wrote a loop which executes our print-command as long as 1 is unequal to 0. (Yes, “!=“ means “is unequal“. You should mark it. It will be very important in further programming processes.)

And everybody knows that 1 is NOT 0. That means that our loop will not end. Our program will laugh a whole while. More exact it won't end to laugh. Very funny right?

But we can't only work with loops that will not end.

Chapter 4: Data Types

And exactly here is a good situation to response a very, very important topic which is called data types. What is a data type? We differentiate between primitive data types and reference data types. At first, it will be enough when we employ with the primitive data types.

They are; byte, short, int, long, float, double, char, and boolean. Data types allow us to work efficiently with values. They reserve disk space and save values which can be edited or not.

You should read the following points to prepare a better comprehension of the issue.

- integers are integral numbers.
- float and double are decimal numbers.
- chars are characters, for example, a or b
- boolean is a truth value. It can be either true or false.

There are more than these but they are the most important for the beginning. The others will be described more precise in the following books.

We can do an experiment by working with an integer in a while-loop. At first, we must declare a variable. That means we show java that we will work with values. We need to show which data type java should use to save our values and we

Here we say java which data type we use.

Here we give the integer any name.

```
class testing {  
    public static void main(String[] args) {  
        int number;  
    }  
}
```

must name it.

We can directly initialize our integer. That means that number gets a value.

We can do these steps in just one. It is just a question about your style.

```
int number = 0;
```



```
class testing {  
    public static void main(String[] args) {  
        int number;  
        number = 0;  
    }  
}
```

Any number which we want to give number.

"=" is not a comparative operator. It's an operator which assigns values to e.g. variables.

Chapter 5: Loop II

Now, what happens when we execute the following code:

```
class testing {  
    public static void main(String[] args) {  
        int number = 0;  
        while(number < 5){  
            System.out.print("ha");  
        }  
    }  
}
```

The result will be the same like before. But why? It's very simple. We wrote that the loop must execute the code in his curly brackets, while number is smaller than 5. We initialized number with 0. That all is ok. But we forgot to edit number so that it will grow so that it can reach the value 5 or any else.

Now, we need a command to let number grow. Do you have an idea? We increase in every loop iteration our integer. With the following command it is possible:

```
number + 1;
```

That's all. But under programmers, this notation is not very popular. When we increase our value about only 1 we can write:

```
number++;
```

```
class testing {  
    public static void main(String[] args) {  
        int number = 0;  
        while(number < 5){  
            System.out.print("ha");  
            number++;  
        }  
    }  
}
```

Now our loop will iterate 5 times because number is 0 at the beginning. After the first iteration, it is 1. After the second it is 2 and so on.

This is the principle of the while-loop. There is another but a similar loop. It's the do-while loop.

The declaration is very short.


```
class testing {  
    public static void main(String[] args) {  
        int number = 0;  
        do{  
            System.out.print("ha");  
            number++;  
        }while(number < 5);  
    }  
}
```

The difference to the while-loop is that there is a “do” at the beginning of the loop and the check of the condition is at the end. And the biggest difference is that the do-while-loop will be executed at least one time. It doesn’t matter if the condition is fulfilled because the check is at the end and we learned that java reads from up to down.

The third loop is the for-loop. This loop works a bit different. We must declare a variable (typically an integer) in the loop and then we must say java how to work with them. I will show you.

We begin with the word “for” followed by round brackets. And the most important things

```
class testing {  
    public static void main(String[] args) {  
        for(int i = 0; i < 5; i++){  
            System.out.print("ha");  
        }  
    }  
}
```

happen in these brackets. We need 3

things there. At first, we declare an integer and we directly initialize it. After the semicolon, we set our condition which says java how often the loop should iterate. Here comes our second semicolon. Now we must say how our variable should be edited. In the example, the loop declares the integer “i” and set our condition. The loop must iterate, as long as i is smaller than 5. And after every iterate i will be increased by one.

Which loop you use is your own decision. You must analyze your situation and choose the loop which would be most qualified in your opinion. It doesn't really matter which loop you choose because they do all the same.

Chapter 6: If-clauses

Here comes another very important thing in the programming. The if-clause. It is simple but essentially important to code bigger projects.

The if-clause is used to check our given situation and deal with it. The syntax is basically easy. We say; If... Then....

```
class testing {  
    public static void main(String[] args) {  
        int i = 0;  
        if (i < 5){  
            System.out.print("i is smaller than 5");  
        }  
    }  
}
```

Another example here:

We write “if” and say in the round brackets what we want to examine. In this case, we want to know if i is smaller than 5. In the curly brackets we write what should happen if our examination is positive. Very easy. But it could happen that our examination is negative. If this would happen in our example we won’t have any output. Normally it is our goal to have any output. Although it is positive or negative. In case that i isn’t smaller than 5 we can code an

else case.

```
class testing {  
    public static void main(String[] args) {  
        int i = 5;  
        if (i < 5){  
            System.out.print("i is smaller than 5");  
        }else {  
            System.out.print("i is not smaller than 5");  
        }  
    }  
}
```

We write “else“ after the curly bracket

from the if-clause it relates to and set new curly brackets with the commands what should happen if the examination from our if-clause is not positive. We have the possibility to make an “else if“. That means if our examination is negative we examine something else or

```
class testing {  
    public static void main(String[] args) {  
        int i = 5;  
        if (i < 5){  
            System.out.print("i is smaller than 5");  
        }else if (i == 5) {  
            System.out.print("i is exactly 5");  
        }  
    }  
}
```

examine it again.

If you are wondering why we have two equality-signs, here comes the explanation. If we use only one equality signs, java tries to initialize values. If we use two java tries to compare. Sometimes coders make simple mistakes and when they search the fault they see it was a simple thing like using one equal sign instead of two. But that is normal.

We can certainly combine else if and else.

```
class testing {  
    public static void main(String[] args) {  
        int i = 9;  
        if (i < 5){  
            System.out.print("i is smaller than 5");  
        }else if (i == 5) {  
            System.out.print("i is exactly 5");  
        }else{  
            System.out.print("i is bigger than 5");  
        }  
    }  
}
```

And we can nest or interleave if-clauses. That means that if our examination is positive we make another examination. That can be helpful in some situations. Therefore we can simply write a new if-clause in the curly brackets of the previous if-clause.

You learned the basics of java programming and you have enough knowledge to code small programs like a calculator. In my following books, we will be faster and we go deeper in the subject matter. But if you understood this book you won't have any problems with the further books.

Conclusion

Thank you again for downloading this book!

I hope this book was able to help you to leaning programming and be motivated to learn much more.

The next step is to understand this literature and make some experiments with your knowledge. If you think you can understand the following books. They will help you to intensify your knowledge.

The next books will be very enormous. it will not be like this one. This book should be just an introduction in the programming.

Finally, if you enjoyed this book, then I'd like to ask you for a favor, would you be kind enough to leave a review for this book on Amazon? It'd be greatly appreciated!

Thank you and good luck!