



# **Training Application Developer Programming curriculum**

## **Introduction to Java programming**

*Domain A Level 1*

*Author: Harmen Steenbergen*

*Date: 8-5-2023*

*Version: 1.3*

# Table of contents

Overview .....	3
Learning objectives .....	3
Program .....	3
Fence .....	3
Part 1: Introduction Java (4 hours).....	4
Getting started with Java .....	4
Exercises .....	4
Part 2 Data (4 hours) .....	5
Variables and operations .....	5
Exercises .....	6
Part 3: Making Choices (8 hours) .....	7
Conditional code .....	7
Exercises .....	7
Part 4: Making loops (12 hours) .....	10
Repeating code.....	10
Exercises .....	10
Part 5 Arrays (16 hours) .....	12
Multiple variables.....	12
Exercises .....	13
Part 6 Object (4 hours) .....	16
Use a Custom Class.....	16
Final assignment (32 hours) .....	17
Assessment.....	19
Annex 1: Integrated Development Environments (IDE) for Java .....	20



Annex 2: Internet Resources .....	23
Annex 3: Java Operators .....	24
Numerical operators .....	24
Relational operator .....	24
Binary operators.....	24
Logical operators.....	24
Toewijzingsoperatoren.....	24

## Overview

Level:	Domain A Level 1
Expensive:	40 hours (if familiar with basic programming structures) 80 hours
Method:	Weekplanning

## Learning objectives

- Installing and using a Java development environment (IDE)
- Create, compile, and launch a Java program.
- Be able to use primitive variables in Java.
- Be able to perform operations with primitive variables.
- Create conditional code.
- Create repeating code
- Using arrays
- Simple object creation and use

## Program

Part 1: Getting started with Java  
Part 2: Variables and operations  
Part 3: Conditional code  
Part 4: Repeating code  
Part 5: Multiple variables  
Part 6: Using a Custom Class

## Fence

Eindopdracht

## Part 1: Introduction Java (4 hours)

### Getting started with Java

Read about different IDEs and choose which IDE you want to use. Install the IDE and program, compile and launch HelloWorld.java. For an overview of the IDEs, see Annex 1.

If you can't choose from the IDEs then use IntelliJ IDEA, the IDE where you can later create Android Apps is based on this IDE. You can request a full version of the IntelliJ IDEA by applying for an

```
public class HelloWorld {

    public static void main(String[] args) {
        System.out.println("Hello, World");
    }

}
```

educational license with your Drenthe College email address.

Results:

- You have a working IDE in which you can create, compile and test Java programs.
- You are familiar with the naming of Java source files.
- You can manage the different project/programs in your IDE. You know the functions within your chosen IDE for managing different code files, such as projects, workspace, tool sets, etc.
- You know how to back up the projects you create.

### Exercises

#### Exercise 1 Hello World

Install the JDK and an IDE of your choice (see overview). Create a working 'Hello World' program that can run without errors in your IDE.

#### Exercise 2: Make Mistakes

If you haven't already, make a mistake in your code and see how the IDE responds when the program tries to run. Search the error messages to see if you can find which line of code found an error.

#### Exercise 3: Find the Files

Search your computer's disk for the source files of your first program. Also look at which folders and files are in this place, and find out what they are for.

### Exercise 4: Run your program

Your project folder also contains a HelloWorld.class file. Figure out how to launch it without using the IDE.

### Exercise 5: Search the Resources

In the resource list there are several links to sites where you can find more about programming in Java. Take a look at the different sites and make a top 3 for yourself.

## Part 2 Data (4 hours)

### Variables and operations

In Java you can use variables of the so-called primitive types. Find out what primitive types there are

```
byte a = 20;
short b = 21;
int c = 200409;
long d = 2348902;
float e = 2.30f;
double f = 20399.345d;
boolean g = true;
char h = 'c';
```

and how to declare a variable.

Variables allow you to perform arithmetic and other operations. You do this by using different operators. For example, you can add two variables of the type integer together. So, in this example,

```
int a, b, am;
a = 10;
b = 15;
som = a + b;
```

the plus sign (+) is the operator. Note: for float and double there is a letter after the number, f or d.

Find the different operators and use examples in Java to see how they work. In appendix 3 there is a list of operators in Java, of which you only need to know the bold. The italics (= italics) printed operators are optional.

Results:

- You know the different primitive type for variables that you can use within Java.
- You can do math and operations with all types



## Exercises

### Exercise 6 Types

Make a list of the different primitive types that Java has. Make each type a variable, give this variable a value.

### Exercise 7 Operators

Examine the workings of the different operators with variables of different types. Keep your code as simple as possible and show the result of the operation on the console (see HelloWorld).

### Exercise 8 Consultation

Consult with a classmate what the different operators do and come up with examples in Java together how you can use them.

## Part 3: Making Choices (8 hours)

### Conditional code

One of the most powerful possibilities in programming is to execute (pieces of) code based on a condition. For example, the value of a variable allows you to determine whether or not a piece of code is executed. In Java you have a number of possibilities:

- if structure

```
if ( ... ) {
    ...;
}
else {
    ...;
}
```

- Switch structure

```
switch( ... ) {
    case ...:    ... break;
    case ...:    ... break;
    case ...:    ... break;
    .....
}
```

- Ternary operator

Find out how the different structures and the operator work and make sure you can use them in a Java program.

Result:

- You know the syntax of the if, switch and the ternary operator and can use them in your own code.

### Exercises

#### Exercise 9: Use the arguments

Copy the following code:

```
public class ConsoleApp {

    public static void main(String[] args) {
        System.out.println(args[0]);
    }

}
```

In this Java application you use the arguments that the program receives with the call. To give arguments to your program, you can start your program at the command prompt (in a so-called dosbox to start with 'cmd'):

java ConsoleApp test

You must be in the folder where the ConsoleApp.class file is located. The Java program must be located on your computer (the path environment variable must be set correctly). The word *test* is the parameter that is given.

You can also start your program with arguments/parameters via your IDE. Find out how to do that and test the program above.

### Exercise 10 Making decisions

Create a console program that tests the value of the argument. The output must be correct according to the table below:

Value argument	Export
Less than 5.5	Insufficient
Greater than or equal to 5.5	Sufficient

Note: you must convert the argument (String) to a comma number (double) before you can test it! So google again!

### Exercise 11: Multiple Arguments

If you enter multiple arguments separated by a space in the call, the first argument will be in args[0], the second in args[1] etc.

Create a console program that reads three arguments and if:

Argument	Export
+	Arguments 2 and 3 added together
–	Argument 2 min argument 3
X	Argument 2 and 3 multiplied
/	Argument 2 divided by argument 3

### Exercise 12 Switch

Try creating the previous exercise with a switch structure. Note: a switch where you check a String variable can only be done from Java 8, so check your version.





### Exercise 13 Test numbers

Create a console program that reads the arguments and:

- the first argument is the text one, two, three, four, ... nine or ten
- the second argument is a number from 1 to 10
- the output is "equal" if the two arguments are equal, otherwise the output is "unequal".

Use a switch and/or an if statement where you think it's useful.

### Exercise 14 Divisible

Create a console program that uses two numbers as an argument and shows in the output whether the second argument is a divisor of the first argument:

Example:

Arg1	Arg2	Export
12	3	Parts
12	7	no divisor
4	2	Parts

## Part 4: Making loops (12 hours)

### Repeating code

There are several ways to run a piece of code multiple times. In addition to conditional code, this is the most important structure in your code. In Java, you can use the following iteration structures:

- `for ( ...; ...; ...) { ... }`
- `while (...) { ... }`
- `do { ... } while (...);`

Find out how the three structures work and make sure you can use them in a Java program.

Result:

- You know the for and the while structure and can use it in your own code.

### Exercises

#### Exercise 15 Creating loops

Create a console program that uses the value of the first argument (a number) to repeat the value of the second argument (a text) on the screen. Of course, just as often as the value of the first argument.

So:

```
C:\> Java JeProgramma 4 "One line of text"
```

Outputs:

```
A line of text A line of text A line
of text
A line of
text
```

## Exercise 16: Repeat Input

The example below reads input on the console and stores it in the variable `import`. After that, the

```
import java.io.Console;

public class ConsoleApp {

    public static void main(String[] args) {
        Console with = System.console();
        String import = with.readLine();
        System.out.println(import);
    }
}
```

value of `import` is put on the screen.

Create a program that reads the input in a loop. If the input is not empty, it is added to the previous entry. If it is empty, all previous input is put on the screen.

So:

```
C:\> Java JeProgramma
```

Outputs:

One	(input)
rule	(input)
text	(input)
One line of text	(export)

## Exercise 17 Faculty

The factorial of number  $n$  is a calculation in which you multiply the numbers from 1 to  $n$ . The faculty is noted with an exclamation point. So  $n! = 1 \times 2 \times 3 \times \dots \times n$ .

N	N!
1	1
2	$1 \times 2 = 2$
3	$1 \times 2 \times 3 = 6$
4	$1 \times 2 \times 3 \times 4 = 24$
5	$1 \times 2 \times 3 \times 4 \times 5 = 120$

Create a program that calculates the argument from the faculty and shows it on the screen.

## Exercise 18: Splitting Text

Create a program that splits the input (text) into single characters.

So:

```
C:\> Java JeProgramma "Example"
```

Outputs:

Preview

## Exercise 19 Rectangle

Create a program that uses a specified character to draw a rectangle. The size of the rectangle is given by the second the third argument.

So:

```
C:\> Java JeProgramma "o" 4 6
```

Outputs:

```
oooo
ooooo oooo
 oooo oooo
 oooo
```

# Part 5 Arrays (16 hours)

## Multiple variables

You can temporarily store one value in a variable. To store different values, you need multiple variables, with all variables having different names. If you need to store multiple values of the same type, you can also use an array.

Find out how your arrays work in Java and make sure you can use them in a Java program.

Result:

- You can create, use, and customize arrays in Java.
- You can use an array in your own code.

## Exercises

### Exercise 20 Weekdays

Create a console program where the value of the argument is a number from 1 through 7. Then the program shows the weekday that belongs to the song. Use an array to store the weekdays.

### Exercise 21: Save Input

The example below reads input on the console and stores it in the variable input. After that, the value of input is put on the screen.

```
import java.io.Console;

public class ConsoleApp {

    public static void main(String[] args) {
        Console with = System.console();
        String import = with.readLine();
        System.out.println(import);
    }
}
```

If you want to use the System.console in a program, it often won't work in the console within your IDE. Then you need to start java program on the command-line. Alternatively, you can use the scanner.

```
import java.util.Scanner;

public class ConsoleApp {

    public static void main(String[] args) {
        Scanner scanner = new Scanner(System.in);
        String import = scanner.next();
        System.out.print(import);
        scanner.close();
    }
}
```

Create a program that reads the input in a loop. If the input is not empty, it is added to an array. The program must be able to handle up to 25 input data and store it in that array. If the input is empty, you must stop retrieving entered text.

### Exercise 22: Printing the input

Use the code from the previous exercise (if all goes well, the reading is already good). Then print all input on the screen. If this succeeds, put the input in reverse order (last entry first) on the screen. You don't have to change anything about the input (it already works) but only the order of on the screen changes.

### Exercise 23 Arrays in Two Dimensions

You can also create a two-dimensional array. Then you don't have a list of data, but a block of data. In this way, you can save a grid of characters, for example:

```
(^V^)  (*>
<( )>  (> )
W W    >>
```

In your program, create a 2-dimensional array, in which such a drawing stores and make sure that your program shows it on the screen when running.

### Exercise 24 Search

Create two arrays:

```
{"yellow", "orange", "green", "white", "blue"}
```

```
{"banana", "orange", "apple", "coconut", "grape"}
```

Create a console program to which you can give an argument. The argument is looked up in the array of colors. If the color is found, the matching fruit is exported, otherwise you will get the output "not found".

### Exercise 25 Sorting

Create a console program that requests input multiple times. The input is stored in an array. If the input is empty, the array is sorted alphabetically and displayed on the screen.

The first part (the input) you have already made (exercise 21). Allows for up to 25 input data.

Find yourself a sorting routine for sorting an array in Java. Note that many routines for sorting data are even made into entire studies. Find yourself a simple routine to use.

## Part 6 Object (4 hours)

### Use a Custom Class

In Java, all code is part of a class. In the first code you used, you already created the *HelloWorld* class. In addition to the class you start and classes from the libraries you can use, you can also create your own classes. Any serious Java program will consist of several classes. Figure out how to create your own class in Java and incorporate it into your own code.

Find out what a *constructor of the class is and incorporate a constructor into your own code.*

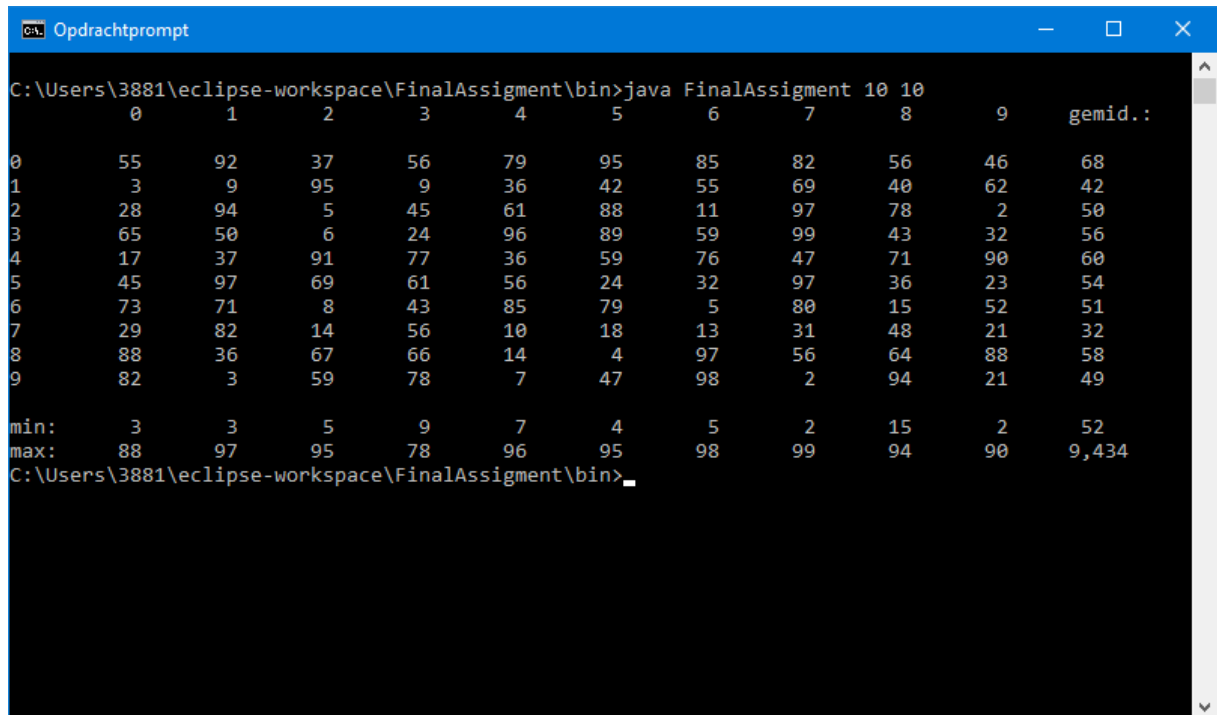
Results:

- You know how to create a class
- You know how to use a class
- You know what a *constructor* is



## Final assignment (32 hours)

Create a java program that produces the following output:



```

C:\Users\3881\eclipse-workspace\FinalAssigment\bin>java FinalAssignment 10 10
  0      1      2      3      4      5      6      7      8      9      gemid.:
0      55      92      37      56      79      95      85      82      56      46      68
1       3       9      95       9      36      42      55      69      40      62      42
2      28      94       5      45      61      88      11      97      78       2      50
3      65      50       6      24      96      89      59      99      43      32      56
4      17      37      91      77      36      59      76      47      71      90      60
5      45      97      69      61      56      24      32      97      36      23      54
6      73      71       8      43      85      79       5      80      15      52      51
7      29      82      14      56      10      18      13      31      48      21      32
8      88      36      67      66      14       4      97      56      64      88      58
9      82       3      59      78       7      47      98       2      94      21      49

min:     3       3       5       9       7       4       5       2      15       2      52
max:    88      97      95      78      96      95      98      99      94      90     9,434
C:\Users\3881\eclipse-workspace\FinalAssigment\bin>
  
```

### Explanation

- The number of rows and columns that are displayed is given to the program as an argument by means of two numbers. The above output is the result of the java `FinalAssignment 10 10` call where `FinalAssignment` is the name of the class that contains the main method.
- The code uses an array of two dimensions to (temporarily) store the values.
- The table in the output is tabbed
- The numbers in the table are randomly generated numbers from 1 through 100.
- The right column shows the average value of the numbers in one row.
- Below the table are two rows with the minimum and maximum values in the column.
- At the far right of the min row is the average of the average values of the rows (the right column). In the example, it's 52.

The challenge:

- At the far right of the row max is the standard deviation calculated over the average values of the rows (the right column). The standard deviation is represented by three decimal places. In the example, it's 9,434.

```

C:\Users\3881\eclipse-workspace\FinalAssigment\bin>java FinalAssigment 4 4
      0      1      2      3      4
gemid.:
0      20      65      9      18      28
1      28      77      93      57      63
2      22      56      32      100     52
3      36      98      84      65      70

min:    20      56      9      18      53
max:    36      98      93      100     15,930
C:\Users\3881\eclipse-workspace\FinalAssigment\bin>
    
```

### Hand in

When you have finished the assignment, create a zip file from your project folder. You hand this in in the assignment in Magister.

## Assessment

The assignment will be assessed on the following points:

- Operation according to command 50 points
- Markup code (indent and whiteregels) 10 points
- Naming in code, for example, variable names. 10 points
- Comment in code, each piece is provided with commentary indicating what happens in that piece. 5 points
- Use of array(s) 25 points
- Use of loop structure (for/while) 25 points
- Use of conditional code (if) 25 points

## Appreciation

+	0	1	2	3	4	5	6	7	8	9
0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
10	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
20	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
30	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
40	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0	1,0
50	1,0	1,0	1,0	1,5	1,5	1,5	1,5	1,5	1,5	2,0
60	2,0	2,0	2,0	2,0	2,5	2,5	2,5	2,5	2,5	2,5
70	3,0	3,0	3,0	3,0	3,0	3,5	3,5	3,5	3,5	3,5
80	3,5	4,0	4,0	4,0	4,0	4,0	4,0	4,5	4,5	4,5
90	4,5	4,5	5,0	5,0	5,0	5,0	5,0	5,0	5,5	5,5
100	5,5	5,5	5,5	6,0	6,0	6,0	6,0	6,0	6,0	6,5
110	6,5	6,5	6,5	6,5	7,0	7,0	7,0	7,0	7,0	7,0
120	7,5	7,5	7,5	7,5	7,5	8,0	8,0	8,0	8,0	8,0
130	8,0	8,5	8,5	8,5	8,5	8,5	8,5	9,0	9,0	9,0
140	9,0	9,0	9,5	9,5	9,5	9,5	9,5	9,5	10,0	10,0

# Annex 1: Integrated Development Environments (IDE) for Java

## IntelliJ IDEA

*Link*

<https://www.jetbrains.com/idea/download/>

*License/Price*

Community Edition Apache 2 license / free  
Ultimate Edition / Personal Licence: €159,- For students possible to get for free

*Available for  
Description*

Windows, MAC OSX on Linux  
An IDE that is easy to install and works out-of-the-box as much as possible. Many additional features are already built-in without the need for plugins. This prevents problems with missing plugins when importing projects. The standard IDE for creating an Android App, Android Studio, is based on IntelliJ and has the same *look-and-feel*. That is why this IDE is preferable if you want to make Android Apps (later).

## Eclipse

*Link*

<http://www.eclipse.org/downloads/>

*License / Price*

EPL / free

*Available for  
Description*

Windows, MAC OSX on Linux  
Complete IDE for programming in Java and other languages. Eclipse can be fully expanded with plug-ins and customized. Can also be used with a Dutch interface. Code is checked and possibilities suggested. Customization in Eclipse can be difficult due to the many settings and possibilities.  
In many (older) online Java tutorials, Eclipse is used as an IDE.

## Netbeans

*Link*

<https://netbeans.org/downloads/index.html>

*License / Price*

CDDL/GPL / free

*Available for  
Description*

Windows, MAC OSX on Linux  
Netbeans is an extended IDE for programming mainly in Java. Code is checked as you type and proposals are made for finishing the code. There are several versions of Netbeans, including a complete version with an integrated Glassfish server allowing development of jsp pages.

## JCreator

<i>Link</i>	<a href="http://www.jcreator.org/download.htm">http://www.jcreator.org/download.htm</a>
<i>License/Price</i>	\$35 license
<i>Available for</i>	Windows
<i>Description</i>	"JCreator is a simple and lightweight JAVA IDE from XINOX Software. It runs only on Windows platforms. It is very easy to install and starts quickly, as it is a native application. This is a good choice for beginners."

## BlueJ

<i>Link</i>	<a href="http://www.bluej.org/">http://www.bluej.org/</a>
<i>License/Price</i>	GPL2: free
<i>Available for</i>	Windows, MAC OSX, Linux(Ubuntu/Debian) en Raspberry PI
<i>Description</i>	<p>BlueJ has a simple interface that works from the class diagram of the application being developed. This gives you a good overview of the classes and objects. It can compile and run your code.</p> <p>"BlueJ is an IDE that includes templates and will compile and run the applications for you. BlueJ is often used by classes because it is not necessary to set classpaths. BlueJ has its own sets of libraries and you can add your own under preferences. That sets the classpath for all compilations that come out of it to include those you have added and the BlueJ libraries.</p> <p>BlueJ offers an interesting GUI for creation of packages and programs. Classes are represented as boxes with arrows running between them to represent inheritance/implementation or if one is constructed in another. BlueJ adds all those classes (the project) into the classpath at compile time."</p>

## DRJava

<i>Link</i>	<a href="http://www.drjava.org/">http://www.drjava.org/</a>
<i>License/Price</i>	BSD License: free
<i>Available for</i>	Windows, MAC OSX and as jar-file
<i>Description</i>	Simple Java editor with ability to compile and start the code. Self-written in Java and works on any machine with a JRE installed.

## geany

<i>Link</i>	<a href="http://www.geany.org/Download/Releases">http://www.geany.org/Download/Releases</a>
<i>License/Price</i>	GPL: Free
<i>Available for</i>	Windows
<i>Description</i>	Geany is a light IDE made for different platforms and in addition to Java, suitable for different languages, including C, C++, C#, Java, JavaScript, PHP, HTML, LaTeX, CSS, Python, Perl, Ruby, Pascal, Haskell, Erlang and Vala.

## Processing

<i>Link</i>	<a href="https://www.processing.org/download/">https://www.processing.org/download/</a>
<i>License/Price</i>	GPL/LGPL, gratis,
<i>Available for</i>	Windows, MAC OSX on Linux
<i>Description</i>	Insimplified Java to create visual products in both 2D and 3D through programming. Processing has been specially developed as a tool to learn programming.

## Zeus Lite

<i>Link</i>	<a href="http://www.zeusedit.com/lite/index.html">http://www.zeusedit.com/lite/index.html</a>
<i>License/Price</i>	limited version free
<i>Available for</i>	Windows
<i>Description</i>	Zeus is a simple IDE for several languages including Java. Zeus continuously compiles the code in the background and shows the structure of the code in different colors. <sup>3</sup>

## Zeus IDE

<i>Link</i>	<a href="http://www.zeusedit.com/index.html">http://www.zeusedit.com/index.html</a>
<i>License/Price</i>	licentie €89
<i>Available for</i>	Windows
<i>Description</i>	Zeus is a simple IDE for several languages including Java. Zeus continuously compiles the code in the background and shows the structure of the code in different colors. <sup>3</sup>

## JDeveloper

<i>Link</i>	<a href="http://www.oracle.com/technetwork/developer-http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html">http://www.oracle.com/technetwork/developer-http://www.oracle.com/technetwork/developer-tools/jdev/overview/index.html</a>
<i>License/Price</i>	free
<i>Available for</i>	Windows, Linux in Generic
<i>Description</i>	Oracle JDeveloper is a free integrated development environment that simplifies the development of Java-based applications addressing every step of the application lifecycle. JDeveloper offers complete end-to-end development for Oracle's platform and Oracle's applications.

## jEdit

<i>Link</i>	<a href="http://www.jedit.org/index.php?page=download">http://www.jedit.org/index.php?page=download</a>
<i>License/Price</i>	GPL2: free
<i>Available for</i>	Windows, MAC OSX, Linux (Debian/Slackware) and as jar-file
<i>Description</i>	Simple Java code editor, without standard possible to compile and start the code.

## Annex 2: Internet Resources

[https://nl.wikibooks.org/wiki/Programmeren\\_in\\_Java](https://nl.wikibooks.org/wiki/Programmeren_in_Java)

<https://docs.oracle.com/javase/tutorial/>

<http://docs.oracle.com/javase/tutorial/java/>

<http://www.tutorialspoint.com/java/>

<https://www.udemy.com/java-tutorial/> (clips, eclipse)

<http://www.learnjavaonline.org/> (online IDE)

<https://www.youtube.com/watch?v=3u1fu6f8Hto> (eclipse)

[https://www.youtube.com/watch?v=uYnY\\_7V\\_I6I](https://www.youtube.com/watch?v=uYnY_7V_I6I) (netbeans)

<https://www.thenewboston.com/videos.php?cat=31> (clips, eclipse)

<https://netbeans.org/kb/articles/learn-java.html> (netbeans)

## Annex 3: Java Operators

### Numerical operators

#### Numeric operators

+	add
-	subtract
*	multiply
/	share
%	Rest after division
++	increase
--	lower

### Relational operator

#### Relational operators

==	Similar to
!=	not equal to
>	greater than
<	less than
>=	greater than or equal to
<=	less than or equal to

### Binary operators

#### Bitwise operators

&	binary EN
	binaire OF
^	binaire exclusieve OF
~	binary NOT
<<	shift to the left
>>	shift to the right
>>>	shift to right and fill with zero

### Logical operators

#### Logical operators

&&	IN
	OF
!	NOT

### Toewijzingsoperatoren

#### Assignment operators

=	assign to
+=	Add to
-=	subtract from
*=	multiply by
/=	Share by
%=	remainder after division by
<<=	Shift to the left with
>>=	shift to the right with
&=	AND with ...
^=	exclusieve OF met ...
=	OF met ...