## horizontal line



Compiler & Operating Systems

09.02.2021

**─**

Group:

Jóhan Abrahamsen - 478250

Max Sterken - 485736

# Overview

# Goals

# Specifications

# Milestones

# Manual for everyone:

Arithmetic Operators

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| + | add | Arithmetic | + (Addition) |
| - | minus | Arithmetic | - (Subtraction) |
| \* | times | Arithmetic | \* (Multiplication) |
| / | div | Arithmetic | / (Division) |
| ++ | increm | Arithmetic | Increment |
| -- | decrem | Arithmetic | Decrement |

Assignment Operator

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| = | is | Assignment | Assigning |

Relational Operator - Comparison Operator

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| == | equals | Relational Operators | Is Equal |
| != | not\_equals | Relational Operators | Not Equal |
| > | bigger | Relational Operators | Greater than |
| < | smaller | Relational Operators | Less than |

Logical Operator

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| && | and | Logical Operator | Returns true if, both statements are true |
| || | or | Logical Operator | Returns true if one of the statement is true |

Looping & Condition

|  |  |  |  |
| --- | --- | --- | --- |
| Java Loops & Checking | Maximus Loops & Condition | Java Loop or Condition statement | Maximus loops & Condition |
| For loop | Loop[x]Times{} | For (int i = 0; i < j.lenght(); i++{} | repeat(N) |
| While loop | doOn | While(condition is true or false) { } | doOn(condition) |
| If/elseif/else statement | condition(condition){  }notMet{  } | if(i ==0){  }  elseif(i ==1){  }  Else{  System.out.println(“stress”); } |  |
| Print String | ShowString(string) |  |  |
| Print Int | ShowInt(int) |  |  |
| Print boolean | ShowBool(boolean) |  |  |
| Scanner | ask() |  |  |

Operator ; - end sentences

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| ; | $ |  | End sentence |

Primitive Data Types

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| Int (4 bytes) | num | Primitive Data Type | Stores whole numbers from -2,147,483,648 to 2,147,483,647 |
| Double (8 bytes) | numnum | Primitive Data Type | Stores fractional numbers. Sufficient for storing 15 decimal digits |
| Boolean (1 bit) | Booly (can be valid or invalid) | Primitive Data Type | Stores true or false values |

Non-Primitive Data Types

|  |  |  |  |
| --- | --- | --- | --- |
| Java Operators | Maximus Operator type | Operand Type | Description |
| Class | Blueprint Animal{} | Non-Primitive Data Type | Class Animal{  } |
| Object | Animal animal = create Animal()  (Object call method with >, example animal>makeNoise() | Non-Primitive Data Type | Animal animal = new Animal() |
| String | CharCollection | Non-Primitive Data Type | String abc = "Educative is an excellent tutor"; |
| Array\* | Arr | Non-Primitive Data Type | Array a1 = {“a”, “b”} |

Functions:

<return type> <name>(<arguments>) {

<code>}

|  |  |  |  |
| --- | --- | --- | --- |
| return | result | Function return | Return statement |

Generic functions:

Translate to our code:

**function** f( value: integer ) -> integer

**return** value + 2 \* 4

**end**

**do**

**ask** user for a *number*

**until** *number* is between 2 and 10

**for** *i* **from** 1 **to** *number*

**if** *i* < 5 **then**

**print** f(i)

**else**

**print** i

**end**

**print** “Done”

num f(num value){

result value add 2 times 4}$

Num number is ask()$

doOn(number bigger 2 and number smaller 10){

loop[number]times{

condition(repeatCount smaller 5){

showConsole(f(repeatcount))}

notMet{

showconsole(repeatcount)}

}

}

showconsole(“Done”)$

**Example 2 Class:**

Blueprint Animal {

CharCollection type = “Cow”$

CharCollection Noise = “Mooo”$

CharCollecion makeNoise(CharCollection type){

result Noise$}

}

**Example 3 Loop:**

booly loop is valid$

doOn(loop){

Loop[10]times{

showString(“hello”)$

}

Loop is invalid$

}

**Example 4 Object:**

Animal cow is create Animal(“cow”, “moo”)$

cow>makeNoise()$

**Example 5 Math:**

numnum gradeFirstAssignment is 5.5$

numnum gradeSecondAssignment is 7.0$

numnum finalGrade is (gradeFirstAssignment add gradesecondassignment) div 2$