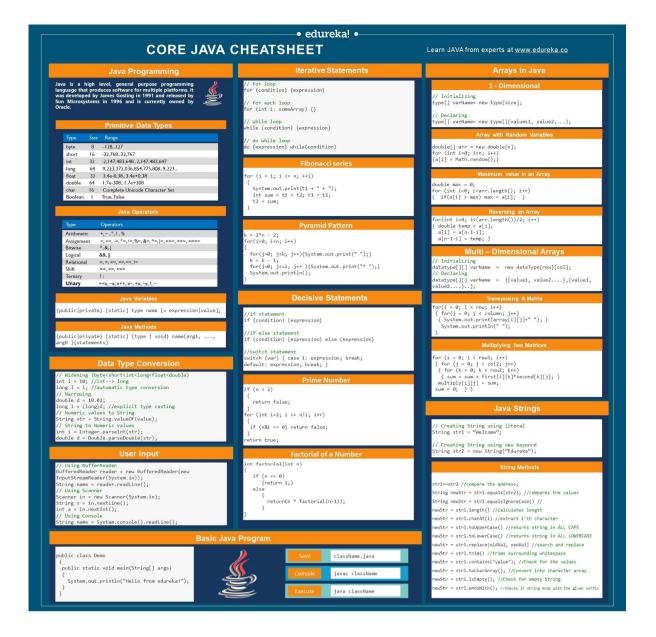
Java Tutorial Java Methods Java Methods Java HOME Java Intro Java Method Overloading Java Get Started Java Scope Java Syntax Java Recursion Java Output Java Comments Java Classes Java Variables Java OOP Java Data Types Java Classes/Objects Java Type Casting Java Class Attributes Java Operators Java Class Methods Java Strings Java Math Java Constructors Java Booleans Java Modifiers Java If...Else Java Encapsulation Java Packages / API Java While Loop Java For Loop Java Inheritance Java Break/Continue

Java Polymorphism Java Exceptions



Java Tutorial

Husk dette om Java Syntax:

- in Java must be inside a class
- The name of the java file must match the class name. When saving the file, save it using the class name and add ".java" to the end of the filename
- The main() method is required and you will see it in every Java program:
- public static void main(String[] args) The curly braces {} marks the beginning and the end of a block of code.
 De bruges hver gang man laver Tab

- Husk dette om Java Output:

 You can use the println() method to output values or print text in Java
- 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

Husk dette om Java Comments:

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

Husk dette om Java Variables:

- JSK DECTEO OM JAVA Val'IADIES:

 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 declimals, 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 "2" or "B'. Char values are surrounded by single quotes

 boolean stores values with two states: true or false
- - type variableName = value
- · 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):
- · Anderledes ved floats:
 - o En float skal deklareres med tallet som normalt, efterfuldt af et "f". Ellers bliver værdien en double.
- Flere variabler i samme linje:

Declare Many Variables



One Value to Multiple Variables

Husk dette om Java Data Types:

- String skal skrives med stort, alt andet med småt.
 Data types are divided into two groups:
 Primitive data types includes byte, short, int, long, float, double, boolean and char
 Non-primitive data types such as String, Arrays and Classes (you will learn more about these in a later chapter)
- Numbers

Integer types stores whole numbers, positive or negative (such as 123 or -456), without decimals. Valid types are byte, short, int and long. Which type you should use, depends on the numeric value.

Husk dette om Java Type Casting:

Java Type Casting

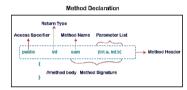
Type casting is when you assign a value of one primitive data type to another type.

- In Java, there are two types of casting:
- Widening Casting (automatically) converting a smaller type to a larger type size byte -> short -> char -> int -> long -> float -> double
- Narrowing Casting (manually) converting a larger type to a smaller size type double -> float -> long -> int -> char -> short -> byte

Java Methods

Husk dette om Java Methods:

- Samme som funktioner i Python
 A method must be declared within a class.
- Syntax:



Access Specifiers:

- public/private - Der kan også skrives andre modifiers: o https://www.w3schools.com/JAVA/java modifiers.asg

- Datatype (int, String, osv) eller void

Husk dette om Java Method Parameters:

• Ligesom i python, bortset fra at man skal skrive data-type på en parameters.

Husk dette om Java Method Overloading:

- Til når man skal have en method som kan tage imod flere forskellige datatyper. . With method overloading, multiple methods can have th
- Example int myMethod(int x)
 float myMethod(float x)
 double myMethod(double x, double y)

Husk dette om Java Scope:

Husk dette om Java Recursion:

Java Classes

Husk dette om Java OOP:

Afsnit med forklaring af og fordele ved objekt orienteret programmering

Husk dette om Java Classes/Objects:
Ingen_init_() - i stedet skrives disse ting blot i scopet for ens class. SE UNDER "Java Constructors"!
Man laver en .java-fil til HVER CLASS.

Using Multiple Classes

Remember that the name of the java file should match the class name. In this example, we have created two files in the same directory/folder

Main.java

Husk dette om Java Class Attributes:

Man kan ændre attributes som i python. Dog kan man låse en attribute ved at definere den med "final"-keyword



Husk dette om Java Class Methods:

Static vs. Non-Static

You will often see Java programs that have either static or public attributes and methods.

In Java, there are two types of casting:

Widening Casting (automatically) - converting a smaller type to a larger type size byte -> short -> char -> int -> long -> float -> double

Narrowing Casting (manually) - converting a larger type to a smaller size type double -> float -> long -> int -> char -> short -> byte

Widening Casting

Widening casting is done automatically when passing a smaller size type to a larger size type:

```
ublic class Main (
public static void main(string[] args) (
int myInt = 9;
double myDouble - myInt; // Automatic casting: int to double
 System.out.println(myInt); // Outputs 9
System.out.println(myDouble); // Outputs 9.0
```

Narrowing Casting

Narrowing casting must be done manually by placing the type in parentheses in front of the value:

```
Example
         ublic class Main (
public static void main(String[] args) (
double myCouble - 9.78d;
int myInt = (int) myCouble; // Manual casting; double to int
         System.out.println(myDouble); // Outputs 9.78
System.out.println(myInt); // Outputs 9
```

Husk dette om Java Operators:

Java Logical Operators

Operator	Name	Description	Example	Try it
8.8.	Logical and	Returns true if both statements are true	x < 5 && x < 10	Try it »
II	Logical or	Returns true if one of the statements is true	x < 5 x < 4	Try it >
1.	Logical not	Reverse the result, returns false if the result is true	I(x < 5 && x < 10)	Try it >

Husk dette om Java Strings:

Der findes String methods - ligesom i Python. Disse kan læses herinde:

 https://www.w3schools.com/java/java-strings.asp

Special Characters

Because strings must be written within quotes, Java will misunderstand this string, and generate an error:

```
String txt = "We are the so-called "Vikings" from the north."
```

The solution to avoid this problem, is to use the backslash escape character.

Escape character	Result	Description
V.	*	Single quote
7		Double quote
\\	\	Backslash

Other common escape sequences that are valid in Java are:

Code	Result	Try it
\n	New Line	Try it »
\r	Carriage Return	Try it »
\t	Tab	Try it »
\b	Backspace	Try it »
A	Form Feed	

Husk dette om Java Math:

Nogle matematiske ting, som skal bruges med Math.XXXX, som i python bliver bruges ved .xxxx For eksempel:

Math.max(x,y)

The Math.max(x,y) method can be used to find the highest value of x and y:

Example
Math.max(5, 10);

Husk dette om Java Booleans:

Husk dette om Java If...Else: • Forskel - Alt i if-statement skal stå i parentes ():

The if Statement

```
if (condition) {

// block of code to be executed if the condition is true
Note that if is in lowercase letters. Uppercase letters (If or IF) will generate an error.
In the example below, we test two values to find out if 20 is greater than 18. If the condition is true, print some text:
Example
```

• else if, i stedet for elif:

The else if Statement

if (20 > 18) (System.out.println("20 is greater than 18");

Use the else if statement to specify a new condition if the first condition is false.

```
Syntax
    if (confit(on)) (
    // Block of code to be executed (f condition) is true
) alsa if (condition)? {
    // Block of code to be executed if the condition] is folse and condition2 is true
) alsa (
      ) else {
// block of code to be executed if the condition1 is false and condition2 is false
```

Short Hand If...Else

```
Syntax
   variable = (condition) ? expressionTrue : expressionFalse;
Example
   int time = 20;
String result - (time < 18) ? "Good day." : "Good evening.";
System.out.println(result);</pre>
```

Husk dette om Java Switch

• Udfør kode til ét af mange forudbestemte scenarier:

Trifork side 2

HUSK WELLE OHI JAVA CIASS IVIELHOUS. - Ligesom du har lært tidligere om methods.

Static vs. Non-Static

You will often see Java programs that have either static or public attributes and methods.

```
An example to demonstrate the differences between static and public methods:
   public class Main {
// Static method
static void mystaticMethod() {
    System.out.println("Static methods can be called without creating objects");
    // Public method
public void myPublicMethod() {
   System.out.println("Public methods must be called by creating objects");
      // Main method
public static void main(String[] args) {
   myStaticMethod(); // Call the static method
   // myPublicMethod(); This would compile an error
         Main myObj = new Main(); // Create an object of Main myObj.myPublicMethod(); // Call the public method on the object
```

In the example above, we created a static method, which means that it can be accessed without creating an object of the class, unlike public, which can only be accessed by objects

Husk dette om Java Constructors:

- Hvis man vil angive variabler specifikt til objekt (parameters til class):

```
Example
     public class Main {
  int modelYear;
  String modelName;
       public Main(int year, String name) {
  modelYear = year;
  modelName = name;
       public static void main(String[] args) {
    Main myCar = new Main(1969, "Mustang");
    System.out.println(myCar.modelYear + " " + myCar.modelName);
```

Husk dette om Java Modifiers:

Access Modifiers

For classes, you can use either public or default:

	Modifier	Description	Try it
	public	The class is accessible by any other class	Try it »
	default	The class is only accessible by classes in the same package. This is used when you don't specify a modifier. You will learn more about packages in the <u>Packages chapter</u>	Try it »
For attributes, methods and constructors, you can use the one of the following:			

To distributed in Constitutions, yet can use the one of the following:		
Modifier	Description	Try it
public	The code is accessible for all classes	Tryitx
private	The code is only accessible within the declared class	Tryitx
default	The code is only accessible in the same package. This is used when you don't specify a modifier. You will learn more about packages in the <u>Packages chapter</u>	Try it x
protected	The code is accessible in the same package and subclasses . You will learn more about subclasses and superclasses in the Inheritance chapter	Try it s

Non-Access Modifiers

For classes, you can use either final or abstract:

Modifier	Description	Try it
final	The class cannot be inherited by other classes (You will learn more about inheritance in the <u>Inheritance chapter</u>)	Try it >
abstract	The class cannot be used to create objects (To access an abstract class, it must be inherited from another class. You will learn more about inheritance and abstraction in the <u>Inheritance</u> and <u>Abstraction</u> chapters)	Try it »

For **attributes and methods**, you can use the one of the following:

Modifier	Description
final	Attributes and methods cannot be overridden/modified
static	Attributes and methods belongs to the class, rather than an object
abstract	Can only be used in an abstract class, and can only be used on methods. The method does not have a body, for example abstract void run();. The body is provided by the subclass (inherited from). You will learn more about inheritance and abstraction in the Inheritance and Abstraction chapters
transient	Attributes and methods are skipped when serializing the object containing them
synchronized	Methods can only be accessed by one thread at a time
volatile	The value of an attribute is not cached thread-locally, and is always read from the "main memory"

```
Example
                     int day - 4;
switch (day) {
case 1:
    System.out.println("Monday");
    break;
    case 2:
    System.out.println("Tuesday");
    break;
    case 3:
    System.out.println("Nednesday");
    break;
    case 4:
    System.out.println("Thursday");
    break;
    case 5:
    System.out.println("Friday");
    break;
    case 6:
    System.out.println("Stunday");
    break;
    case 7:
    System.out.println("Stunday");
    break;
    case 7:
    System.out.println("Stunday");
    break;
    case 7:
    System.out.println("Sunday");
    break;
```

- Man kan også have "case default", til når ens switch-værdi ikke er en af de givne cases.

Husk dette om Java While Loop:

```
while (condition) {

// code block to be executed
```

Do-while loop:
 The do/while loop is a variant of the while loop. This loop will execute the code block once, before checking if the condition is true, then it will repeat the loop as long as the condition is true.

```
Syntax
  do {
// code block to be executed
  while (condition);
```

The example below uses a do/while loop. The loop will always be executed at least once, even if the condition is false, because the code block is executed before the condition is tested:

Husk dette om Java For Loop:

```
Syntax
    for (statement 1; statement 2; statement 3) (
  // code block to be executed
```

Statement 2 defines the condition for executing the code block.

Statement 3 is executed (every time) after the code block has been executed

The example below will print the numbers 0 to 4:

```
Example
     for (int i = 0; i < 5; i++) {
   System.out.println(i);</pre>
```

For-Each Loop

There is also a "for-each" loop, which is used exclusively to loop through elements in an array:

The following example outputs all elements in the cars array, using a "for-each" loop:

```
Example
```

Husk dette om Java Break/Continue:

Break er som i Python - Stopper alle iterationer i loop
 Continue stopper nuværende iteration i loop, men fortsætter til næste.

Husk dette om Arrays:

Array = List

Java Arrays

String[] cars;

Arrays are used to store multiple values in a single variable, instead of declaring separate variables for each value.

To declare an array, define the variable type with square brackets:

We have now declared a variable that holds an array of strings. To insert values to it, we can use an array literal - place the values in a comma-separated list, inside curly braces:

String[] cars = {"Volvo", "BMW", "Ford", "Mazda"};

To create an array of integers, you could write:

int[] myNum = {10, 20, 30, 40};

• Fungerer i høj grad som lists i Python - Samme syntaks til at få adgang til indeks i array, samt array i array. F.eks:

```
int[][] myNumbers = { {1, 2, 3, 4}, {5, 6, 7} };
```

Loop Through an Array

You can loop through the erray elements with the first loop, and use the length property to specify how many times the loop should run. The following exemple outputs all elements in the care erray:

Example string[] cars = ("volvo", "BMA", "Ford", "Mazda");
for (int i = 0; i < cars.length; i++) {
 System.out.println(cars[i]);</pre>

Loop Through an Array with For-Each There is also a "for-each" loop, which is used exclusively to loop through elements in arrays:

for (type variable : arrayname) (

The following example outputs all elements in the cars array, using a "for-each" loop:

string[] cars = {"volvo", "EMA", "Ford", "Mazda"};
for (String i : cars) {
 system.out.println(i);