Java Cheatsheet

==========

KLASSE & MAIN-METHODE

==========

Eine Klasse ist der grundlegende Baustein in Java. Die `main`-Methode ist der Startpunkt jedes Java-Programms.

```java
// Definiert den "Ordner" bzw. das Paket der Klasse
package packageName;

// Jede Datei enthält (meist) eine öffentliche Klasse
public class MeineKlasse {

    // Die main-Methode, hier startet die Ausführung
    public static void main(String[] args) {
        System.out.println("Hallo Welt!");
    }
}
```

==========

PRIMITIVE DATENTYPEN

==========

Dies sind die 8 grundlegenden Datentypen, die nicht auf Objekten basieren.

Typ	Größe	Beschreibung	Beispiel
`byte`	8-Bit	Sehr kleine Ganzzahlen	`byte b = 120;`
`short`	16-Bit	Kleine Ganzzahlen	`short s = 30000;`
`int`	32-Bit	Standard-Ganzzahlen	`int i = 2000000;`
`long`	64-Bit	Große Ganzzahlen	`long l = 9000000000L;`
`float`	32-Bit	Gleitkommazahlen	`float f = 3.14F;`
`double`	64-Bit	Standard-Gleitkommazahlen	`double d = 3.14159;`
`char`	16-Bit	Einzelnes Unicode-Zeichen	`char c = 'A';`
`boolean`	1-Bit	Wahrheitswert	`boolean aktiv = true;`

==========

STRINGS

==========

Ein `String` ist ein Objekt zur Darstellung von Zeichenketten.

```
String text = "Hallo Welt";

// Nützliche Methoden
int laenge = text.length();                 // Gibt die Länge zurück (10)
char zeichen = text.charAt(0);              // Holt das Zeichen am Index 0 ('H')
boolean leer = text.isEmpty();              // Prüft, ob der String leer ist
(false)
String gross = text.toUpperCase();          // Wandelt in Großbuchstaben um
("HALLO WELT")
boolean enthaelt = text.contains("Welt");   // Prüft, ob "Welt" enthalten ist
(true)
String teil = text.substring(6, 10);        // Extrahiert einen Teilstring
("Welt")
String[] teile = text.split(" ");           // Teilt den String am Leerzeichen
({"Hallo", "Welt"})

// Verketten
text = text + "!"; // "Hallo Welt!"
```

==========

ARRAYS

==========

Ein Array ist ein Container, der eine feste Anzahl von Werten eines einzigen Typs speichert.

```
// Deklaration und Initialisierung
int[] zahlen = {10, 20, 30, 40};
String[] namen = new String[5]; // Array für 5 Strings, anfangs mit null gefüllt

// Zugriff auf Elemente (Index beginnt bei 0)
int ersteZahl = zahlen[0]; // 10

// Element ändern
zahlen[1] = 25;

// Länge des Arrays
int anzahl = zahlen.length; // 4

// Mehrdimensionales Array (Matrix)
int[][] matrix = {
    {1, 2, 3},
```

```
    {4, 5, 6}
};
int wert = matrix[1][2]; // 6
```

==========

OPERATOREN

==========

Kategorie	Operatoren	Beschreibung
Arithmetisch	`+`, `-`, `*`, `/`, `%` (Modulo)	Grundrechenarten
Zuweisung	`=`, `+=`, `-=`, `*=`, `/=`, `%=`	Wertzuweisung / Kurzschreibweise
Inkrement/Dekrement	`++` (erhöhe um 1), `--` (verringere um 1)	`i++` (danach), `++i` (davor)
Vergleich	`==`, `!=`, `>`, `<`, `>=`, `<=`	Vergleicht zwei Werte, ergibt `boolean`
Logisch	`&&` (UND), `||` (ODER), `!` (NICHT)	Verknüpft `boolean`-Werte

```
int a = 10;
int b = 4;

// Arithmetisch
int summe = a + b; // 14
int rest = a % b;  // 2

// Zuweisung
a += 5; // a ist jetzt 15

// Vergleich
boolean istGleich = (a == b); // false

// Logisch (mit Kurzschluss-Auswertung)
boolean test = (a > 0) && (b > 0); // true
```

==========

KONTROLLSTRUKTUREN

==========

Steuern den Ablauf des Programms basierend auf Bedingungen.

IF-ELSE

```
int alter = 20;
if (alter >= 18) {
    System.out.println("Volljährig");
} else if (alter >= 16) {
    System.out.println("Jugendlich");
} else {
    System.out.println("Kind");
}
```

TERNÄRER OPERATOR

Eine kompakte Form für `if-else`.

```
String status = (alter >= 18) ? "Erwachsen" : "Minderjährig";
```

SWITCH

Für den Vergleich einer Variable mit mehreren möglichen Werten.

```
int note = 2;
String bewertung;
switch (note) {
    case 1:
        bewertung = "Sehr gut";
        break;
    case 2:
        bewertung = "Gut";
        break;
    case 3:
        bewertung = "Befriedigend";
        break;
    default:
        bewertung = "Nicht bestanden";
        break;
}
```

==========

SCHLEIFEN

==========

Wiederholen einen Codeblock, solange eine Bedingung erfüllt ist.

FOR-SCHLEIFE

Ideal, wenn die Anzahl der Wiederholungen bekannt ist.

```
for (int i = 0; i < 5; i++) {
    System.out.println("Durchlauf: " + i);
}
```

FOR-EACH-SCHLEIFE

Zum einfachen Durchlaufen von Arrays oder Collections.

```
String[] fruechte = {"Apfel", "Banane", "Kirsche"};
for (String frucht : fruechte) {
    System.out.println(frucht);
}
```

WHILE-SCHLEIFE

Prüft die Bedingung *vor* der Ausführung.

```
int counter = 0;
while (counter < 3) {
    System.out.println("Zähler: " + counter);
    counter++;
}
```

DO-WHILE-SCHLEIFE

Prüft die Bedingung *nach* der Ausführung (läuft mindestens einmal).

```
int i = 10;
do {
    System.out.println("Dieser Code läuft einmal.");
    i++;
} while (i < 5);
```

`break` & `continue`

- `break`: Beendet die Schleife sofort.
- `continue`: Überspringt den Rest des aktuellen Durchlaufs und startet den nächsten.

==========

METHODEN

==========

Methoden sind wiederverwendbare Codeblöcke.

```
// Methode ohne Rückgabewert und ohne Parameter
public static void gruesse() {
    System.out.println("Hallo!");
}

// Methode mit Rückgabewert und mit Parametern
public static int addiere(int a, int b) {
    return a + b;
}

// Methodenaufruf im Code
gruesse();
int ergebnis = addiere(5, 3); // ergebnis ist 8
```
