

Hello World-Programm <pre>// Klasse public class HelloWorld{ // main-Methode public static void main(String[] args){ // Ausgabe System.out.println("Hello World"); } }</pre>	Tastatur einlesen (Scanner) <pre>import java.util.Scanner; Scanner eingabe = new Scanner(System.in); String text = eingabe.next(); // bis Leerzeichen String zeile = eingabe.nextLine(); // ganze Zeile int iZahl = eingabe.nextInt(); double dZahl = eingabe.nextDouble();</pre> Hinweis: eingabe.close() schließt die Eingabe → danach keine weitere Eingabe möglich	Rechenoperatoren + Addition - Subtraktion * Multiplikation / Division % Modulo (Rest) Beispiel: <pre>int a = 7; int b = 3; a / b → 2 (int / int → Ganzzahl-Division!) 7.0 / 3 → 2.333 a % b → 1</pre> Inkrement / Dekrement: <pre>++, i-- ++ nach Verwendung erhöhen ++i vorher erhöhen</pre> Kurzschreibweise (nicht empfohlen): <pre>i = i + 2 → i += 2 i = i * 4 → i *= 4 i += 2, i -= 2 i *= 4, i /= 2</pre>	Logische Operatoren && UND ODER ! NICHT Beispiele: <pre>x > 5 && x < 10 → beide Bedingungen wahr x < 3 x > 10 → mindestens eine wahr !(x == 5) → kehrt true/false um</pre> Ergebnis: true oder false	switch (modern) Alternative zu vielen if-Abfragen Syntax: <pre>switch (variable) { case wert -> Anweisung; case wert -> Anweisung; default -> Anweisung; }</pre> Beispiel: <pre>String text = ""; int nrTag = 2; switch (nrTag) { case 1 -> text = "Montag"; case 2 -> text = "Dienstag"; case 3 -> text = "Mittwoch"; default -> text = "Unbekannt"; }</pre> Mehrere Werte: case 1, 2, 3 -> text = "Wochenanfang"; Vorteil: Kein break notwendig.	while-Schleife wiederholt Anweisungen, solange Bedingung true ist Syntax: <pre>while (Bedingung) { ... }</pre> Beispiel: <pre>int i = 0; while (i < 5) { System.out.println(i); i++; }</pre> Wichtig: Bedingung wird vor jedem Durchlauf geprüft kann 0-mal ausgeführt werden
Primitive Datentypen (Java) Ganzzahlen byte (8 Bit) -128 bis 127 short (16 Bit) -32.768 bis 32.767 int (32 Bit) -2.147.483.648 bis 2.147.483.647 long (64 Bit) sehr große Ganzzahlen int zahl = 5; Kommazahlen float (32 Bit) geringere Genauigkeit double (64 Bit) höhere Genauigkeit Beispiel: double pi = 3.14159; Zeichen char (16 Bit) einzelnes Zeichen Beispiel: char zeichen = 'A'; Wahrheitswert boolean (true / false) Beispiel: boolean fertig = true;	String (Zeichenkette) <pre>String text = "Hallo"; Länge: text.length() Zeichen: text.charAt(0) Teilstring: text.substring(0,3) Groß: text.toUpperCase() Klein: text.toLowerCase() Vergleich: text.equals("Hallo") Enthält: text.contains("all") Ersetzen: text.replace("a","o") Verbinden: text + " Welt" Leerzeichen entfernen: text.trim() text.strip() (modern - unicode fähig) stripLeading() links stripTrailing() rechts</pre>	Typumwandlung (Casting) Zahl → Zeichen: <pre>char z = (char) 65;</pre> Zeichen → Zahl: <pre>int zahl = (int) 'A';</pre> Weitere Beispiele: (double) 5 → 5.0 (int) 3.7 → 3 (Achtung!)	Namenskonventionen Klassen: Groß (CamelCase) HelloWorld, MeineKlasse Variablen / Methoden: klein (camelCase) zahl, meinName berechneSumme() Konstanten: GROSS_UND_MIT_UNDERSORE MAX_WERT, PI Wichtig: sprechende Namen! Keine Abkürzungen: anzahlSchueler statt a	switch (klassisch) <pre>switch (variable) { case wert: Anweisungen; break; case wert1: case wert2: case wert3: Anweisungen; break; default: Anweisungen; }</pre> Wichtig: break verhindert, dass der Code in den nächsten case weiterläuft. Ohne break werden auch die folgenden case-Blöcke ausgeführt (Fallthrough).	do-while-Schleife führt Anweisungen mind. 1x aus Bedingung wird am Ende geprüft Syntax: <pre>do { ... } while (Bedingung);</pre> Beispiel: <pre>int i = 0; do { System.out.println(i); i++; } while (i < 5);</pre> Wichtig: läuft mindestens einmal
Konstanten (final) Werte können nicht geändert werden. Syntax final Datentyp NAME = Wert; Beispiel final double PI = 3.14159;	Arrays Speichern mehrere Werte eines Datentyps. Erstellen: int[] zahlen = new int[5]; Mit Werten: int[] zahlen = {1, 2, 3, 4, 5}; Zugriff: zahlen[0] Ändern: zahlen[2] = 10; Länge: zahlen.length (ohne !) Ausgabe über Schleife: <pre>for (int i = 0; i < zahlen.length; i++) { System.out.println(zahlen[i]); }</pre> System.out.println(zahlen); → gibt NICHT die Werte aus, sondern eine Referenz (Speicheradresse)	for-Schleife wiederholt Anweisungen eine bestimmte Anzahl von Malen Syntax: <pre>for (Start; Bedingung; Änderung) { ... }</pre> Beispiel: <pre>for (int i = 0; i < 5; i++) { System.out.println(i); }</pre> über String: <pre>String text = "Hallo"; for (int i = 0; i < text.length(); i++) { System.out.println(text.charAt(i)); }</pre>	if-Verzweigung führt Anweisungen aus, wenn Bedingung true ist Syntax: <pre>if (Bedingung) { ... }</pre> <pre>if (Bedingung) { ... } else { ... }</pre> <pre>if (Bedingung1) { ... } else if (Bedingung2) { ... } else { ... }</pre> else if → weitere Bedingung prüfen Beispiel: <pre>int zahl = 10; if (zahl > 5) { System.out.println("größer als 5"); }</pre>	Methode (ohne Rückgabewert / Prozedur) Syntax modifier void name(Parameter) { Anweisungen } Beispiel: <pre>public static void verdopple(int zahl) { zahl = zahl * 2; System.out.println(zahl); }</pre> Aufruf: verdopple(5);	Methode (mit Rückgabewert / Funktion) Syntax modifier type name(Parameter) { return Wert; } Beispiel: <pre>public static int verdopple(int zahl) { return zahl * 2; }</pre> Aufruf: int ergebnis = verdopple(5); return beendet die Methode
Daten umwandeln (Parsing) String → int: int zahl = Integer.parseInt("123"); String → double: double zahl = Double.parseDouble("3.14"); String → boolean: boolean b = Boolean.parseBoolean("true"); int → String: String s = String.valueOf(zahl); String s2 = zahl + ""; Beispiel: String eingabe = "42"; int zahl = Integer.parseInt(eingabe);	Zufallszahlen (Math.random()) liefert eine Zufallszahl von 0.0 bis < 1.0 double zufall = Math.random(); <pre>int zahl = (int) (Math.random() * 10); // 0 bis 9 int wuerfel = (int) (Math.random() * 6) + 1; // 1 bis 6</pre>	Vergleichsoperatoren == gleich != ungleich < kleiner als > größer als <= kleiner oder gleich >= größer oder gleich Ergebnis: true oder false = Zuweisung (kein Vergleich!)	Parameterübergabe (Java): Übergabe erfolgt immer „call by value“ → Methode arbeitet mit Kopie der Variable	Bedingungen (boolean) true / false → Wahrheitswerte (x > 5) → true oder false boolean b = (x > 5); <pre>if (b) { ... } while (b) { ... }</pre> Bedingungen in if und Schleifen Bedingung = true oder false	Escape-Sequenzen Sonderzeichen in Strings \n neue Zeile \t Tabulator \" Anführungszeichen \\ Backslash
Pseudocode Algorithmus in einfacher, programmiernaher Sprache, keine feste Syntax Beispiel: lies Zahl wenn Zahl > 0 dann gib "positiv" aus	Klammern {} → Block → geschweifte Klammern () → Bedingungen / Methoden → runde Klammern [] → Arrays → eckige Klammern 2D-Array (Tabelle) Zugriff: [Zeile][Spalte] Erstellen: int[][] tabelle = new int[2][4]; → 2 Zeilen, 4 Spalten Beispiel: tabelle[0][1] = 5;	Kommentare // einzellig /* mehrzeilig */	Java API (Referenz) - Nachschlagewerk für Klassen & Methoden docs.oracle.com/en/java/javase/21/docs/api/	Programmablauf steuern break: beendet Schleife continue: überspringt aktuellen Durchlauf	