📓 Zusammenfassung: Klassen definieren – Datenfelder, Konstruktoren, Methoden

◇ 1. Datenfelder (Instanzvariablen)

Speichern die Daten eines `Objekts`

`→ definieren den Zustand.`

Jede Instanz einer Klasse hat ihre eigenen Datenfeldwerte.

Beispiel:

```
class Konto {
    double saldo;   // Datenfeld / Instanzvariable
}
```

◇ 2. Konstruktoren

- Werden beim Erzeugen eines `Objekts` aufgerufen.
- Setzen das `Objekt` in einen gültigen Anfangszustand.
- Haben keinen Rückgabewert, nicht einmal `void`.
- Tragen den Namen der Klasse.

Beispiel:

```
class Konto {
    double saldo;

    Konto(double startSaldo) {    // Konstruktor
        saldo = startSaldo;       // Initialisierung
    }
}

Konto k = new Konto(100.0);       // Objekt mit Startwert
```

◇ 3. Methoden

`Methoden` definieren das Verhalten eines `Objekts`.

Bestandteile:

- Kopf der `Methode` → **Signatur**
- Rumpf der `Methode` → **Anweisungen**

Arten:

Sondierende `Methode` → liefert Informationen `(z. B. getSaldo())`

Verändernde `Methode` → ändert Zustand `(z. B. einzahlen())`

Beispiel:

```
class Konto {
    double saldo;

    void einzahlen(double betrag) {     // verändernde Methode
        saldo += betrag;
    }

    double getSaldo() {                 // sondierende Methode
        return saldo;
    }
}
```

◇ 4. Parameter (formal vs. aktuell)

Formale `Parameter` = in der Methodendefinition

Aktuelle `Parameter` = beim Aufruf übergebene Werte

Beispiel:

```
void einzahlen(double betrag)   // betrag = formaler Parameter
k.einzahlen(50.0);              // 50.0 = aktueller Parameter
```

◇ 5. Variablen, Sichtbarkeit & Lebensdauer

Arten:

- `Instanzvariablen` → gelten für das ganze Objekt
- `Lokale Variablen` → existieren nur im Methodenrumpf

Sichtbarkeit:

Der Codebereich, in dem eine Variable erreichbar ist.

Lebensdauer:

- `Instanzvariable` → solange das Objekt existiert
- `Lokale Variable` → nur während der Methoden-Ausführung

Beispiel:

```
void teste() {
    int x = 5;  // lokale Variable, nur hier sichtbar
}
```

◇ 6. Anweisungen & Kontrollstrukturen

- Zuweisung

```
saldo = 200;
```

- Ausdruck & Operatoren

```
int sum = 5 + 7;  // Ausdruck mit Operator +
```

- Bedingte Anweisung

```
if (saldo > 0) {
System.out.println("Positiv");
} else {
System.out.println("Negativ");
}
```

- Boolescher Ausdruck

```
saldo > 0      // true oder false
```

7. Kommentare

- Erklären den Code für Menschen.
- Haben keine Auswirkung auf das Programm.

```
// Das ist ein Kommentar
```

Komplettes Lernbeispiel

```
class Konto {
private double saldo;   // Datenfeld

    // Konstruktor
    Konto(double startSaldo) {
        saldo = startSaldo;
    }

    // Verändernde Methode
    void einzahlen(double betrag) {
        saldo += betrag;
    }

    // Bedingte Anweisung + sondierende Methode
    double abheben(double betrag) {
        if (betrag <= saldo) {
            saldo -= betrag;
            return betrag;
        }
        return 0; // nicht genug Geld
    }

    // Sondierende Methode
    double getSaldo() {
        return saldo;
    }
}

class Main {
    public static void main(String[] args) {
    Konto k = new Konto(100);
    k.einzahlen(50);
    System.out.println(k.getSaldo()); // 150
    }
}
```

🧠 Merksätze

`Datenfelder` → definieren den Zustand eines `Objekts`.

`Konstruktoren` → initialisieren `Objekte`.

`Methoden` → bestimmen das Verhalten.

`Lokale Variablen` → existieren nur während der `Methode`.

Bedingte Anweisungen steuern den Ablauf.

Boolesche Ausdrücke entscheiden über `true/false`.