2.2 Klassen und Objekte

Definition: Klasse und Objekt

Eine **Klasse** ist der Bauplan: Sie legt fest, welche **Attribute** (Daten) und welche **Methoden** (Verhalten) die nach ihr gebauten Einheiten haben.

Ein **Objekt** (auch **Instanz**) ist eine nach diesem Bauplan erzeugte Einheit. Es belegt Speicher und trägt eigene Werte. Von einer Klasse kann es beliebig viele Objekte geben — gleich gebaut, verschieden gefüllt.

Der Vergleich, der trägt: Die Klasse ist das Formular, das Objekt das ausgefüllte Formular. Es gibt ein Formular und tausend Ausfüllungen.

Beispiel: Die Klasse Konto

Python

```python
class Konto:
    def __init__(self, inhaber):
        self.__inhaber = inhaber
        self.__stand = 0.0
```

Object Pascal

```
type
  TKonto = class
  private
    FInhaber: String;
    FStand: Real;
  public
    constructor Create(inhaber: String);
  end;
```

Zwei Konventionen, die Sie in fremdem Quelltext überall antreffen: In Object Pascal beginnen Klassennamen mit `T` (für *Type*) und Attributnamen mit `F` (für *Field*). Beides ist Vereinbarung, nicht Zwang — aber halten Sie sich daran, sobald jemand anderes Ihren Quelltext liest.

2.2.1 Der Konstruktor

Ein Objekt entsteht nicht dadurch, dass man eine Variable dafür deklariert. Es muss ausdrücklich erzeugt werden — durch den **Konstruktor**.

Definition: Konstruktor

Der **Konstruktor** ist die Methode, die ein neues Objekt erzeugt und seine Attribute mit Startwerten belegt. In Python heißt er `__init__`, in Object Pascal üblicherweise `Create`.

```
konto1 = Konto("A. Lovelace")
konto2 = Konto("C. Babbage")
```

```
constructor TKonto.Create(inhaber: String);
begin
  inherited Create;
  FInhaber := inhaber;
  FStand := 0.0;
end;

{ Aufruf: }
var konto1: TKonto;
begin
  konto1 := TKonto.Create('A. Lovelace');
  ...
  konto1.Free;        { Speicher wieder freigeben }
end;
```

Stolperstein

In Object Pascal ist eine Objektvariable ein **Zeiger**. Nach `var konto1: TKonto;` allein existiert noch kein Konto — `konto1` hat den Wert `nil`. Erst `TKonto.Create` legt das Objekt an. Und was erzeugt wurde, muss am Ende mit `Free` wieder freigegeben werden. Python räumt selbst auf, Pascal nicht.

Stolperstein

`konto2 := konto1;` kopiert **kein** Objekt. Danach zeigen beide Namen auf dasselbe Objekt, und das zweite ist unerreichbar geworden — sein Speicher bleibt belegt, ohne dass noch jemand hinkommt. Man nennt das ein **Speicherleck**.

Arbeitsauftrag 1

1. Erzeugen Sie drei Konten mit verschiedenen Inhabern und erklären Sie in einem Satz, was im Speicher geschieht.
2. Zeichnen Sie auf, was nach `konto2 := konto1;` im Speicher steht — vorher und nachher.