2.4 Datenkapselung und das Geheimnisprinzip

In beiden Fassungen stehen die Attribute geschützt: in Pascal hinter `private`, in Python hinter den beiden Unterstrichen von `__stand`.

Definition: Geheimnisprinzip

Nach dem **Geheimnisprinzip** verbirgt ein Objekt seine Daten vor der Außenwelt. Zugriff gibt es ausschließlich über Methoden, die das Objekt selbst anbietet. Man nennt das **Datenkapselung**.

Zwei Gründe, und der zweite ist der wichtigere:

Erstens die Prüfung. `konto.stand = -80` ist nicht möglich. Wer etwas am Stand ändern will, muss durch `einzahlen` oder `abheben`, und dort wird geprüft.

Zweitens die Freiheit, es später anders zu machen. Solange niemand von außen an `__stand` heranreicht, dürfen Sie das Attribut jederzeit umbauen — in Cent statt Euro rechnen, es aus einer Datei nachladen, es aus einer Liste von Buchungen berechnen. Kein fremder Quelltext bricht dabei. Hätten hundert Stellen direkt auf `__stand` zugegriffen, müssten Sie hundert Stellen ändern.

Merke

Ein direkter Zugriff auf ein Attribut von außen widerspricht der Kernidee des Message-Passing-Modells. Er verhindert jede spätere Umorganisation des Objektinneren.

Stolperstein

Pythons `__stand` ist kein echter Schutz — wer will, kommt mit `konto._Konto__stand` heran. Das ist Absicht: Python verlässt sich auf Übereinkunft statt auf Verbote. Die Übereinkunft gilt trotzdem. Was mit Unterstrichen beginnt, fasst man nicht an.

2.4.1 Lesen erlauben, Schreiben prüfen

Oft soll ein Attribut von außen *lesbar*, aber nicht frei *beschreibbar* sein. Object Pascal hat dafür die **Eigenschaft** (`property`):

```
type
  TKonto = class
  private
    FInhaber: String;
    FStand: Real;
    procedure SetzeInhaber(wert: String);
  public
    property Inhaber: String read FInhaber write SetzeInhaber;
    property Stand: Real read FStand;          { nur lesen }
  end;

procedure TKonto.SetzeInhaber(wert: String);
begin
  if wert <> '' then
    FInhaber := wert;
end;
```

Von außen sieht `Inhaber` aus wie ein ganz normales Attribut — tatsächlich läuft beim Schreiben die Prüfung in `SetzeInhaber`. `Stand` hat kein `write` und ist damit nur lesbar.

Python erreicht dasselbe mit `@property`:

```
1 class Konto:
2     @property
3     def stand(self):
4         return self.__stand
```

Arbeitsauftrag 2

3. Ergänzen Sie die Klasse `Konto` um eine Methode `ueberweisen(ziel, betrag)`, die Geld auf ein anderes Konto überweist. Benutzen Sie dabei ausschließlich vorhandene Methoden — greifen Sie **nicht** auf `__stand` des Zielkontos zu.
4. Was soll `ueberweisen` zurückgeben? Begründen Sie Ihre Entscheidung mit den Begriffen Auftrag und Anfrage.
5. Begründen Sie an einem eigenen Beispiel, warum ein Attribut `alter` in einer Klasse `Person` besser nicht öffentlich beschreibbar ist.

Gymnasium Ottweiler