

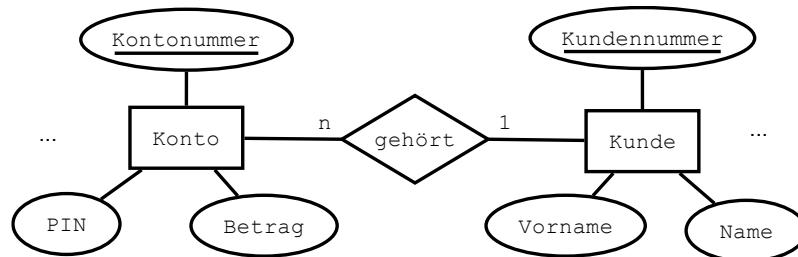


## Arbeitsstand

Das Bank-Projekt soll Transaktionen zwischen Konten von Kunden nach den Geschäftsregeln der Wossi-Bank simulieren. In Analogie zur Datenbankentwicklung wurden bislang die Klassen **Kunde** und **Konto** entworfen.

### Analogiebetrachtung Datenbankentwicklung:

ER-Modell:



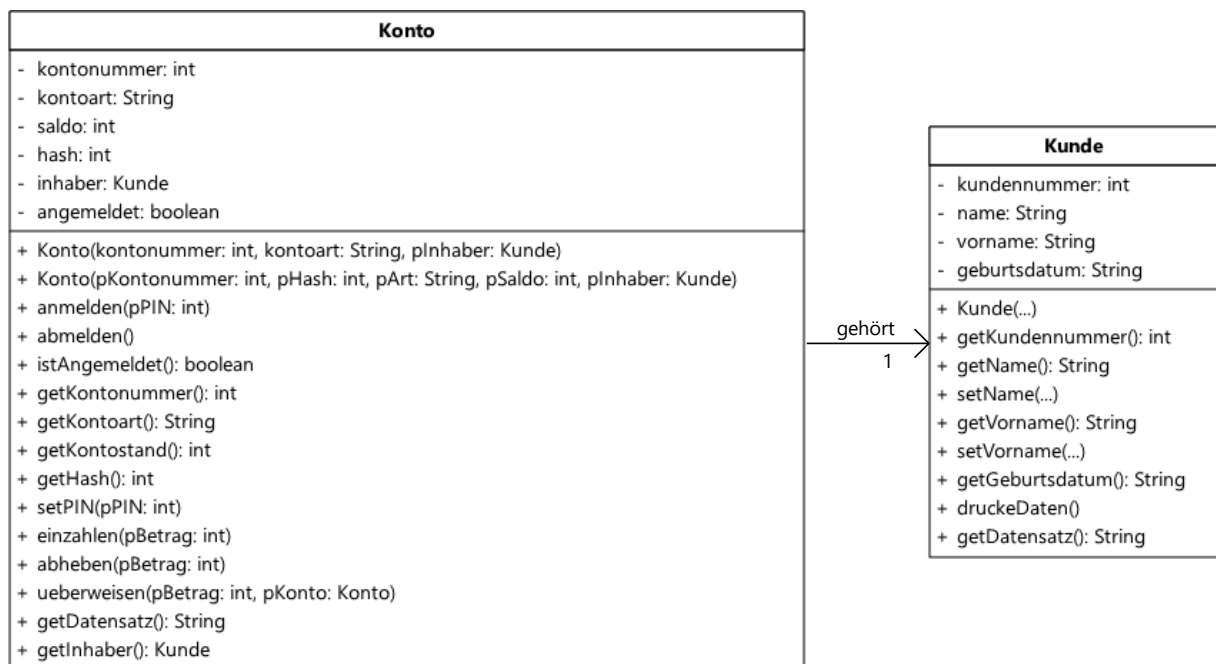
Relationenschema:

- Kunde (Kundennummer, Name, Vorname, ...)
- Konto (Kontonummer, Betrag, PIN, ..., ↑Kunde.Kundennummer)

## Softwareentwicklung

Die Klasse **Kunde** ist eine Vorlage für Kunden der Bank mit ihren Attributen und mit Methoden zum Abfragen und Ändern von Attributwerten.

Die Klasse **Konto** ist eine Vorlage für Konten der Bank, die bislang durch die Attribute `nummer`, `saldo`, `pin` und `art` beschrieben wird. **Konto** verfügt über Methoden zum Abheben, Einzahlen und Überweisen von Geldbeträgen. Die `get`-Methoden ermöglichen die Abfrage des jeweiligen Attributes. Der Konstruktor besitzt Parameter, über die die Eigenschaften beim Erzeugen eines Objekts definiert werden.





## Klasse Kunde

- 1) Prüfen und korrigieren Sie Ihre Klasse hinsichtlich
  - a) Name und Sichtbarkeit der Attribute und Methoden entsprechend der Klassenkarte
  - b) Parameterreihenfolge im Konstruktor: Kundennr., Name, Vorname, Geburtsdatum
  - c) Dokumentation von Klasse, Konstruktor und Methoden mit @-Angaben
- 2) Vervollständigen Sie die Klasse um die Methoden
  - a) `getDatensatz()` – gibt Daten als *kundennummer; name; vorname; geburtsdatum* zurück
  - b) `druckeDaten()` – gibt Daten auf der Konsole aus
- 3) Testen Sie die Funktionalität. Bei Fehlerfreiheit ist die Entwicklung der Klasse beendet.

## Klasse Konto

- 4) Prüfen und korrigieren Sie Ihre Klasse hinsichtlich
  - a) Name und Sichtbarkeit der Attribute und Methoden entsprechend der Klassenkarte
  - b) Parameterreihenfolge im Konstruktor entsprechend der Klassenkarte
  - c) Dokumentation von Klasse, Konstruktor und Methoden mit @-Angaben
- 5) Prüfen und korrigieren Sie die Beziehung zwischen Kunde und Konto mithilfe des Attributs `inhaber` vom Typ `Kunde`. Implementieren Sie eine `get`-Methode für das Attribut `inhaber`. Informieren Sie sich im Buch „Informatik 1“ ab S. 100 über Assoziation und Multiplizität. Vergleichen Sie mit dem Beziehungsbegriff im ER-Modell.
- 6) Prüfen und korrigieren Sie die Methoden `setPin`, `getKontostand`, `abheben` und `ueberweisen`. Sie sollen nur dann ausführbar sein, wenn sich der Kunde mit seiner PIN über die Methode anmelden registriert hat und das Attribut angemeldet gesetzt wurde. Eine Methode zum Abmelden und zur Rückgabe des Anmeldestatus ist zu implementieren.
- 7) Prüfen und korrigieren Sie alle Methoden der Klasse `Konto` auch mit „kritischen“ Werten.
- 8) Die Speicherung der PIN im Klartext ist gefährlich. Informieren Sie sich über das Konzept der Hash-Funktion unter <https://t1p.de/n5gi>. Ändern Sie Klasse `Konto` so ab, dass
  - a) statt der PIN der PIN-Hash-Wert im Objekt liegt und dieser abgefragt werden kann.
  - b) beim Anlegen eines Kontos mithilfe des vorhandenen Konstruktors die PIN zufällig generiert und dem Kunden mitgeteilt wird. Dann ist der Hash-Wert zu speichern.
  - c) die PIN geändert werden kann und der Hash-Wert gespeichert wird.
  - d) alle PIN-Prüfungen via Hash-Wert-Vergleich funktionieren.
  - e) ein zweiter Konstruktor mit Kontonummer, Hash, Kontoart, Kontostand und Inhaber existiert, um später die Daten darüber wiederherstellen zu können.
- 9) Die Methode `getDatensatz` soll eine durch Semikolon strukturierte Zeichenkette bestehend aus Kontonummer, Hash, Kontoart, Saldo und Inhabernummer zurückgeben.

### Hinweise:

- Die Funktion `hashCode()` erzeugt aus einer Zeichenkette eine eindeutige Zahl. Die Rekonstruktion der Zeichenkette ist kaum möglich.
- Da die PIN vom Typ `GANZZAHL` eingegeben wird, muss diese zunächst in eine Zeichenkette umgewandelt werden (siehe Hilfe zur Typumwandlung). Anschließend kann der Hashcode der Zeichenkette bestimmt und gespeichert werden.