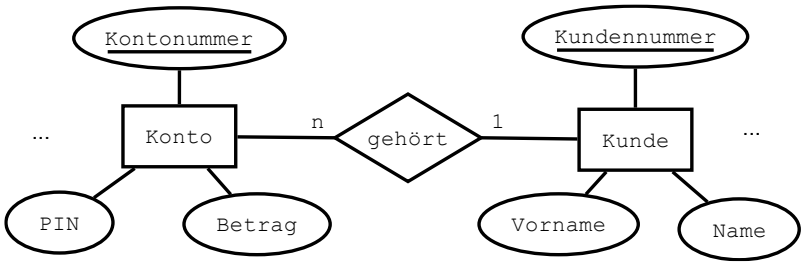
Arbeitsstand

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:



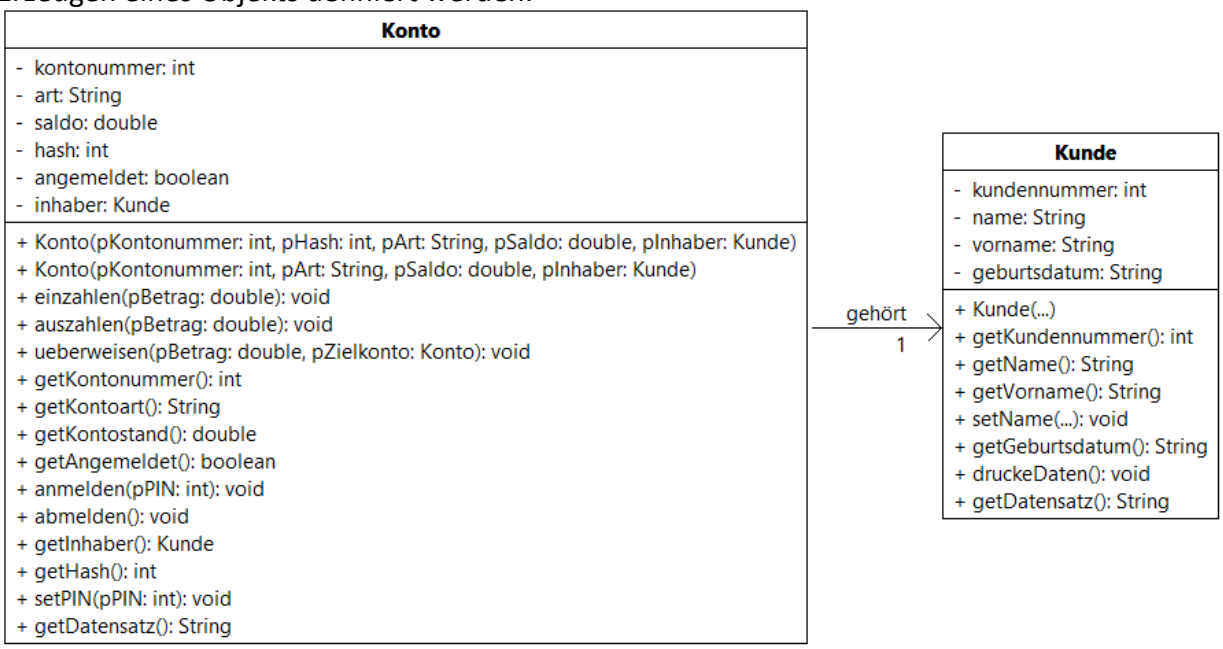
Relationenschema:

- Kunde (Kundenummer, 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 ermöglicht die Abfrage des jeweiligen Attributes. Der Konstruktor besitzt Parameter, über die die Eigenschaften beim Erzeugen eines Objekts definiert werden.



Projekt Bankverwaltung 2024/25

Name: **Vorname:** **Klasse:**

Klasse Kunde

1) Prüfen/Korrigieren Sie Ihre Klasse **Kunde.** Alle Bezeichnungen müssen der Klassenkarte entsprechen. Prüfen/Korrigieren Sie die Reihenfolge der Parameter des Konstruktors: Kundennummer, Name, Vorname, Geburtsdatum. Die Eigenschaften und Methoden sind passend zu kapseln.
2) Erzeugen Sie für alle Methoden und die Klasse vollständige JavaDoc-Kommentare. Achten Sie auf die Pflichtangaben `@param`, `@return`, `@author` und `@version`.
3) Testen die Funktionalität der Klasse. Die Methode `druckeDatensatz` soll eine durch Semikolon strukturierte Zeichenkette bestehend aus Kundennummer, Name, Vorname und Geburtsdatum zurückgeben.
4) Sollte alles fehlerfrei funktionieren, ist die Entwicklung dieser Klasse abgeschlossen.

Klasse `Konto`

5) Vervollständigen/Ändern Sie die Klasse **`Konto`** gemäß Klassendiagramm. Achten Sie auf Gleichheit der Bezeichnungen und die Reihenfolge der Parameter, ändern Sie ggf. Methoden/Parameter/Attributbezeichner. Die Eigenschaften und Methoden sind passend zu kapseln.
6) Erzeugen Sie für **alle** Methoden und die Klasse vollständige JavaDoc-Kommentare. Achten Sie auf die Pflichtangaben `@param`, `@return`, `@author` und `@version`.
7) Zur Abbildung der Beziehung zwischen **Kunde** und **`Konto`** fügen Sie – in Analogie zum Relationenschema – in die Klasse **`Konto`** das Attribut `inhaber` vom Typ **Kunde** ein. Implementieren Sie außerdem eine zugehörige get-Methode für das Attribut `inhaber`. Informieren Sie sich im Lehrbuch „Informatik 1“, S. 100ff über die Begriffe Assoziation und Multiplizität.
   Vergleichen Sie mit dem Beziehungsbegriff im ER-Modell. Beschreiben Sie die Realisierung der Beziehung im Projekt.
8) Die Methoden `setPin`, `getKontostand`, `auszahlen` und `ueberweisen` sollen nur dann ausführbar sein, wenn sich der Kunde mit seiner PIN über die Methode `anmelden` registriert hat. Die Klasse verfügt über je eine Methode zum An- und Abmelden, der Status ist in der Eigenschaft `angemeldet` zu speichern und eine Methode zur Abfrage der Eigenschaft muss existieren.
9) Testen Sie alle Methoden der Klasse **`Konto`** auch mit „kritischen“ Werten. Passen Sie die Klasse ggf. an.
10) Die Methode `druckeDatensatz` soll eine durch Semikolon strukturierte Zeichenkette bestehend aus Kontonummer, Hash, Kontoart, Saldo und Inhabernummer zurückgeben.

11) Die Speicherung einer PIN im Klartext ist aus Sicherheitsgründen kritisch. Informieren Sie sich im Online-Lehrbuch INF-Schule unter https://t1p.de/n5gi über das Konzept der Hash-Funktion und beschreiben Sie dieses Prinzip mit eigenen Worten.

12) Ändern Sie Klasse **`Konto`** so ab, dass

   a. statt der PIN der über die Funktion `hashCode()` ermittelte Hash-Wert der PIN im Objekt liegt,

   b. ein neues Konto mit den Eigenschaften Kontonummer, Hash, Kontoart, Kontostand und Inhaber erzeugt werden kann,

   c. die PIN beim Anlegen eines neuen Kontos (Konstruktor Nr. 2) automatisch zufällig generiert, dem Kunden mitgeteilt und im Objekt der zugehörige Hash-Wert gespeichert wird,

   d. die PIN nach Anmeldung am Konto durch den Kunden geändert werden kann und nur als Hash-Wert im Objekt gespeichert wird,

   e. alle PIN-Prüfungen korrekt via Hash-Wert-Vergleich funktionieren,

   f. der Hash-Wert auch abgefragt werden kann.

Hinweise:

- Die von uns verwendete 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.