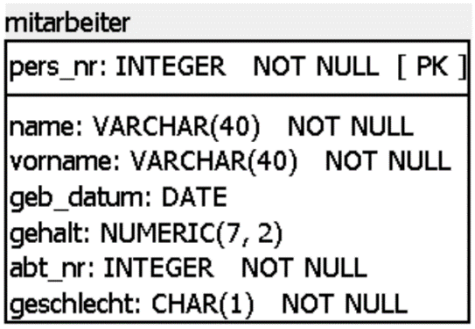
Beispiel MitarbeiterDB:

Erstelle eine JDBC-Applikation um Mitarbeiterdaten in einer Datenbank zu speichern.

Erzeuge programmatisch die postgres-Datenbank **`mitarbeiterdb`** mit der Tabelle **`mitarbeiter`**, die folgende Eigenschaften besitzt:

- Die Struktur der Tabelle ist entsprechend dem ERD abzubilden
- Der PK wird von der Java-Applikation vergeben
- Es dürfen keine Mitarbeiter (vorname + name + geb_datum) doppelt in die Datenbank eingefügt werden
- Verwende den pgAdmin um die Daten in die Tabelle einzuspielen

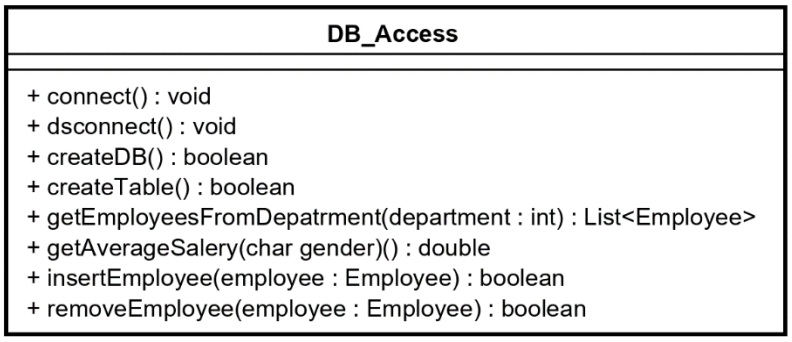


Folgende User-stories sind zu implementieren:

- ☑ ***As a user I want to connect to the database***: Aufbau einer Verbindung zur Datenbank **`mitarbeiterdb`**
- ☑ ***As a user I want to close the connection***: Schließen der Verbindung zur DB
- ☑ ***As a user I want to create database and table***: Erzeugen der Datenbank **`mitarbeiterdb`** und der Tabelle **`mitarbeiter`**. Wenn die DB und/oder die Tabelle bereits existieren werden sie neu erzeugt (die alten Daten werden dabei gelöscht). Verwende den **pgAdmin** und das SQL-Skript **`mitarbeiter_sql_inserts.sql`** um Daten in die Tabelle einzufügen
- ☑ ***As a user I want to view all employees of one department sorted by name***: Nach Eingabe einer Abteilungsnummer werden alle Mitarbeiter dieser Abteilung, sortiert nach Nachnamen und Vornamen angezeigt.
- ☑ ***As a user I want to know the average salary of all male or female employees***: Nach Eingabe eins Geschlechts (M/F) wird das Durchschnittsgehalt aller dieser Mitarbeiter angezeigt.
- ☑ ***As a user I want to insert new employees into the database***: Einfügen von neuen Mitarbeitern in die DB. Erstelle dazu eine .csv-Datei im package **`res`**, die Daten von zumindest 3 Mitarbeitern enthält. Diese Datei wird eingelesen. Der Primärschlüssel ist von der Java-Applikation zu vergeben. Es dürfen allerdings keine Mitarbeiter doppelt in die Datenbank eingefügt werden.
- ☑ ***As a user I want to remove an employees from the database***: Löschen eines Mitarbeiters aus der DB. Der Mitarbeiter wird durch seine Personalnummer identifiziert.

Die Applikation ist auf Basis eines 3-Schichten-Modells (BL-DB-UI) zu erstellen:

- □ **`Employee`**
  Java-Beans-Klasse, die alle Daten eines Mitarbeiters enthält. Die Namen der Instanzvariablen sind identisch zum ERD. Verwende sinnvolle Java-Datentypen (z.B. **`LocalDate`** für das Geburtsdatum).
- □ **`DB_Access`**
  Java-Klasse für den Datenbank-Zugriff entsprechend dem KD
- □ **`EmployeesUI/EmployeesGUI`**
  UI oder GUI-Klasse mit der alle User-stories durchgeführt werden können. Neben den Daten aus der DB sollen auch weitere Informationen, z.B. wie viele Mitarbeiter eingefügt oder gelöscht wurden, angezeigt werden.



Es dürfen keine unbehandelten Exceptions auftreten!

Lernstoff:

- □ JDBC mit postgresql
- □ Java NIO

Umfang: 1-2 DP-Std