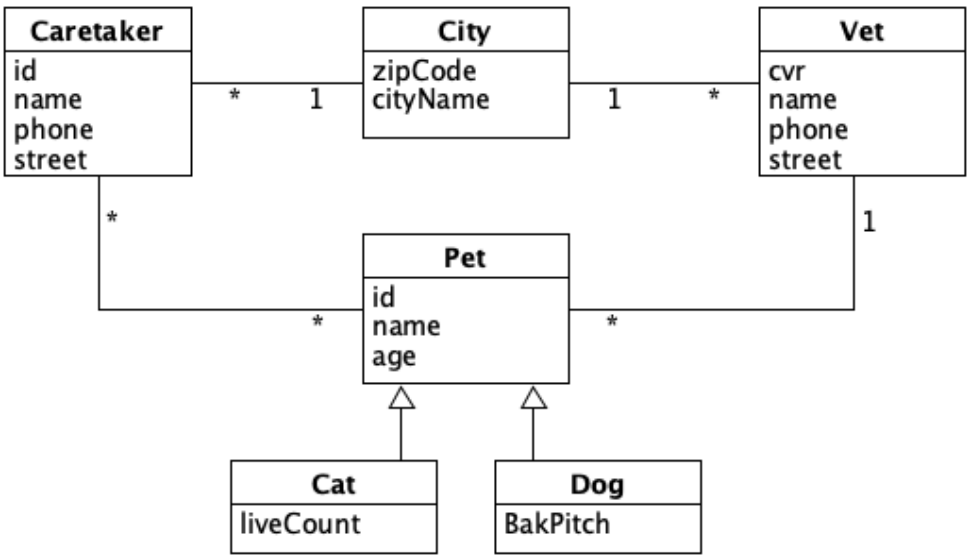
Assignment #1 - Relational Databases
Databases for Developers

Anders Kalhauge, Martin Vestergaard, and Dora Dimitrova

Spring 2021

Relational Databases

In groups, consider the following domain model.



Design

Create an ER diagram covering the domain. You should choose a strategy to implement the inheritance, and argue with pros and cons for each strategy.

- Joint-table strategy
- Table-per-class strategy
- Single-table strategy

Conceptual level implementation

Create an SQL-script for a PostgreSQL™database that creates the tables accordingly. Bevare that the script should be reentrant[1]

Create an SQL-script with sample data for your tables. You should have **at least** two vetenarians, twenty pets of various kinds including some that are neither cats nor dogs, and ten caretakers some with common pets. Also this script should be reentrant.

External level implementation

Create views and/or stored procedures to deal with the chosen inheritance strategy. It should be possible to:

- See cats and dogs as separate views
- See all pets as in the single table strategy
- Update cats and dogs with a single SQL call, stored procedure or update on a view with a trigger.

Create a script that creates a *designated user* for accessing the database and revokes the rights for that user to access the underlaying tables, implementing the inheritance strategy.

[1] can be executed several times with the same end-result

Interface implementation

Create a simple program in Java or similar object oriented language that is able to:

- retrieve a list of pets from the database. The types of instances of `Pet`s in the list should reflect the actual type:
  - `Pet`
  - `Cat`
  - `Dog`
- insert a new `Dog`, `Cat`, and/or `Pet` in the database.

The program should use the *designated user*

Example of a small Java application accesing a PostgreSQL™database:

```
public static void main(String[] args) throws Exception {
  String url = "jdbc:postgresql://localhost/soft2021";
  Properties props = new Properties();
  props.setProperty("user","softdbd");
  props.setProperty("password","softdbd");
  try (Connection conn = DriverManager.getConnection(url, props)) {
    String sql = "SELECT * FROM EXAMPLE;";
    PreparedStatement statement = conn.prepareStatement(sql);
    try(ResultSet result = statement.executeQuery()) {
      while (result.next()) {
        System.out.println(""+result.getInt(1)+" "+result.getString(2));
        }
      }
    }
  }
```

See `https://www.javatpoint.com/CallableStatement-interface` for examples of calling a stored procedure from Java.

Hand in

A link to the github repository with all scripts and a README.dm file explaining the code and the choices. In groups on Peergrade Thursday March 11th no later than 08:30.