Application Development Frameworks

Assignment 1 Document

Gearóid Sheehan (R00151523)

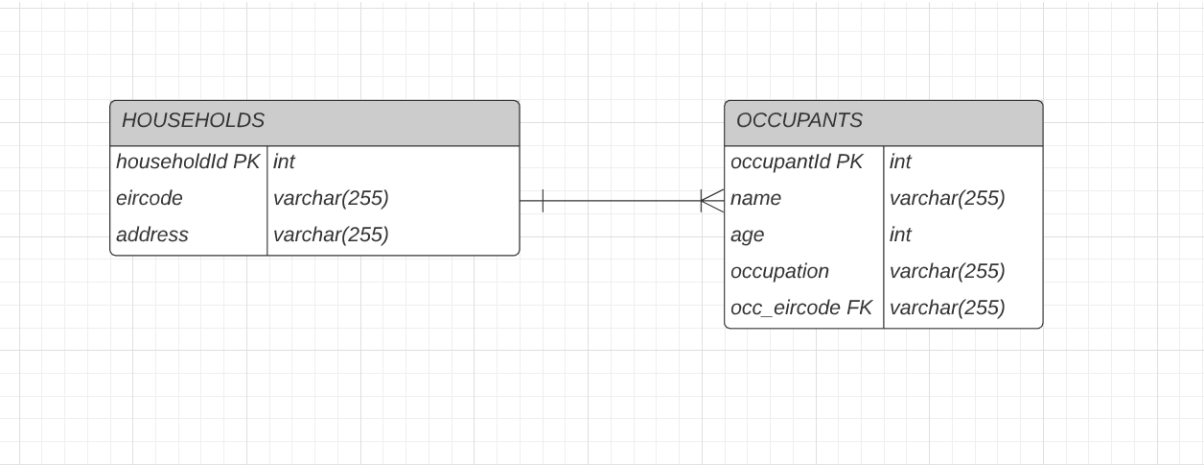
For this assignment the Data Access Object Pattern is used for the architecture, as is required in the brief. As described by Baeldung.com, The Data Access Object (DAO) pattern is a structural pattern that allows us to isolate the application/business layer from the persistence layer (usually a relational database).

My application consists of a main method, a singular service layer and its interface, and two data access object layers with each of their interfaces. The reason I chose to have two as the number of data access object layers is because there are two tables in my database; the occupant table and the household table. Each of these tables is also represented in the code as the occupant and household classes. I only use the one service layer as the application is not large enough to warrant a service layer for each of the data access object layers. There is also a large crossover for each of the menu tasks where it suited better to retrieve the data from both of the data access object layers from the same service.

As required by the Data Access Object pattern, my data access object layers consist of the logic needed only run SQL queries and return the required data from the database, and nothing else. The service layer retrieves this data and takes care of the business logic, such as error checking and counting. I return the data as strings where possible to the main method from the business layer so that the main method will only have to print the data to the console and nothing else.

The main method also includes error handling. This is to prevent non-numeric values being passed into the program where numeric values are required. This is done using try/catch exception handling and while loops.

My database is a simple design, with only two tables. The households table has a primary key of householdId, while the occupants table has a primary key of occupantId and a foreign key of occ_eircode, which references eircode on the households table.



```
HCREATE TABLE households (
    householdId int NOT NULL AUTO_INCREMENT,
    eircode varchar(255) NOT NULL,
    address varchar(255) NOT NULL,
    PRIMARY KEY (householdId)
);

CREATE TABLE occupants (
  occupantId int NOT NULL AUTO_INCREMENT,
  name varchar(255) NOT NULL,
  age int NOT NULL,
  occupation varchar(255) NOT NULL,
  occ_eircode varchar(255),
  PRIMARY KEY (occupantId),
  FOREIGN KEY (occ_eircode) REFERENCES households(eircode) ON DELETE CASCADE
);
```

I managed to finish the project in time and completed all the required specifications. I also took care of all events where error handling was needed, so the application is structurally sound from input which would otherwise cause error. Careful consideration has been taken to make sure that all events have been covered, for example if all occupants are deleted from a household then the household at that Eircode is also deleted as there cannot be empty households on the system. All Eircode's entered are also changed to uppercase if they are not originally input as so. My Junit tests also work as desired. They test the average age, delete occupant, add occupant and count old age pensioners queries.