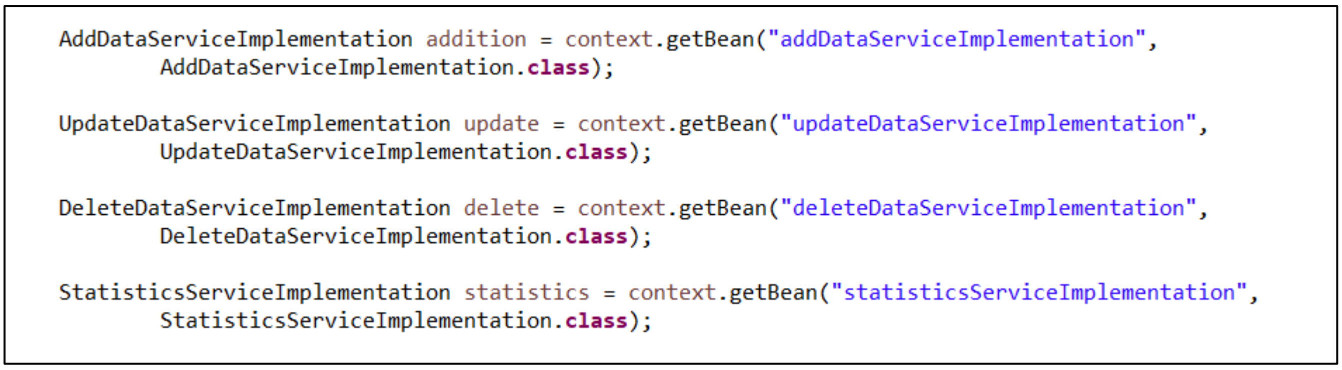
Design Document for Project 1

Student name: Mantvydas Luksas

Student ID: R00150390

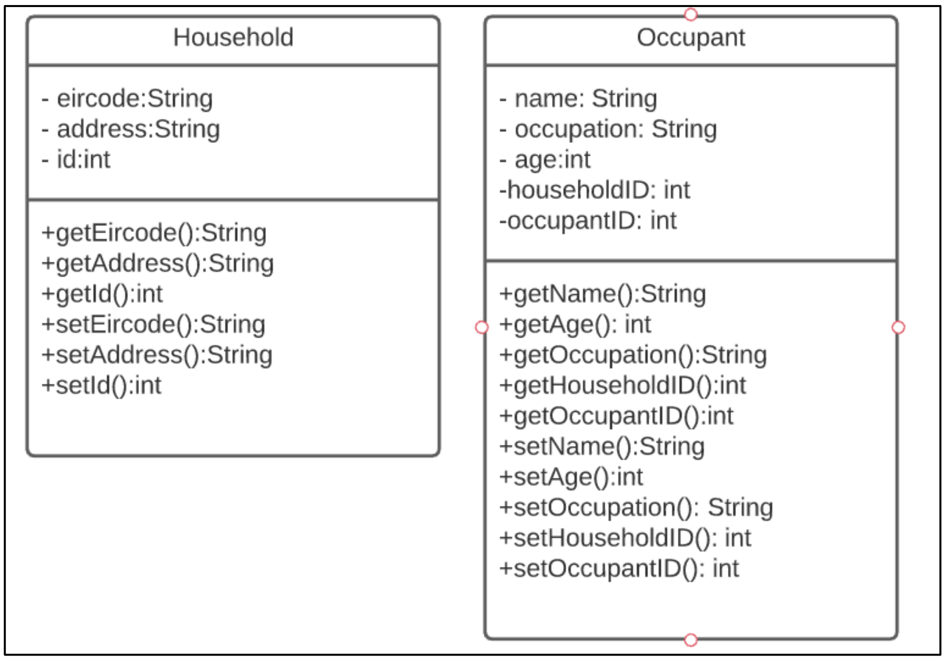
The system is primarily designed by first establishing the necessary beans required for the project. The beans have been implemented in an XML format under beans.xml. No profiles in XML were used, as beans were selected by different service implementations. These service beans were easily found, as the beans file searched the base package ie.luksas.

Tasks such as finding a household, inserting a new household/person, updating, and deleting were implemented by an individual service. Of course, their functionality would not be possible without access to the database. It was essential to use the JDBC Template bean from the beans.xml file which was accessed through inversion of control by autowiring. The different services could be accessed, as beans in the main class by calling the service.

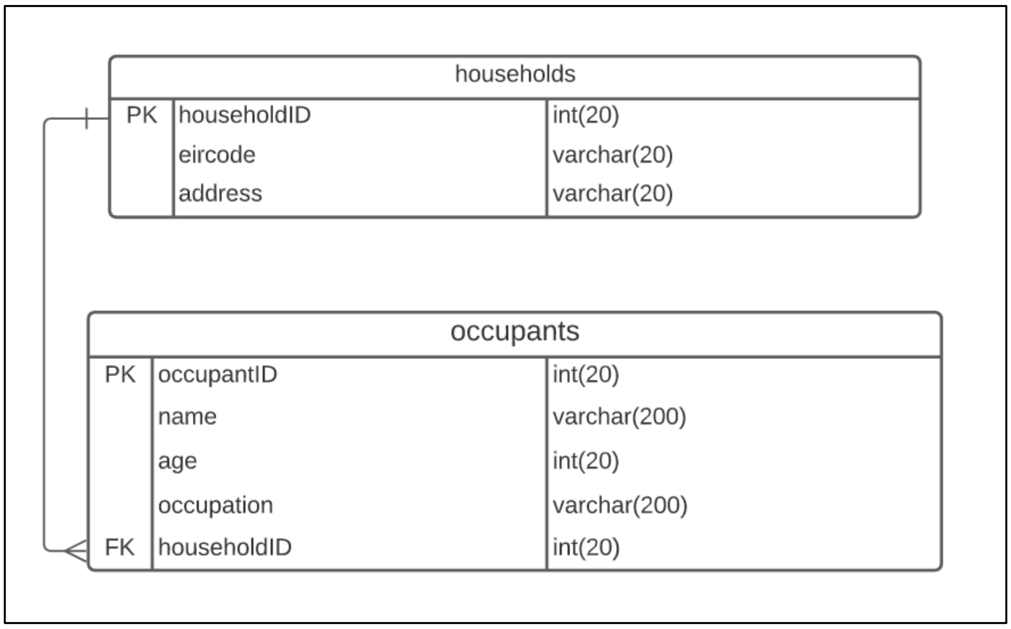


```
AddDataServiceImplementation addition = context.getBean("addDataServiceImplementation",
        AddDataServiceImplementation.class);

UpdateDataServiceImplementation update = context.getBean("updateDataServiceImplementation",
        UpdateDataServiceImplementation.class);

DeleteDataServiceImplementation delete = context.getBean("deleteDataServiceImplementation",
        DeleteDataServiceImplementation.class);

StatisticsServiceImplementation statistics = context.getBean("statisticsServiceImplementation",
        StatisticsServiceImplementation.class);
```

As per the requirements listed, I had decided that I will need two classes to store information about households and its occupants.



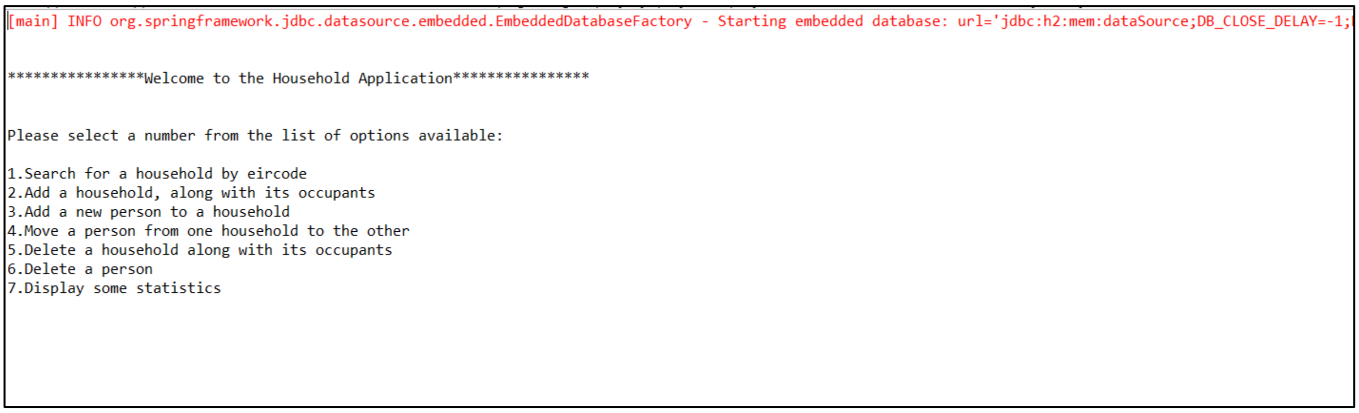
These classes were obviously linked to the database to their own relational tables.



As seen from the entity relationship diagram above, each household that was entered into the database had to be unique. To make sure households were unique, they all were required to have a unique Eircode. However, to link each occupant of a particular household, the household id had to act as a foreign key in the occupants table. Many occupants had the same household id as indicated by the multiplicity relationship on the diagram.

In Java it was determined, if the occupant is either a “scholar” or in “pre-school” by the age entered by the user. If the age of the occupant was below 7, the occupant was labelled to have the “pre-school” occupation. Likewise, if the age of the occupant was above 7 but below 19 years old, then the occupant was labelled as “scholar”. If the occupant had listed any other type of age, they were asked to provide their occupation in the application.

The application runs in a menu user interface through the console of the Spring Eclipse IDE as seen below.



```
[main] INFO org.springframework.jdbc.datasource.embedded.EmbeddedDatabaseFactory - Starting embedded database: url='jdbc:h2:mem:dataSource;DB_CLOSE_DELAY=-1;


***************Welcome to the Household Application***************


Please select a number from the list of options available:

1.Search for a household by eircode
2.Add a household, along with its occupants
3.Add a new person to a household
4.Move a person from one household to the other
5.Delete a household along with its occupants
6.Delete a person
7.Display some statistics
```

Any errors that the user makes in selecting a specific operation or indicating information is handled and the user can try again.