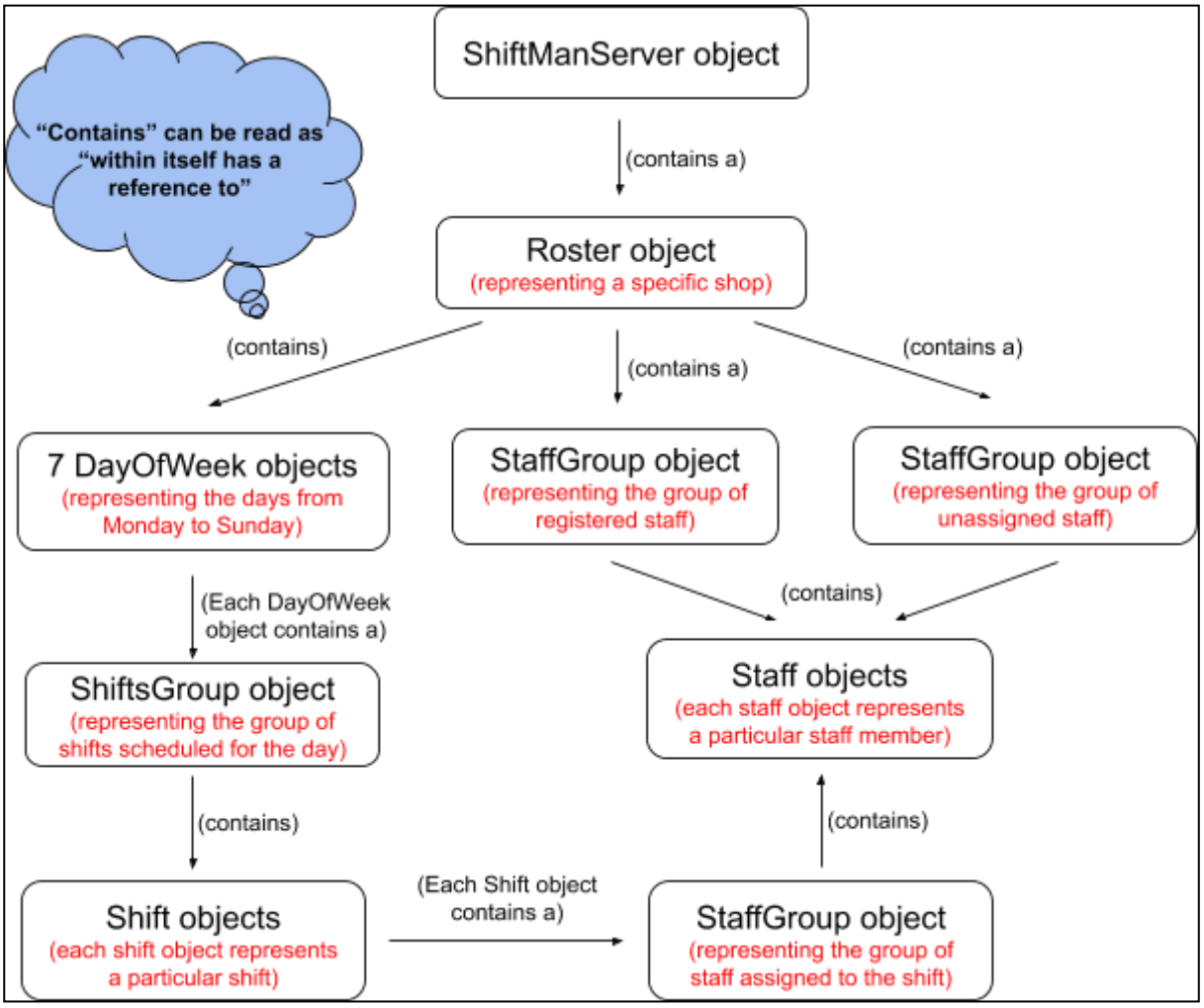
Assignment 2 Report

Domain Model and its representation in the design

No.	Main Concepts	Why Is It A Main Concept	Representation in the design
1.	Staff member	The entire problem domain partly revolves around managing a number of staff members and each staff member has unique details associated with it such as the staff name.	Staff class
2.	Shift	The entire problem domain partly revolves around managing a number of shifts and each shift has unique details associated with it such as start and end time of the shift and the workers assigned to that shift.	Shift class
3.	Group of staff members	The problem domain has ideas of ‘registered staff’, ‘unassigned staff’ and ‘staff assigned to a shift’, all of which are fundamentally the same as the idea of a group of staff members.	StaffGroup class
4.	Group of shifts	The problem domain has ideas of ‘overstaffed shifts’, ‘understaffed shifts’ and ‘shifts managed by a particular manager’, all of which are fundamentally the same as the idea of a group of shifts.	ShiftsGroup class
5.	Day of the week	The problem domain of managing shift work staff can be looked at as a daywise operation, as each day has particular working hours and has a certain number of shifts scheduled, assigned to which are a number of staff members.	DayOfWeek class
6.	Shop	All the above mentioned concepts can be associated with/contained in the idea of a shop. Also, the problem domain has the idea of a shop being managed at a given time.	Roster class

7.	Server	It is convenient for the functionalities mentioned in the problem domain, such as being able to retrieve the registered staff etc., to be able to be accessed from a single location. So all the functionalities can be grouped under the idea of a server.	ShiftManServer class

Design



Justifications Of Some Design Decisions

1) **Roster class** was created for it to represent the roster of a particular shop. By doing this, the group of registered staff and the group of unassigned staff were able to be stored in one location for the particular shop. Also, these two groups of staff are applicable to the shop as a whole as there will be only one set of registered and only one set of unassigned staff for the entire shop.

2) **DayOfWeek class** was created because firstly, managing shift work staff can be seen as a day-wise operation. That is, the same set of procedures are carried out on all the days the shop is open. So, it seems to be a good choice to divide the problem being solved into 7 days. Secondly, the process of retrieving information (such as the shifts managed by a particular manager or the roster for a particular worker) which is sorted by the days of the week becomes trivial because the required data will already be organised day-wise due to the presence of 7 DayOfWeek objects.

3) **ShiftsGroup class** was created because then the implementation of responsibilities related to a group of shifts could be grouped together under one class. Examples of such responsibilities are: retrieving the group of understaffed shifts & overstaffed shifts, retrieving shifts managed by a particular manager and getting an appropriate string representation of the group of staff working on a particular day.

4) **StaffGroup class** was created because then the implementation of responsibilities related to a group of staff could be grouped together under one class. Example: getting a string representation for all the staff in a particular group such as the group of registered staff, unassigned staff or the group of staff assigned to a particular shift.

Objects Created

Note: Refer to Figure 1 for easy comprehension of the following

1) 1 ShiftManServer object will be created for the program to be used.
2) 1 Roster object is created each time newRoster() is called with valid parameters.
3) 7 DayOfWeek objects are created as soon as a Roster object is created.
   2 StaffGroup objects representing the group of registered staff and the group of unassigned staff are created as soon as a Roster object is created.
4) 7 ShiftsGroup objects are created as soon as a Roster object is created as 1 ShiftsGroup object per DayOfWeek object is created.
5) 1 Shift object is created each time addShift() is called with valid parameters.
6) 1 StaffGroup object is created each time a Shift object is created. This StaffGroup object represents the staff assigned to the shift.
7) 1 Staff object is created each time registerStaff() is called with valid parameters.

In total, 21 objects (1+1+7+2+7+1+1+1) are created when 1 shift is added and 1 staff member is registered. After that, 1 object per shift addition and 1 object per staff member registration are created.

Errors Detected

There are 5 methods in which errors are detected. The methods followed by the errors detected in them, respectively, are:

1) newRoster():
    - When the method is called with the provided name for the shop being null or empty
2) setWorkingHours():
    - When the working hours' start time is the same as or after the end time
3) addShift():
    - When the value given for the day of the week is invalid
    - When the shift start time is the same as or after the end time
    - When the shift is not within working hours
    - When the shift to be added is for a day on which the shop is not open
4) getRegisteredStaff():
    - When newRoster() has not yet been called.
5) getRosterForWorker():
    - When the method is called for a worker who is not registered

Use Of Checked Exceptions

A ShiftManServerException class, which extends the Exception class, was created. When any of the errors in methods mentioned under the "Errors Detected" heading above is detected, a checked exception with an appropriate message is thrown which is caught in the respective method in the ShiftManServer class. Then, the message contained in the exception is added to the output of the particular method (in which the error was detected).

For example: When it is tried to add a shift whose start time is the same as or after the end time, the error is first detected in the addShift() method of the ShiftsGroup class. It is then passed to the DayOfWeek class, which again passes it to the Roster class, which in turn passes it to the ShiftManServer class where the exception is finally caught and the error message string is made the output of the method.