Edinburgh Napier University

Report

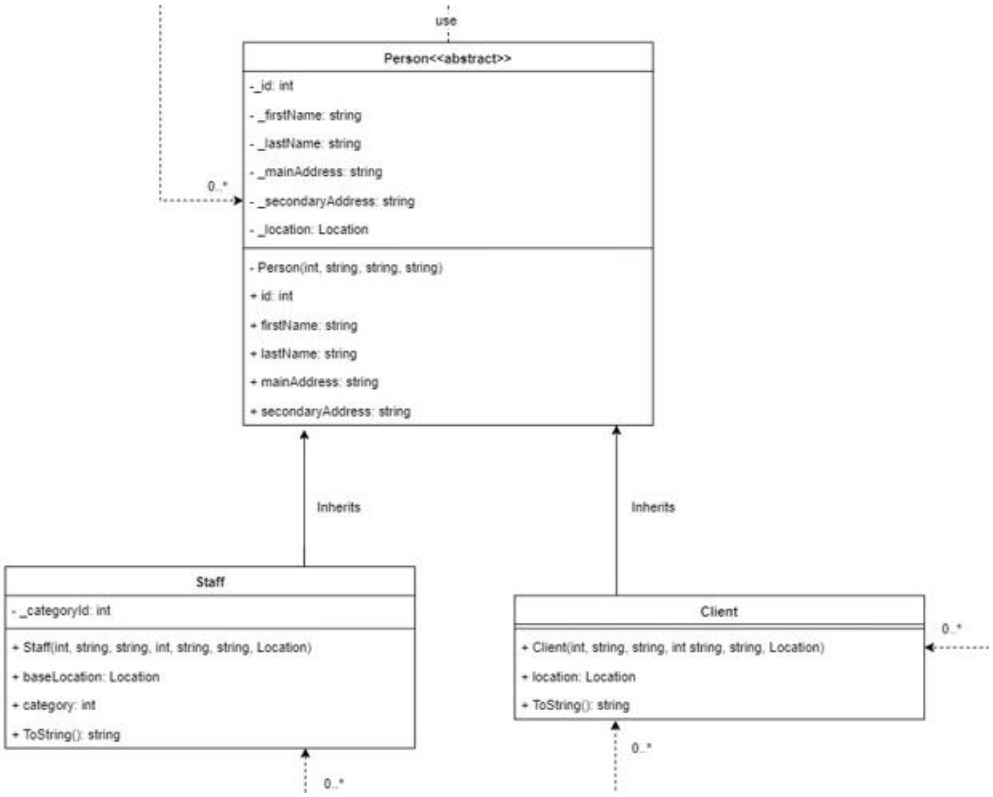
Matric: 40342250 (Mihail Yonchev)

Module: SET08119 2019 (Object Software Development)

Project Introduction: Edinburgh Health Board provide a number of services to elderly and housebound clients. A system is required to ensure to keep track of visits that must be made to clients. They have a collection of staff of the following categories: General Practitioner, Community Nurse, Social Worker, Care Worker.

1. **Class Diagram. Aspects:**
   - **a.** Identification of classes:
     - Presentation Layer: **MainWindow.**
     - Business Layer: **HealthFacade**, **VisitFactory, Visit, PersonFactory, Client**, **Staff**, **Person**, **Location.**
     - Data Layer: **Database.**
   - **b.** Use of relations

     Person class is an abstract class, which is being inherited by Staff and Client.



   - **c.** Use of three layer architecture

     The architecture I used is the 3 tier architecture: **Business Layer**, **Presentation Layer** and **DataLayer**. The reason is to make the logical separation between the different layers,

which operate and serve solutions to the different tasks. Using the Three Layer Architecture makes managing the project easier and allows me to make my work scalable and the project more perspective.
Purposes of each layer:

1. Presentation Layer – Used for getting user data and then passing it to the Business Layer for further procedure.
2. Business Layer – Sitting between the Presentation Layer and the Data Layer and implements business logic (rules from the specification). It communicates with Data Layer for further procedures such as saving the Data to secondary memory.
3. Data Layer – Provides a way for storing and accessing data in persistent storage. The storage type I choose is C# MySQL Database.

d. Use of design Patterns.
1. My Database class is **Singleton** because I don't want two different instances of this class to simultaneously use the class and write – which may lead to problems such as missing information.
2. I have implemented two **Singleton Factories**: PersonFactory, which is taking care of creating the two types of persons – Staff and Client and VisitFactory, which takes care of creating the different types of visits.

What are the principle advantages of the 3-layer architecture?

Improved scalability, performance and availability. When we use the three tiers, we can develop each layer individually and concurrently by different team who is specialized in this. For example, the data tier stores information, the business layer handles the logic and the presentation layer is the graphical user interface. The layers may not run on the same server. Another important advantage of using the 3-layer architecture is that the exiting application can be temporarily retained and encapsulated within new tier of which it becomes a component.
References: Lecture Slides + https://searchsoftwarequality.techtarget.com/definition/3-tier-application

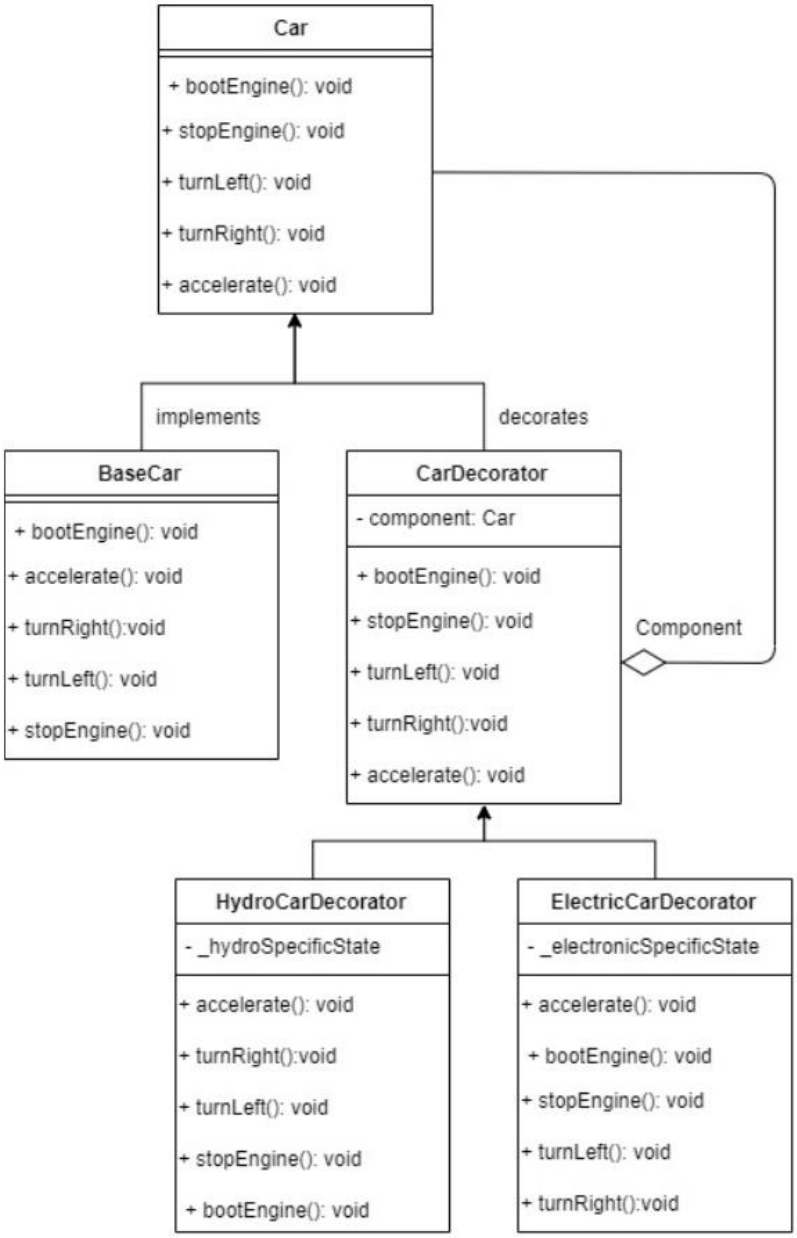
Describe, with an example, the use of the decorator pattern. Maximum 2 A4 sides, you must include at least 1 class diagram showing the pattern.

Intro: Before diving to my example I will first explain what is the use of the decorator pattern. It is used to add extra features to a particular object. The reason why it is so popular it is because unlike inheritance it can be carried out at runtime.

Example: A car is a vehicle on it's own to which we can apply decorations to make it more specific, accurate and to apply to the users need. For example a Car Class defines the basic requirements

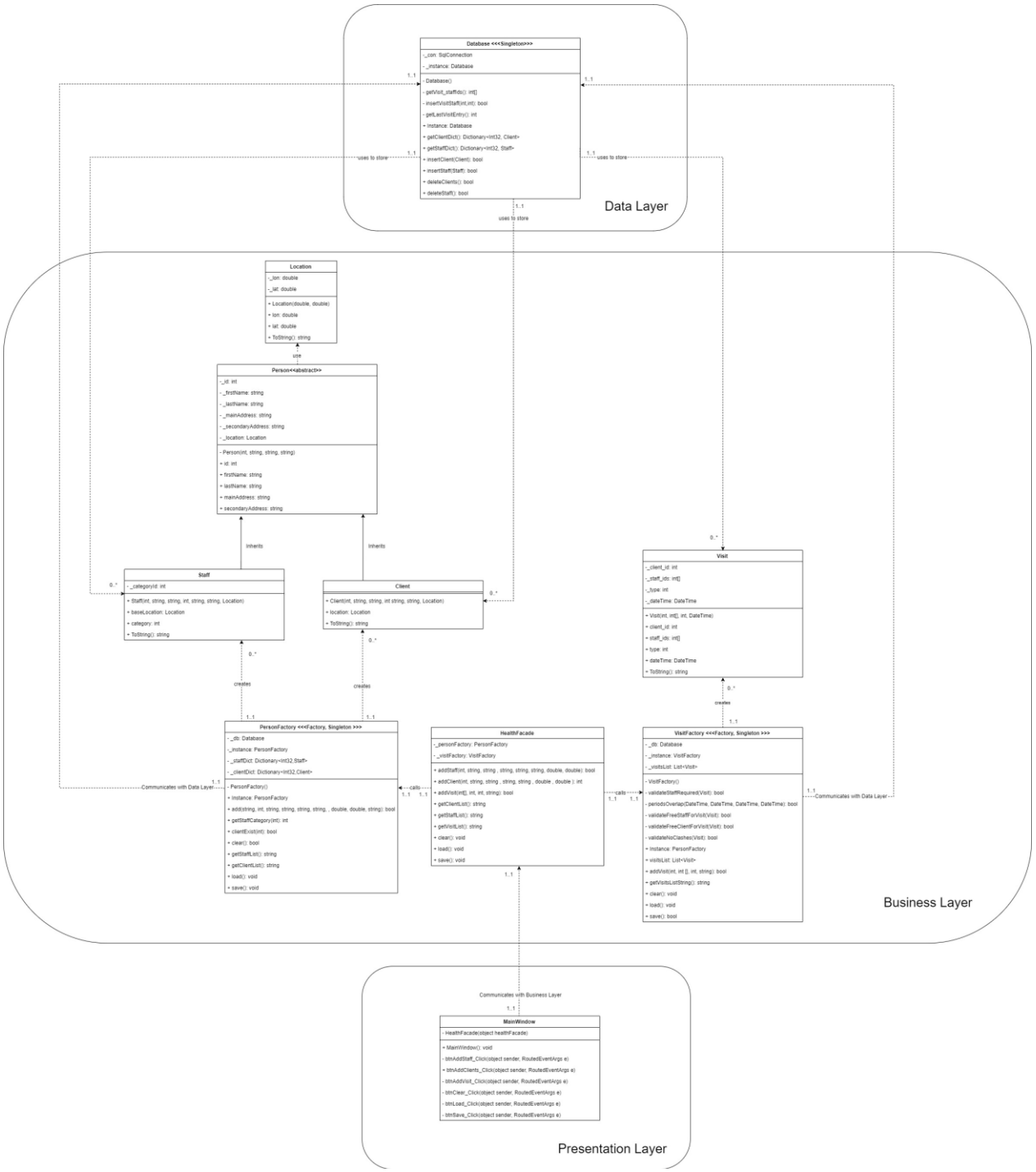
for every car – as we see from the UML Below these are: bootEngine(), stopEngine(), turnRight(), turnLeft(), accelerate(). (In real life there would be many more properties but I will keep it simple in order to explain the pattern). We all know that there is not only one type of car – there is so much different kinds of additions that can be added such as making the car Electric, or making it Hydraulic. If we now think of a way to get a car to be electric and hydraulic at the same time at run-time the implementation will be very complex. To solve this kind of problem we use the decorator pattern. In this way we will be able to make modifications at runtime and be more flexible.

UML Showing the Example Pattern:

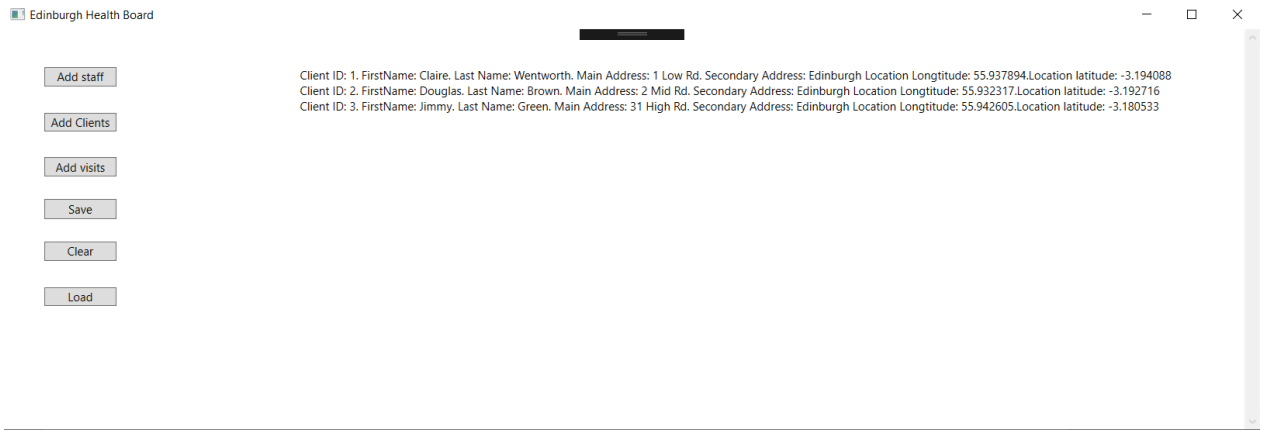


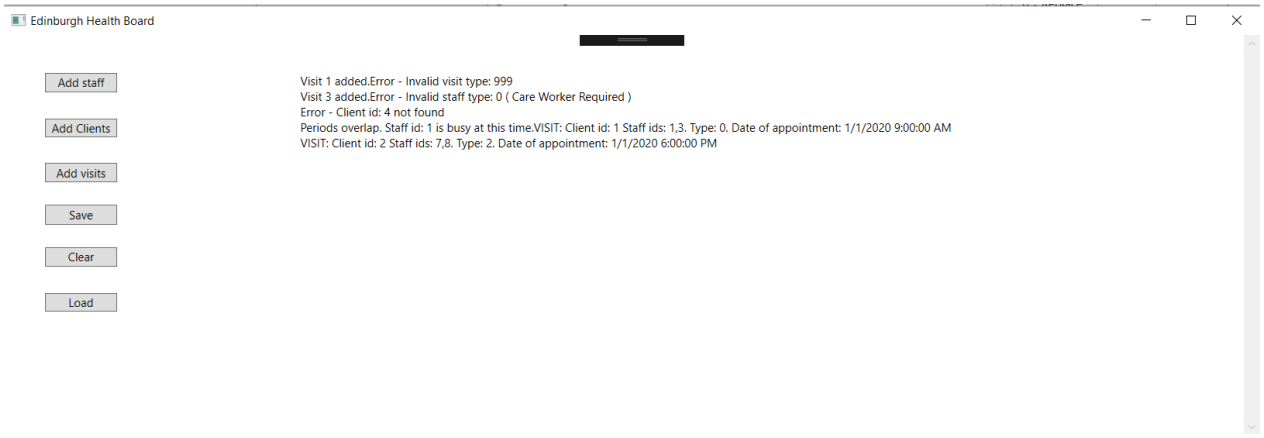
References: Used draw.io to draw the UML and lecture slides for information.

Scalability: In order to make the app more scalable and more secure in terms of to ensure that it is operating correctly I created the BusinessTest Layer containing unit tests (PersonFactoryGroup).
//Thanks for bearing with me up to here! Here is a full UML Diagram of the Application Design and GUI.


Database <<<Singletoo>>>
- _con: SqlConnection
- _instance: Database
+ Database()
- getVisit_staffIds(): int[]
- insertVisitStaff(int,int): bool
- getLastVisitEntry(): int
+ instance: Database
+ getClientDict(): Dictionary<Int32, Client>
+ getStaffDict(): Dictionary<Int32, Staff>
+ insertClient(Client): bool
+ insertStaff(Staff): bool
+ deleteClients(): bool
+ deleteStaff(): bool
Data Layer
uses to store
Location
- _lon: double
- _lat: double
+ Location(double, double)
+ lon: double
+ lat: double
+ ToString(): string
use
Person<<abstract>>
- _id: int
- _firstName: string
- _lastName: string
- _mainAddress: string
- _secondaryAddress: string
- _location: Location
- Person(int, string, string, string)
+ id: int
+ firstName: string
+ lastName: string
+ mainAddress: string
+ secondaryAddress: string
Inherits
Staff
- _categoryId: int
+ Staff(int, string, string, int, string, string, Location)
+ baseLocation: Location
+ category: int
+ ToString(): string
Client
+ Client(int, string, string, int string, string, Location)
+ location: Location
+ ToString(): string
Visit
- _client_id: int
- _staff_ids: int[]
- _type: int
- _dateTime: DateTime
+ Visit(int, int[], int, DateTime)
+ client_id: int
+ staff_ids: int[]
+ type: int
+ dateTime: DateTime
+ ToString(): string
creates
PersonFactory <<<Factory, Singleton >>>
- _db: Database
- _instance: PersonFactory
- _staffDict: Dictionary<Int32,Staff>
- _clientDict: Dictionary<Int32,Client>
- PersonFactory()
+ Instance: PersonFactory
+ add(string, int, string, string, string, string, , double, double, string): bool
+ getStaffCatagory(int): int
+ clientExist(int): bool
+ clear(): bool
+ getStaffList(): string
+ getClientList(): string
+ load(): void
+ save(): void
Communicates with Data Layer
HealthFacade
- _personFactory: PersonFactory
- _visitFactory: VisitFactory
+ addStaff(int, string, string , string, string, string, double, double): bool
+ addClient(int, string, string , string, string , double , double): int
+ addVisit(int[], int, int, string): bool
+ getClientList(): string
+ getStaffList(): string
+ getVisitList(): string
+ clear(): void
+ load(): void
+ save(): void
calls
VisitFactory <<<Factory, Singleton >>>
- _db: Database
- _instance: VisitFactory
- _visitsList: List<Visit>
- VisitFactory()
- validateStaffRequired(Visit): bool
- periodsOverlap(DateTime, DateTime, DateTime, DateTime): bool
- validateFreeStaffForVisit(Visit): bool
- validateFreeClientForVisit(Visit): bool
- validateNoClashes(Visit): bool
+ Instance: PersonFactory
+ visitsList: List<Visit>
+ addVisit(int, int [], int, string): bool
+ getVisitsListString(): string
+ clear(): void
+ load(): void
+ save(): bool
Communicates with Data Layer
Business Layer
Communicates with Business Layer
MainWindow
- HealthFacade(object healthFacade)
+ MainWindow(): void
- btnAddStaff_Click(object sender, RoutedEventArgs e)
+ btnAddClients_Click(object sender, RoutedEventArgs e)
- btnAddVisit_Click(object sender, RoutedEventArgs e)
- btnClear_Click(object sender, RoutedEventArgs e)
- btnLoad_Click(object sender, RoutedEventArgs e)
- btnSave_Click(object sender, RoutedEventArgs e)
Presentation Layer


Edinburgh Health Board
Add staff
Add Clients
Add visits
Save
Clear
Load
Staff ID: 1. FirstName: James. Last Name: Henry. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 2. FirstName: Sally. Last Name: Wilson. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 3. FirstName: Roland. Last Name: Smith. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 4. FirstName: Mary. Last Name: Jones. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 5. FirstName: Phillip. Last Name: Jones. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 6. FirstName: Martha. Last Name: Rigg. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 7. FirstName: Mike. Last Name: Heathcoat. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164
Staff ID: 8. FirstName: Jo. Last Name: Shaw. Main Address: 21 Accia Road. Secondary Address: Edinburgh Location Longtitude: 55.932221.Location latitude: -3.214164


Edinburgh Health Board
Add staff
Add Clients
Add visits
Save
Clear
Load
Client ID: 1. FirstName: Claire. Last Name: Wentworth. Main Address: 1 Low Rd. Secondary Address: Edinburgh Location Longtitude: 55.937894.Location latitude: -3.194088
Client ID: 2. FirstName: Douglas. Last Name: Brown. Main Address: 2 Mid Rd. Secondary Address: Edinburgh Location Longtitude: 55.932317.Location latitude: -3.192716
Client ID: 3. FirstName: Jimmy. Last Name: Green. Main Address: 31 High Rd. Secondary Address: Edinburgh Location Longtitude: 55.942605.Location latitude: -3.180533


Edinburgh Health Board
Add staff
Add Clients
Add visits
Save
Clear
Load
Visit 1 added.Error - Invalid visit type: 999
Visit 3 added.Error - Invalid staff type: 0 (Care Worker Required)
Error - Client id: 4 not found
Periods overlap. Staff id: 1 is busy at this time.VISIT: Client id: 1 Staff ids: 1,3. Type: 0. Date of appointment: 1/1/2020 9:00:00 AM
VISIT: Client id: 2 Staff ids: 7,8. Type: 2. Date of appointment: 1/1/2020 6:00:00 PM