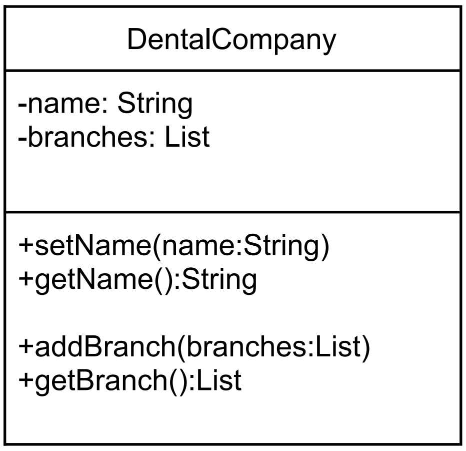
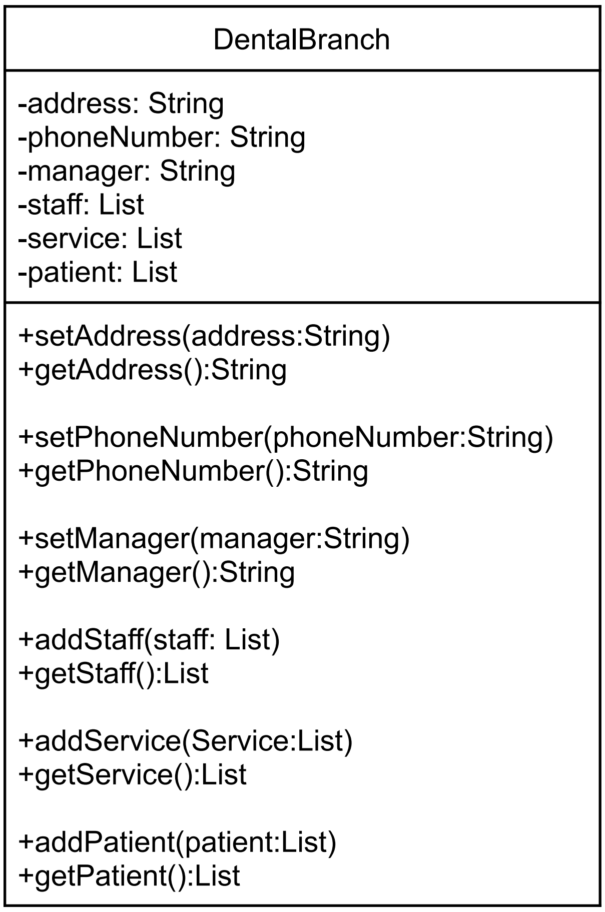
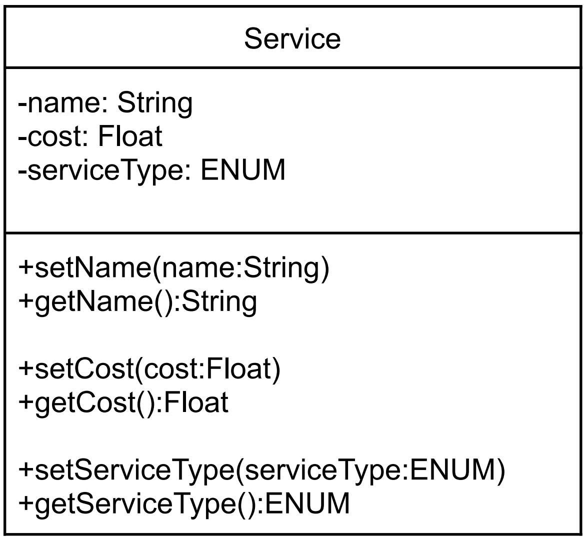
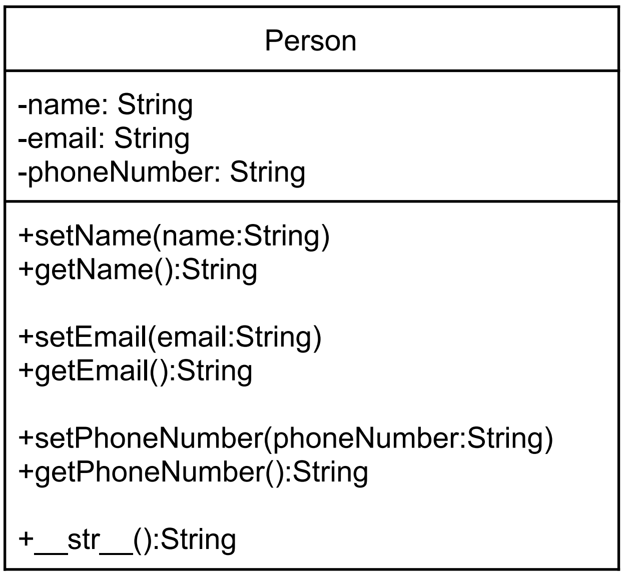
# Assignment 2: Software Implementation - OO Project with GUI

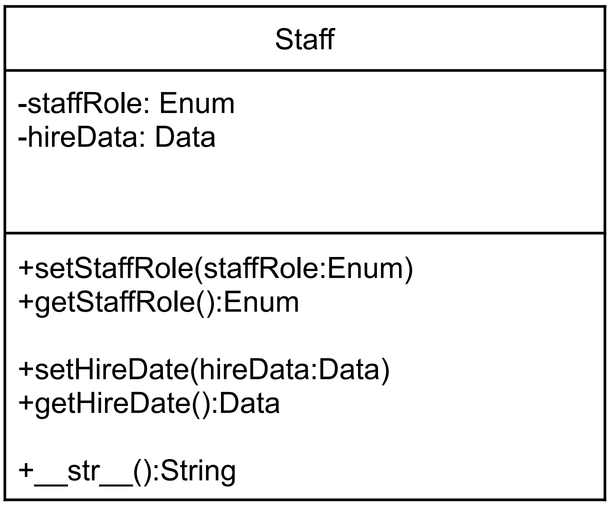
1. **UML Class** Diagram and Description

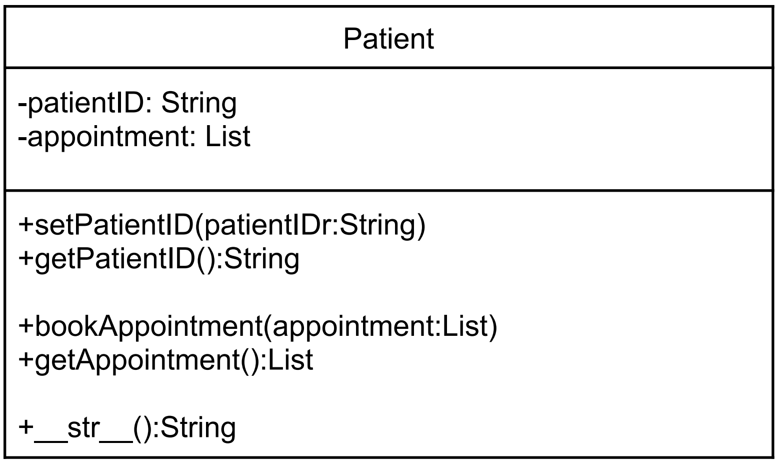


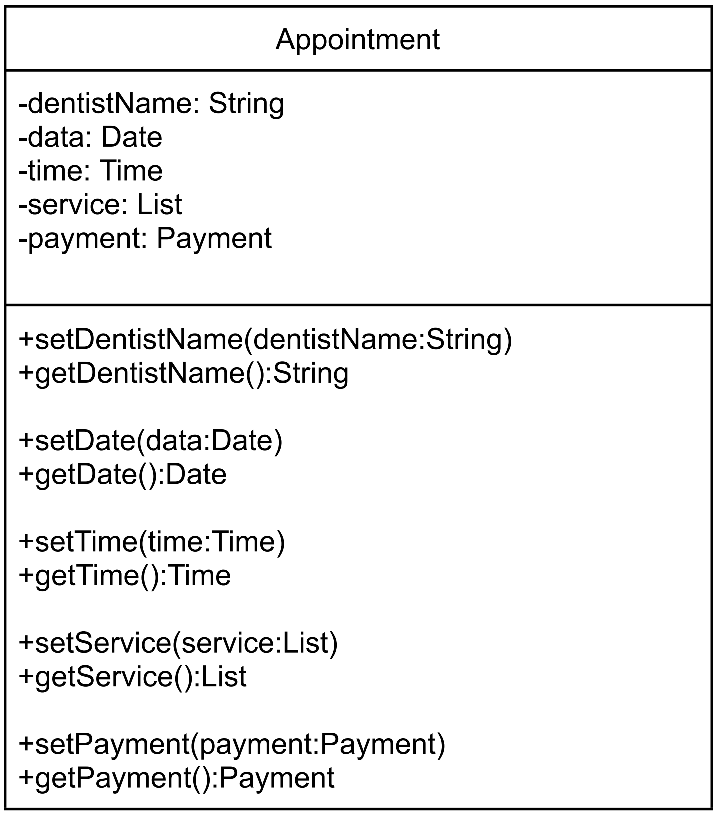


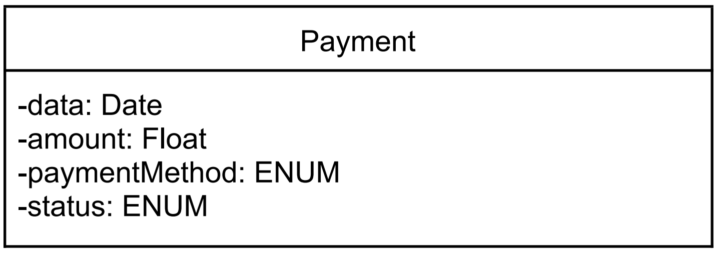

Service
-name: String
-cost: Float
-serviceType: ENUM
+setName(name:String)
+getName():String
+setCost(cost:Float)
+getCost():Float
+setServiceType(serviceType:ENUM)
+getServiceType():ENUM


Person
-name: String
-email: String
-phoneNumber: String
+setName(name:String)
+getName():String
+setEmail(email:String)
+getEmail():String
+setPhoneNumber(phoneNumber:String)
+getPhoneNumber():String
+__str__():String


Staff
-staffRole: Enum
-hireData: Data
+setStaffRole(staffRole:Enum)
+getStaffRole():Enum
+setHireDate(hireData:Data)
+getHireDate():Data
+__str__():String


Patient
-patientID: String
-appointment: List
+setPatientID(patientIDr:String)
+getPatientID():String
+bookAppointment(appointment:List)
+getAppointment():List
+__str__():String


Appointment
-dentistName: String
-data: Date
-time: Time
-service: List
-payment: Payment
+setDentistName(dentistName:String)
+getDentistName():String
+setDate(data:Date)
+getDate():Date
+setTime(time:Time)
+getTime():Time
+setService(service:List)
+getService():List
+setPayment(payment:Payment)
+getPayment():Payment


Payment
-data: Date
-amount: Float
-paymentMethod: ENUM
-status: ENUM

```
+setDate(data:Date)
+getDate():Date

+setAmount(total:Float)
+getAmountl():Float

+setPaymentMethod(paymentMethod:ENUM)
+getPaymentMethod():ENUM

+setStatus(status:ENUM)
+getStatus():ENUM
```

Relationships:
**dentalCompany** has a **dentalBranch.** One dentalCompany can have many branches, so It is a composition as if the dentalCompany was removed, the branch cannot still exist independently since it is related to and managed through the company. Also, the lifetime of a branch depend on the company itself, and the addition of it is controlled by the class dentalCompany.
—----------
**dentalBranch** has one-to-many binary association with **Staff** and vice versa. One dentalBranch can have a lot of staff, while each staff is related to one dentalBranch. If the dentalBranch closed, the staff could go to a different branch and look for a different job, they will be able to exist without it.
—----------
**dentalBranch** has many-to-many association with **Service** and vice versa. One dentalBranch can have many services, and each service can be related to multiple dental branches. The same service may be provided by more than one dentalBranch and vice versa. There is no ownership or containment link between the entities, and each one may interact with several instances of the other while remaining separate from it. In our case, it is a dental company, so each clinic will have the same servies as it is not a hospital(if it was the relationship will differ).
—----------
**dentalBranch** has one-to-many association with **Pateint** and vice versa. One dentalBranch can have a lot of patients, while each patient is related to one dentalBranch. It is a composition since the branch cannot exist without patients, and a patient should be related to a branch. The patients would be impacted and unable to receive treatment if the branch were to be eliminated or ceased to exist. The branch would also be impacted if it has no patients, it won't be able to gain profit or continue operating.
—----------
**dentalBranch** has a one-to-many association with **Appointment** and vice versa. One dentalBranch can have a lot of appointments, while each appointment is related to one dentalBranch, also many appointments can be related to one dentalBranch . It is a composition since the branch cannot exist without appointments, and an appointment should be related to a branch. The appointment would be impacted if the branch were to be eliminated or ceased to exist. The branch would also be impacted if it has no appointments, it won't be able to gain profit or continue operating, also there will be disorganization.
—----------

The relationship between an **appointment** and a **service** is many-to-many association since an appointment may be for a multiple services and a service may be provided to multiple appointments. It is a binary association as it connects two classes.
—----------
**Patient** has a one-to-many relationship with the class **Appointment** and vice versa because one patient can make several appointments yet only one patient can book an appointment. It is a binary association as it connects two classes.
—----------
The relationship between an **appointment** and a **staff** is a binary association. Each Appointment is associated with one or more Staff members who are in charge of delivering the services necessary for it to take place. A number of staff members may be involved in delivering a service for a specific appointment, and each staff member may be involved in a number of appointments, making this association a many-to-many relationship and vice versa.
—----------
There is a one-to-one binary association between **Payment** and **Appointment** and vice versa as a Payment can be made for a single Appointment and an Appointment can only have one Payment . Assuming that just a single payment can be made only and also a composition as a payement cannot be done without having an appointment first, so they depend on each other.
—----------
The class **Person** is the parent class and Both **Staff** and **Patient** inherit from it. It is a hierarchical inheritance, where more than one child(Staff and Patient) class is derived from a single-parent class(Person).
—---------
There is one-to-many binary association between **Staff** and **Patient**, and vice versa. A staff member can interact with one or more patients and a patient can interact with one or more staff member. (It could also be represented as a composition relationship if a staff member can only be associated with one patient at a time (e.g. a dentist working on a patient).)
—---------
Each service can has one or more staff member, and the staff members will be repsible for that service. If service didn't exist, the staff members can take another service or choose a different one to work with. So the relationship between **Staff** and **service** is aggregation where there is an independence between the conitaed and container class.
—---------
There is one-to-many binary association between **service** and **Patient** and vice versa. Each patient can one or more service, each service can has one or more patient. Assuming that the serivce can be done by multiple staff.
—---------
The relationship between Service and Payment, it is an aggregation because a Payment can be done for one or more instances of Service, but a Payment cannot exist without having at least one Service associated with it.
—---------
The relationship between the classes **Patient** and **Payment** is compsotion, as if we didn't have patient, they won't be a payment, so they depend on each other. Each patient has one single payment, but a single payment can be done for one patient at a time. So it is considered a one-to-one binary association.
—---------

Each **patient** has one **payment**(one-to-one binary association) , and **payment** can be done for multiple **patients**(one-to-many binary association . Also, it is a composition as a payment cannot be done without the existence of a patient.

**Assumptions:**

- Each service has a unique name.
- Each staff member has a unique name.
- Each patient has a unique name and phone number.
- Each appointment has a unique combination of date, time, patient, and staff.
- Each payment has a unique combination of date, patient, and branch.