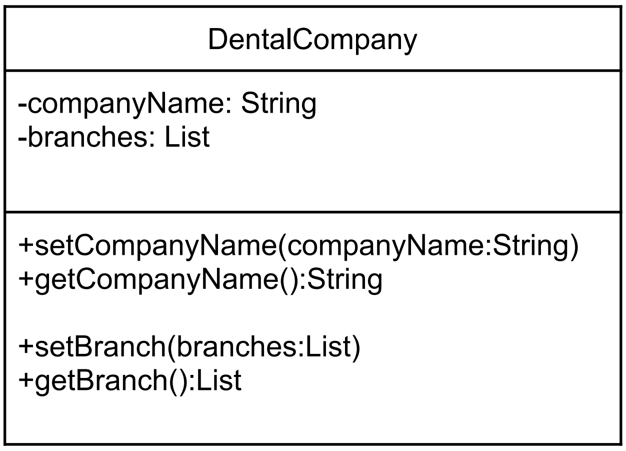
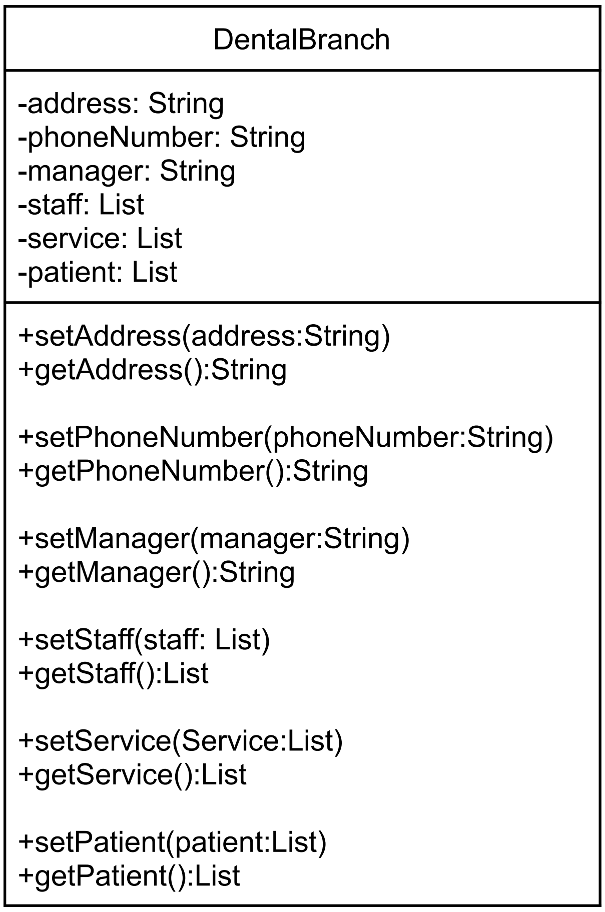
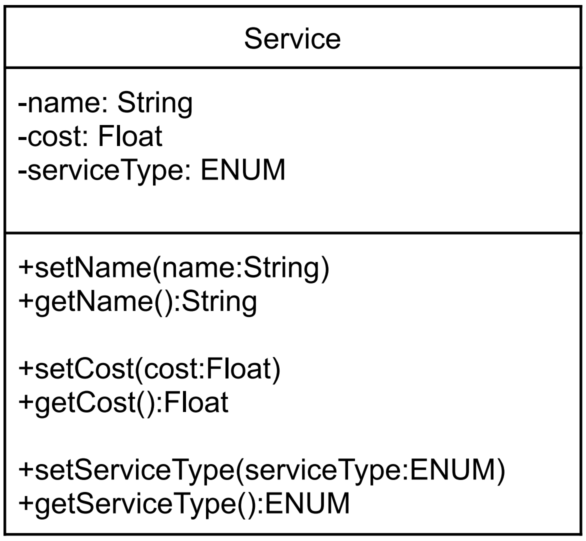
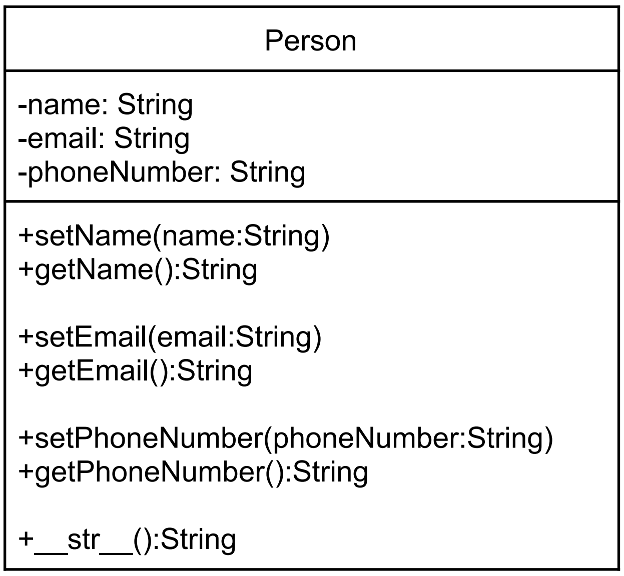
Assignment 2: Software Implementation - OO Project with GUI

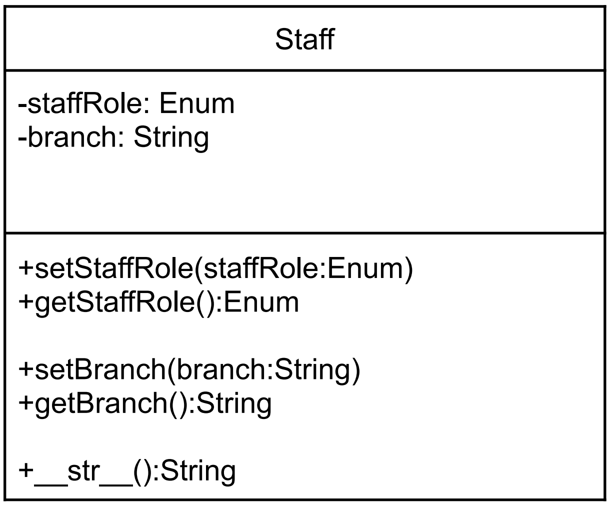
1. **UML Class** Diagram and Description

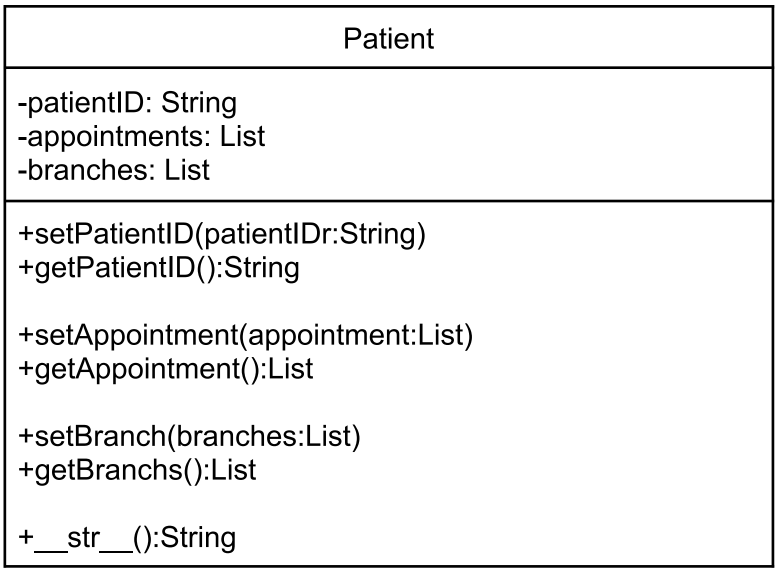





Service
-name: String
-cost: Float
-serviceType: ENUM
+setName(name:String)
+getName():String
+setCost(cost:Float)
+getCost():Float
+setServiceType(serviceType:ENUM)
+getServiceType():ENUM


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
-branch: String
+setStaffRole(staffRole:Enum)
+getStaffRole():Enum
+setBranch(branch:String)
+getBranch():String
+__str__():String



Patient

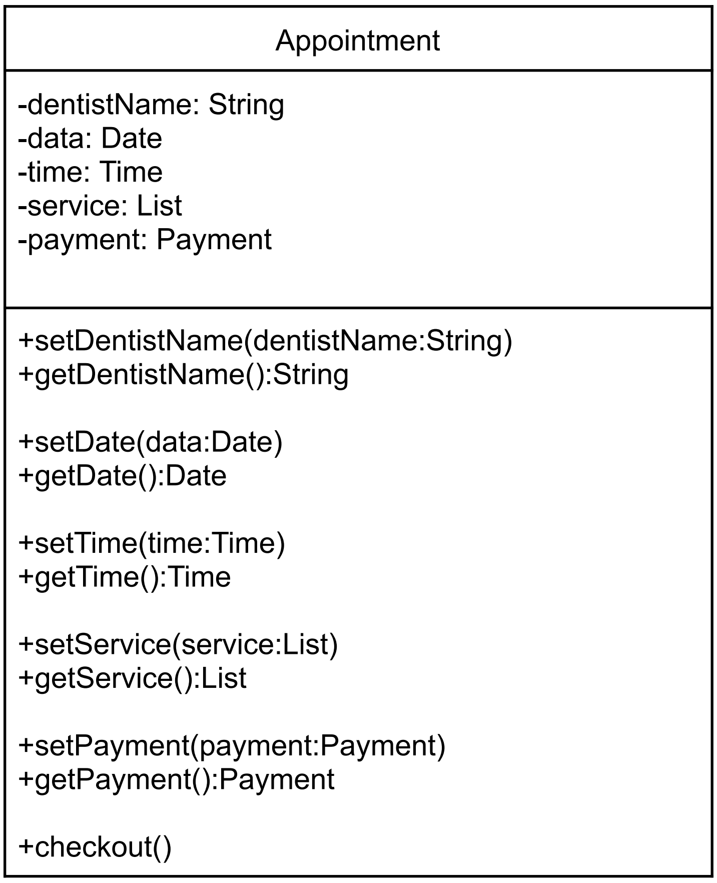
-patientID: String
-appointments: List
-branches: List

+setPatientID(patientIDr:String)
+getPatientID():String

+setAppointment(appointment:List)
+getAppointment():List

+setBranch(branches:List)
+getBranchs():List

+__str__():String



Appointment

-dentistName: String
-data: Date
-time: Time
-service: List
-payment: Payment

+setDentistName(dentistName:String)
+getDentistName():String

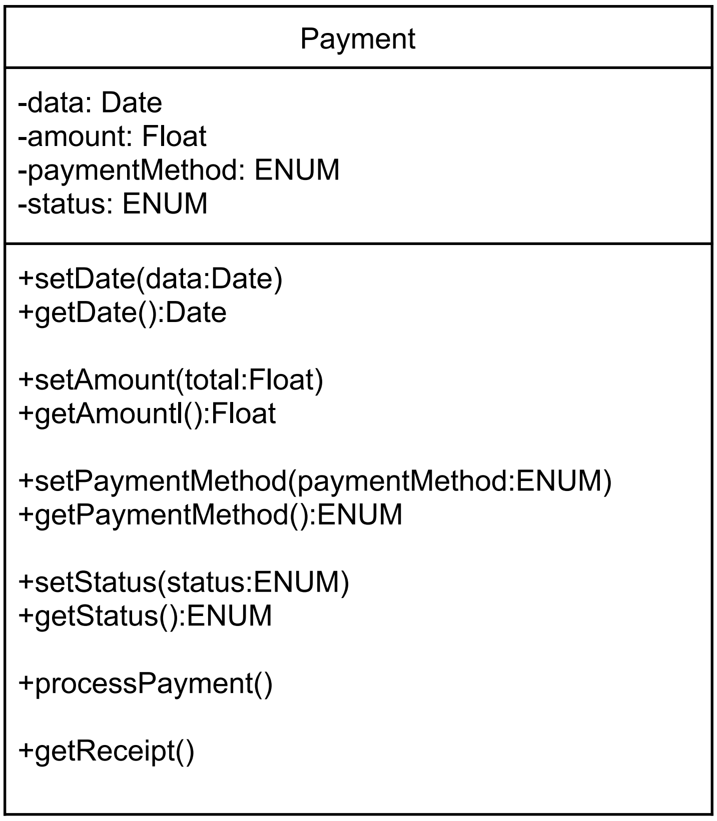
+setDate(data:Date)
+getDate():Date

+setTime(time:Time)
+getTime():Time

+setService(service:List)
+getService():List

+setPayment(payment:Payment)
+getPayment():Payment

+checkout()



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

+processPayment()

+getReceipt()

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 depends on the company itself, and the addition of it is controlled by the class dentalCompany.
—----------
dentalBranch has a 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. Without the Staff class, the DentalBranch class could not keep track of its employees, and the Staff class could not be assigned to a branch or know to which branch it belongs without the DentalBranch class.
—----------
dentalBranch has a many-to-many binary association with **Service**. 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. As the class dentalBranch has a list of services, it keeps track of them. Here in our case, each branch can have the services provided such as cleaning or filling, which are not specific to one of the branches.
—----------
dentalBranch has a many-to-many binary association with **Patient** and vice versa. One dentalBranch can have a lot of patients, and each patient can be related to one or more dentalBranch. If a dentalBranch was deleted, the patient will still exist and go to different branches. As the class dentalBranch has a list of patients, it keeps track of them. It is a good approach as we don't limit the patient and let it be associated with one branch only, they have the freedom to choose.
—----------

The relationship between an **appointment** and a **service** is a one-to-many association since an appointment may be for 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 patient class keeps track of the appointments the patient books before coming to the branch which allows easy access to all appointments for a particular patient. It is simpler and flexible to get the list of appointments through each patient instead of storing it on the dentalBranch which will complicate things and let us search through all the branches to find the relevant appointments.
—----------
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.
—----------
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).
—---------
We maintains a record of the appointment's payment in the Payment class. The payment instance variable is initialized to None when an appointment is established, denoting that the appointment has not yet been paid for. The overall cost of the appointment is determined and a Payment object is created with the final amount when the checkout method is called on the appointment object. The appointment object's payment instance variable is subsequently allocated to the Payment object. Through its instance variables and methods, the Payment object keeps track of the payment amount, date, payment method, and status (paid or unpaid). To receive a formatted receipt with the payment information, use the getReceipt function of the Payment class.
—---------
The relationship between the classes **Patient** and **Payment** is managed through the appointment class as discussed above and in this case it is indirect. In general, each **patient** has one **payment**(one-to-one binary association).

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 date, time, patient, and staff.
- Each payment has a unique date, patient, and branch.
- Each payment can be done a one single time
- each staff member can have only one staff role at a branch.