Hospital Bill Management System

Requirement

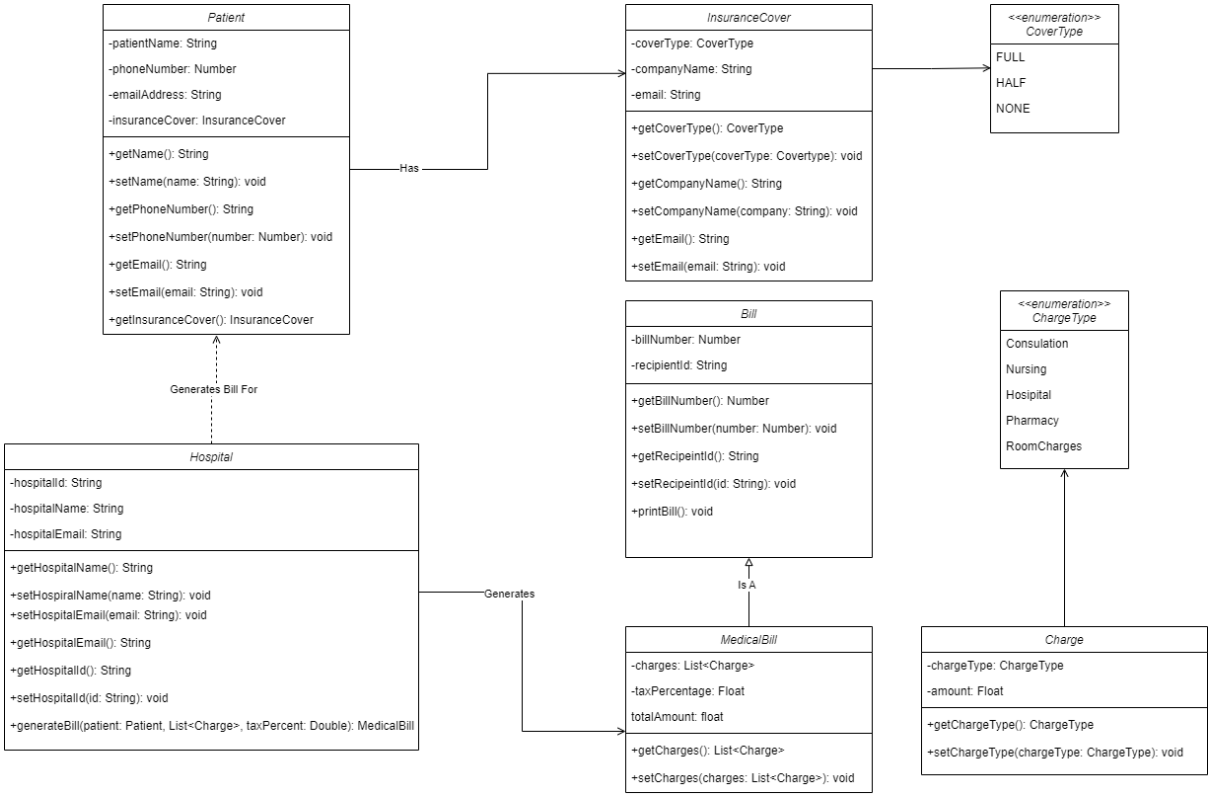
To design a hospital bill management system that generates the bill for the patient based on what services they availed.

Use Case

Two actors were identified in the system:

- The Patient/User provides their information to the system so that it can be persisted and asks for the MedicalBill to be generated.
- The Hospital Admin is responsible for updating the hospital information in the system and for providing tax and charges info as the input to the MedicalBill generating logic.

Class Diagram



A total of **6 classes** and **2 enums** could be identified in the system.

1. Patient- represents the patient entity which contains information about the patient's personal details and their Insurance Cover.
2. Insurance Cover- represents the InsuranceCover entity which contains info about the InsuranceType, and the Insurance Company Info.
3. Hospital- represents the Hospital entity which contains info about the hospital details and contains methods to generate the bill for a given patient.
4. Bill- A generic bill type that consists of billNumber and recipient identifier.
5. MedicalBill- A concrete entity that inherits from Bill class and contains specific information regarding the MedicalBill.
6. Charge- Represents the type of Charge incurred added to the MedicalBill.

Python Code

```
from enum import Enum


class CoverType(Enum):
    """ Enum Class for Cover type"""
    FULL = 1
    HALF = 2
    NONE = 3

class InsuranceCover:
    """Class to represent insurance cover"""

    #Initializer
    def __init__(self, coverType:CoverType, companyName = "",
companyEmail=""):
        self.__coverType = coverType
        self.__companyName = companyName
        self.__companyEmail = companyEmail

    #Setter and Getter functions
    def setCoverType(self, cover:CoverType):
        self.__coverType = cover

    def getCoverType(self):
        return self.__coverType

    def setCompanyName(self, name):
        self.__companyName = name

    def getCompanyName(self):
        return self.__companyName

    def setCompanyEmail(self, email):
        self.__companyEmail = email

    def getCompanyEmail(self):
        return self.__companyEmail

    #Del function
    def __del__(self):
        print("Deleting an instance of the class Insurance Cover")

class ChargeType(Enum):
    """Enum type class for chargeType"""
    Consulatation = 1
    Nursing = 2
```

```
    Hospital = 3
    Pharmacy = 4
    RoomCharges = 5

class Patient:
    """Class to represent a Patient"""

    #Initializer
    def __init__(self, insuranceCover:InsuranceCover, patientName="",
phoneNumber="", emailAddress=""):
        self.__patientName = patientName
        self.__phoneNumber = phoneNumber
        self.__emailAddress = emailAddress
        self.__insuranceCover = insuranceCover

    #Setter and Getter functions
    def setName(self, name):
        self.__patientName = name

    def getName(self):
        return self.__patientName

    def setPhoneNumber(self, number):
        self.__phoneNumber = number

    def getPhoneNumber(self):
        return self.__phoneNumber

    def setEmail(self, email):
        self.__emailAddress = email

    def getEmail(self):
        return self.__emailAddress

    #Function to get Cover type from the insurance cover
    def getInsuranceCover(self):
        return self.__insuranceCover.getCoverType()

    #Del function
    def __del__(self):
        print("Deleting an instance of the class Patient")



class Charge:
    """Class to represent the charge type and amount"""

    #Initialzier
    def __init__(self, chargeType:ChargeType, amount:float):
```

```
        self.__chargeType = chargeType
        self.amount = amount

    #Setter and getter functions
    def setChargetype(self, charge:ChargeType):
        self.__chargeType = charge

    def getChargeType(self):
        return self.__chargeType

    def __del__(self):
        print("Deleting an instance of the class Charge")

class Hospital:
    """Class to represent hospital"""

    #Initializer
    def __init__(self, hospitalId = "", hospitalName = "", hospitalEmail =
""):
        self.__hospitalId = hospitalId
        self.__hospitalName = hospitalName
        self.__hospitalEmail = hospitalEmail
        self.__totalAmount = 0

    #Setter and Getter functions

    def setHospitalName(self, name):
        self.__hospitalName = name

    def getHospitalName(self):
        return self.__hospitalName

    def setHospitalID(self, hospitalId):
        self.__hospitalId = hospitalId

    def getHospitalID(self):
        return self.__hospitalId

    def setHospitalEmail(self, email):
        self.__hospitalEmail = email

    def getHospitalEmail(self):
        return self.__hospitalEmail

    #Function to generate Hospital Bill
    def generateBill(self, patient:Patient, chargeList:list[Charge],
taxPercent:float):
        return MedicalBill(chargeList, taxPercent, patient)
```

```
    #Del function
    def __del__(self):
        print("Deleting an instance of the class Hospital")

class Bill:
    """Class representing a Bill"""

    #Initializer
    def __init__(self, billNumber:float, recipientId = ""):
        self.__billNumber = billNumber
        self.__recipientId = recipientId

    #Setter and Getter Functions
    def setBillNumber(self, billNumber):
        self.__billNumber = billNumber

    def getBillNumber(self):
        return self.__billNumber

    def setRecipientId(self, recipientID):
        self.__recipientId = recipientID

    def getrecipientID(self):
        return self.__recipientId

    def printBill(self):
        print("Bill Details")

    #Del function
    def __del__(self):
        print("Deleting an instance of the class Bill")

class MedicalBill(Bill):
    """Class to represent MedicalBill"""

    #Initializer
    def __init__(self, charges:list[Charge], taxPercentage:float, patient:
Patient):
        super().__init__(123, patient.getEmail())

        self.__charges = charges
        self.__taxPerentage = taxPercentage

        #Calulating the total Amount
        self.__totalAmount = 0
        for charge in self.__charges:
            self.__totalAmount += charge.amount
```

```
        if patient.getInsuranceCover() == CoverType.FULL:
            self.__totalAmount = 0
        elif patient.getInsuranceCover() == CoverType.HALF:
            self.__totalAmount = self.__totalAmount / 2

        self.__totalAmount += ((self.__totalAmount) * self.__taxPerentage)/100

    #Setter and getter functions
    def setCharges(self, charges):
        self.__charges = charges

    def getCharges(self):
        return self.__charges

    def getTotalAmount(self):
        return self.__totalAmount

    #Print Bill method
    def printBill(self):
        """
        This method will contain the logic to calculate, format and print the
entire medical bill
        """
        pass

    #Del function
    def __del__(self):
        print("Deleting an instance of the class MedicalBill")
```

Github Repo Link

Summary of Learnings:

While working on the assignment, I learnt how to write getter and setter functions and what uses they have in real world applications. I also learnt how to identify use cases for a given application. I understood the UML class diagrams and how closely they are related to each other and how the data flows between them. It helped me visualize the classes and how they are behaving in the application. I also learnt why a modular code is preferred and how easy it becomes when a problem is broken into smaller pieces.