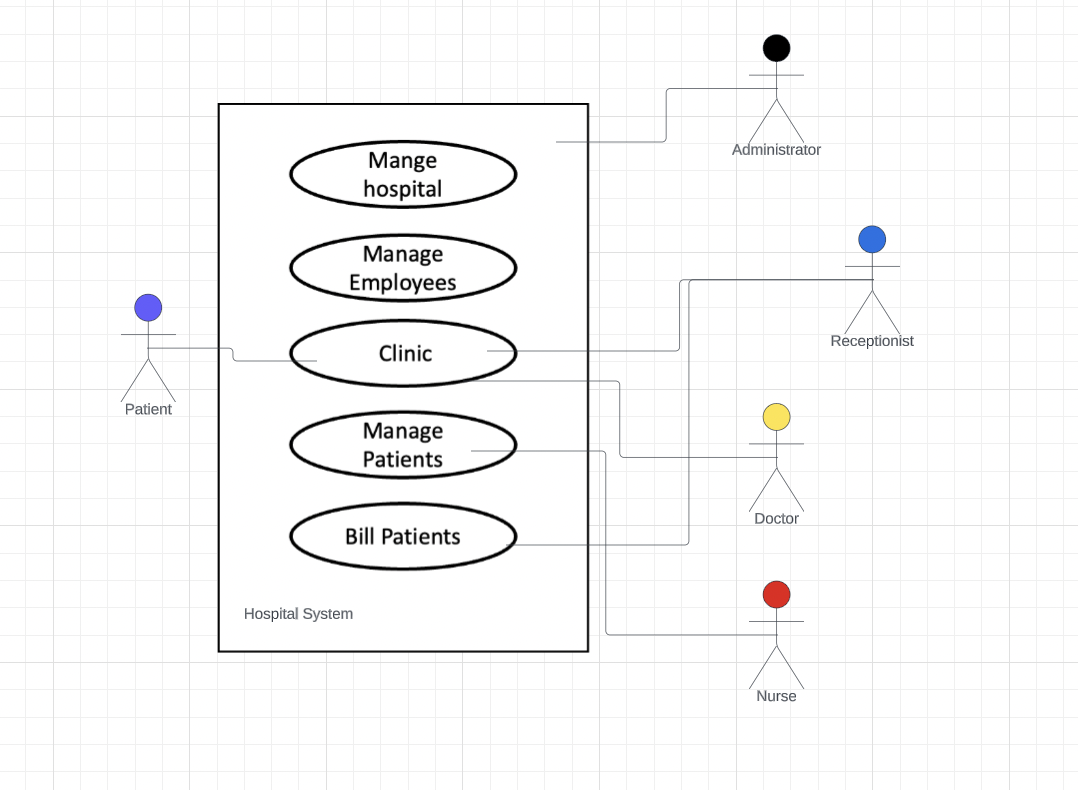
UML CASE DIAGRAM FOR A HOSPITAL SYSTEM



USE CASE DESCRIPTION:

|  |  |
| --- | --- |
| **Use Case:** | **Medical Billing** |
| **Trigger:** | Patients is being discharged |
| **Precondition:** | Doctor accepts the discharge |
| **Main Scenario:** |  |
| **1** | Patient arrives to the clinic |
| **2** | The receptionist takes his personal information and assign him/her to a nurse. |
| **3** | The nurse checks personal and takes medical information and takes him/her to the doctor |
| **4** | The doctor examines the patient and medicate him/her and discharge to the receptionist |
| **5** | The receptionist adds the doctor and nurse names to the billing form and the calculates the total |
| **6** | The receptionist take money |
| **Exceptions:** |  |
| **3a** | 1 The nurse sees medical problem is not emergency asks the patient to wait  2 The patient wait until the doctor call him/her |
| **4a** | 1. The doctor sees an emergency medical problem admits patient to the hospital 2. The patient is fine now , the doctor discharge |
|  |  |

|  |  |
| --- | --- |
| **Use Case:** | **Receptionist inputs Patient information** |
| **Trigger:** | Patient arrived |
| **Precondition:** | Patient token number appears |
| **Main Scenario:** |  |
| **1** | Patient arrives to the clinic and takes a number token |
| **2** | The receptionist presses a token caller |
| **3** | Patient goes to the receptionist |
| **4** | Receptionist asks for personal information and create an information wrist band for patient and sends file to the nurse |

|  |  |
| --- | --- |
| **Use Case:** | **Receptionist inputs Patient information when arrival** |
| **Trigger:** | Patient arrived |
| **Precondition:** | Patient token number appears |
| **Main Scenario:** |  |
| **1** | Patient arrives to the clinic and takes a number token |
| **2** | The receptionist presses a token caller |
| **3** | Patient goes to the receptionist |
| **4** | Receptionist asks for personal information and create an information wrist band for patient and sends file to the nurse |

|  |  |
| --- | --- |
| **Use Case:** | **Discharge** |
| **Trigger:** | Doctor accepts discharge |
| **Precondition:** | Patient has seen the doctor |
| **Main Scenario:** |  |
| **1** | Patient sees a doctor |
| **2** | Doctor examines the patient and medicate |
| **3** | The patient is feels well |
| **4** | The doctor accepts discharge and sends the patient to receptionists |

|  |  |
| --- | --- |
| **Use Case:** | **Receptionist bills the patient** |
| **Trigger:** | Doctor sends patient to the receptionist |
| **Precondition:** | Patient is being discharged |
| **Main Scenario:** |  |
| **1** | Doctor sends the patient to the receptionist |
| **2** | Receptionists add billing information and ask patient for insurance cover |
| **3** | The patient provides insurance cover |
| **4** | The receptionist calculates the total |
| **5** | Patient pays the bill |
| **6** | Receptionist takes the money |
| **Exceptions:** |  |
| **3a** | 1. The patient doesn’t have an insurance cover. 2. The patient pays the full bill. |

|  |  |
| --- | --- |
| **Use Case:** | **Insurance Cover** |
| **Trigger:** | The receptionist ask for the Insurance cover |
| **Precondition:** | Patient waits for the bill from the receptionist |
| **Main Scenario:** |  |
| **1** | Patient waits for the bill |
| **2** | Receptionists asks for the insurance |
| **3** | The patient gives the insurance |
| **4** | The receptionist checks if its Platinum or Gold |
| **Exception:** |  |
| **4a** | 1. If Platinum, the insurance covers the whole bill. 2. If Gold, the insurance cover half 3. If without insurance cover the patient pays the whole bill. |

UML CASE DIAGRAM

A diagram of a computer

Description automatically generated

PYTHON CLASS HOSPITAL

import datetime # Import the datetime module to work with date and time.  
from tabulate import tabulate  
  
  
class Hospital:  
 *"""  
 This class represents a hospital and its associated classes for managing personnel and patients.  
 """* class Person:  
 *"""  
 Represents a generic person with basic personal information.  
 """* def \_\_init\_\_(self):  
 self.\_name = None  
 self.\_emiratesId = None  
 self.\_cellPhone = None  
 self.\_insurance = None  
 self.\_dateofbirth = None  
  
 def set\_person\_info(self, fullname, emirates\_id, phone, insurance, dateofbirth):  
 *"""  
 Set the personal information of a person.  
  
 Args:  
 fullname (str): The full name of the person.  
 emirates\_id (str): The Emirates ID of the person.  
 phone (str): The phone number of the person.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the person.  
 dateofbirth (str): The date of birth of the person in "YYYY-MM-DD" format.  
 """* self.\_name = fullname  
 self.\_emiratesId = emirates\_id  
 self.\_cellPhone = phone  
 self.\_insurance = insurance  
 self.\_dateofbirth = dateofbirth if dateofbirth is not None else datetime.date.today()  
  
 def get\_person\_info(self):  
 *"""  
 Get the personal information of a person.  
  
 Returns:  
 tuple: A tuple containing personal information (name, Emirates ID, phone, insurance, date of birth).  
 """* return self.\_name, self.\_emiratesId, self.\_cellPhone, self.\_insurance, self.\_dateofbirth  
  
 def display\_person\_info(self):  
 *"""  
 Generate a formatted string representing a person's information.  
  
 Returns:  
 str: A formatted string containing the person's information.  
 """* return f'Full Name {self.\_name}, Emirates ID: {self.\_emiratesId}, Phone: {self.\_cellPhone}, ' \  
 f'Insurance: {self.\_insurance}, Date Of Birth: {self.\_dateofbirth}'  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of a person.  
  
 Returns:  
 str: A string containing the person's information.  
 """* return f'Full Name {str(self.\_name)}, ID: {str(self.\_emiratesId)}, Phone: {str(self.\_cellPhone)}, ' \  
 f'Insurance: {str(self.\_insurance)}, Date Of Birth: {str(self.\_dateofbirth)}'  
  
 class Employee(Person):  
 *"""  
 Represents an employee of the hospital with additional department and room information.  
 """* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.\_department = ""  
 self.\_room = ""  
  
 def set\_employee\_info(self, fullname, emirates\_id, phone, insurance, dateofbirth, department, room):  
 *"""  
 Set the information of an employee, including personal information, department, and room.  
  
 Args:  
 fullname (str): The full name of the employee.  
 emirates\_id (str): The Emirates ID of the employee.  
 phone (str): The phone number of the employee.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the employee.  
 dateofbirth (str): The date of birth of the employee in "YYYY-MM-DD" format.  
 department (str): The department where the employee works.  
 room (str): The room where the employee is assigned.  
 """* super().set\_person\_info(fullname, emirates\_id, phone, insurance, dateofbirth)  
 self.\_department = department  
 self.\_room = room  
  
 def get\_employee\_info(self):  
 *"""  
 Get the employee information, including personal information, department, and room.  
  
 Returns:  
 tuple: A tuple containing personal information and employee-specific details  
 (name, Emirates ID, phone, insurance, date of birth, department, room).  
 """* return super().display\_person\_info(), self.\_department, self.\_room  
  
 def display\_employee\_info(self):  
 *"""  
 Generate a formatted string representing an employee's information.  
  
 Returns:  
 str: A formatted string containing the employee's information.  
 """* return f'{super().display\_person\_info()}, Department: {self.\_department}, Room: {self.\_room}'  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of an employee.  
  
 Returns:  
 str: A string containing the employee's information.  
 """* person\_info, department, room = self.get\_employee\_info()  
 return f'{str(person\_info)}, Department: {str(department)}, Room: {str(room)}'  
  
 class Doctor(Employee):  
 *"""  
 Represents a doctor with additional information about their medical field.  
 """* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.\_doctor\_field = ""  
  
 def set\_doctor\_info(self, fullname, emirates\_id, phone, insurance, dateofbirth, department, room, doctor\_field):  
 *"""  
 Set the information of a doctor, including personal information, department, room, and medical field.  
  
 Args:  
 fullname (str): The full name of the doctor.  
 emirates\_id (str): The Emirates ID of the doctor.  
 phone (str): The phone number of the doctor.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the doctor.  
 dateofbirth (str): The date of birth of the doctor in "YYYY-MM-DD" format.  
 department (str): The department where the doctor works.  
 room (str): The room where the doctor is assigned.  
 doctor\_field (str): The medical field of the doctor.  
 """* super().set\_employee\_info(fullname, emirates\_id, phone, insurance, dateofbirth, department, room)  
 self.\_doctor\_field = doctor\_field  
  
 def input\_doct\_info(self):  
 *"""  
 Prompt the user to input information about a doctor.  
 """* fullname = input("Enter Doctor's Full Name: ")  
 emirates\_id = input("Enter Doctor's Emirates ID: ")  
 phone = input("Enter Doctor's Phone number: ")  
 insurance = input("Enter Doctor's Insurance type (None/Gold/Platinum): ")  
 dateofbirth = input("Enter Doctor's Date of Birth (YYYY-MM-DD): ")  
 department = input("Enter Doctor's Department: ")  
 room = input("Enter Doctor's Operating Room: ")  
 doctor\_field = input("Enter Doctor's Field: ")  
  
 self.set\_doctor\_info(fullname, emirates\_id, phone, insurance, dateofbirth, department, room, doctor\_field)  
  
 def get\_doctor\_info(self):  
 *"""  
 Get the doctor's information, including personal information, department, room, and medical field.  
  
 Returns:  
 tuple: A tuple containing doctor's information (name, Emirates ID, phone, insurance, date of birth,  
 department, room, doctor field).  
 """* return super().get\_employee\_info(), self.\_doctor\_field  
  
 def display\_doctor\_info(self):  
 *"""  
 Generate a formatted string representing a doctor's information.  
  
 Returns:  
 str: A formatted string containing the doctor's information.  
 """* return f'{super().display\_employee\_info()}, Doctor Field: {self.\_doctor\_field}'  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of a doctor.  
  
 Returns:  
 str: A string containing the doctor's information.  
 """* return f'{str(super().display\_employee\_info())}, Doctor Field: {str(self.\_doctor\_field)}'  
  
 class Nurse(Employee):  
 *"""  
 Represents a nurse with additional information about their field of work.  
 """* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.\_nurse\_work = ""  
  
 def set\_nurse\_info(self, fullname, emirates\_id, phone, insurance, dateofbirth, department, room, nurse\_work):  
 *"""  
 Set the information of a nurse, including personal information, department, room, and nurse field.  
  
 Args:  
 fullname (str): The full name of the nurse.  
 emirates\_id (str): The Emirates ID of the nurse.  
 phone (str): The phone number of the nurse.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the nurse.  
 dateofbirth (str): The date of birth of the nurse in "YYYY-MM-DD" format.  
 department (str): The department where the nurse works.  
 room (str): The room where the nurse is assigned.  
 nurse\_work (str): The field of work for the nurse.  
 """* super().set\_employee\_info(fullname, emirates\_id, phone, insurance, dateofbirth, department, room)  
 self.\_nurse\_work = nurse\_work  
  
 def input\_nurse\_info(self):  
 *"""  
 Prompt the user to input information about a nurse.  
 """* fullname = input("Enter Nurse's Full Name: ")  
 emirates\_id = input("Enter Nurse's Emirates ID: ")  
 phone = input("Enter Nurse's Phone number: ")  
 insurance = input("Enter Nurse's Insurance type (None/Gold/Platinum): ")  
 dateofbirth = input("Enter Nurse's Date of Birth (YYYY-MM-DD): ")  
 department = input("Enter Nurse's Department: ")  
 room = input("Enter Nurse's Operating Room: ")  
 nurse\_work = input("Enter Nurse's Field: ")  
  
 self.set\_nurse\_info(fullname, emirates\_id, phone, insurance, dateofbirth, department, room, nurse\_work)  
  
 def get\_nurse\_info(self):  
 *"""  
 Get the nurse's information, including personal information, department, room, and nurse field.  
  
 Returns:  
 tuple: A tuple containing nurse's information (name, Emirates ID, phone, insurance, date of birth,  
 department, room, nurse field).  
 """* return super().get\_employee\_info(), self.\_nurse\_work  
  
 def display\_nurse\_info(self):  
 *"""  
 Generate a formatted string representing a nurse's information.  
  
 Returns:  
 str: A formatted string containing the nurse's information.  
 """* return f'{super().display\_employee\_info()}, Nurse Field: {self.\_nurse\_work}'  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of a nurse.  
  
 Returns:  
 str: A string containing the nurse's information.  
 """* return f'{str(super().display\_employee\_info())}, Nurse Field: {str(self.\_nurse\_work)}'  
  
 class Patient(Person):  
 *"""  
 Represents a patient with information about their visit and medical history.  
 """* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.\_visit = None  
 self.\_arriving = None  
 self.\_disease = None  
 self.\_checkout = None  
 self.patient\_file = {}  
  
 def set\_patient\_info(self, fullname, emirates\_id, phone, insurance, dateofbirth, visit, arriving, disease,  
 checkout):  
 *"""  
 Set the information of a patient, including personal information, visit details, and medical history.  
  
 Args:  
 fullname (str): The full name of the patient.  
 emirates\_id (str): The Emirates ID of the patient.  
 phone (str): The phone number of the patient.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the patient.  
 dateofbirth (str): The date of birth of the patient in "YYYY-MM-DD" format.  
 visit (str): The reason for the patient's visit.  
 arriving (str): The date and time of the patient's arrival in "YYYY-MM-DD HH:MM:SS" format.  
 disease (str): Description of the patient's disease.  
 checkout (str): The date and time of the patient's checkout in "YYYY-MM-DD HH:MM:SS" format.  
 """* patient\_dict = self.patient\_file  
 if emirates\_id in patient\_dict:  
 existing\_patient\_info = patient\_dict[emirates\_id]["patient\_info"]  
 print(f"Patient already exists:\n{existing\_patient\_info}")  
 else:  
 super().set\_person\_info(fullname, emirates\_id, phone, insurance, dateofbirth)  
 self.\_visit = visit  
 self.\_arriving = arriving if arriving is not None else datetime.datetime.now()  
 self.\_disease = disease  
 self.\_checkout = checkout if checkout is not None else datetime.datetime.now()  
  
 patient\_number = len(patient\_dict) + 1  
  
 patient\_dict[emirates\_id] = {  
 "patient\_number": patient\_number,  
 "patient\_info": self.get\_patient\_info()  
 }  
  
 def input\_patient\_info():  
 *"""  
 Prompt the user to input information about a patient.  
  
 Returns:  
 tuple: A tuple containing patient information (name, Emirates ID, phone, insurance, date of birth, visit,  
 arriving, disease, checkout).  
 """* fullname = input("Enter Patient's Full Name: ")  
 emirates\_id = input("Enter Patient's Emirates ID: ")  
 phone = input("Enter Patient's Phone Number: ")  
 insurance = input("Enter Patient's Insurance Type (None/Gold/Platinum): ")  
 dateofbirth = input("Enter Patient's Date of Birth (YYYY-MM-DD): ")  
 visit = input("Enter Patient's Visit Reason: ")  
 arriving = input("Enter Patient's Arriving Date and Time (YYYY-MM-DD HH:MM:SS): ")  
 disease = input("Enter Patient's Disease Description: ")  
 checkout = input("Enter Patient's Checkout Date and Time (YYYY-MM-DD HH:MM:SS): ")  
  
 try:  
 arriving = datetime.datetime.strptime(arriving, "%Y-%m-%d %H:%M:%S")  
 except ValueError:  
 arriving = datetime.datetime.now()  
  
 try:  
 checkout = datetime.datetime.strptime(checkout, "%Y-%m-%d %H:%M:%S")  
 except ValueError:  
 checkout = datetime.datetime.now()  
  
 return fullname, emirates\_id, phone, insurance, dateofbirth, visit, arriving, disease, checkout  
  
 def get\_patient\_info(self):  
 *"""  
 Get the patient's information, including personal information, visit details, and medical history.  
  
 Returns:  
 tuple: A tuple containing patient information (name, Emirates ID, phone, insurance, date of birth, visit,  
 arriving, disease, checkout).  
 """* return (  
 super().get\_person\_info(),  
 self.\_visit,  
 self.\_arriving,  
 self.\_disease,  
 self.\_checkout  
 )  
  
 def display\_patient\_info(self):  
 *"""  
 Generate a formatted string representing a patient's information.  
  
 Returns:  
 str: A formatted string containing the patient's information.  
 """* return f"{super().display\_person\_info()}, Visit: {self.\_visit}, Arriving: {self.\_arriving}, " \  
 f"Sickness: {self.\_disease}, Checkout: {self.\_checkout}"  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of a patient.  
  
 Returns:  
 str: A string containing the patient's information.  
 """* return self.display\_patient\_info()  
  
 class MedicalBilling(Patient):  
 *"""  
 Represents a patient's medical billing information.  
 """* def \_\_init\_\_(self):  
 super().\_\_init\_\_()  
 self.\_doctor = Hospital.Doctor() # Initialize doctor to None initially  
 self.\_nurse = Hospital.Nurse() # Initialize nurse to None initially  
 self.\_doctor\_Consultation = 0  
 self.\_nursing\_Services = 0  
 self.\_hospital\_Charges = 0  
 self.\_pharmacy = 0  
 self.\_room\_charges = 0  
 self.\_tax = 0  
 self.\_insurance\_Cover = None  
  
 def input\_medical\_billing(self):  
 *"""  
 Prompt the user to input medical billing information for a patient.  
 """* fullname = input("Enter Patient Last Name: ")  
 emirates\_id = input("Enter Patient Emirates ID: ")  
 phone = input("Enter Patient Phone number: ")  
 insurance = input("Enter Patient Insurance type (None/Gold/Platinum): ")  
 date\_of\_birth = input("Enter Patient Date of Birth (YYYY-MM-DD): ")  
 arriving = input("Enter Patient arriving date and time (YYYY-MM-DD HH:MM:SS): ")  
 visit = input("Enter Patient visit reason: ")  
 disease = input("Enter Patient disease description: ")  
 checkout = input("Enter Patient checkout date and time (YYYY-MM-DD HH:MM:SS): ")  
 consultation = float(input("Enter Patient consultation charges: "))  
 n\_service = float(input("Enter Patient nursing services charges: "))  
 hospital\_charges = float(input("Enter Patient hospital charges: "))  
 pharmacy = float(input("Enter Patient pharmacy charges: "))  
 room\_charges = float(input("Enter Patient room charges: "))  
 tax\_charge = float(input("Enter tax rate (e.g., 0.05 for 5%): "))  
  
 self.set\_medicalbilling\_info(  
 fullname, emirates\_id, phone, insurance,  
 date\_of\_birth, arriving, visit, disease, checkout,  
 consultation, n\_service, hospital\_charges, pharmacy, room\_charges, tax\_charge  
 )  
  
 def get\_insurance\_cover(self):  
 *"""  
 Get the insurance coverage type of the patient.  
  
 Returns:  
 str: The type of insurance coverage (None, Gold, Platinum).  
 """* \_, \_, \_, insurance, \_ = super().get\_person\_info()  
 return insurance  
  
 def set\_medicalbilling\_info(self, fullname, emirates\_id, phone,  
 insurance, date\_of\_birth, arriving, visit, disease, checkout, doctor1, nurse1,  
 consultation,  
 n\_service,  
 hospital\_charges, pharmacy, room\_charges, tax\_charge):  
 *"""  
 Set the medical billing information for a patient.  
  
 Args:  
 fullname (str): The full name of the patient.  
 emirates\_id (str): The Emirates ID of the patient.  
 phone (str): The phone number of the patient.  
 insurance (str): The type of insurance (None, Gold, Platinum) of the patient.  
 date\_of\_birth (str): The date of birth of the patient in "YYYY-MM-DD" format.  
 arriving (str): The date and time of the patient's arrival in "YYYY-MM-DD HH:MM:SS" format.  
 visit (str): The reason for the patient's visit.  
 disease (str): Description of the patient's disease.  
 checkout (str): The date and time of the patient's checkout in "YYYY-MM-DD HH:MM:SS" format.  
 doctor1 (Doctor): The doctor assigned to the patient.  
 nurse1 (Nurse): The nurse assigned to the patient.  
 consultation (float): The consultation charges for the patient.  
 n\_service (float): The nursing services charges for the patient.  
 hospital\_charges (float): The hospital charges for the patient.  
 pharmacy (float): The pharmacy charges for the patient.  
 room\_charges (float): The room charges for the patient.  
 tax\_charge (float): The tax rate (e.g., 0.05 for 5%) for the patient's bill.  
 """* super().set\_patient\_info(fullname, emirates\_id, phone, insurance, date\_of\_birth, visit, arriving, disease,  
 checkout)  
 self.\_doctor = doctor1  
 self.\_nurse = nurse1  
 self.\_\_doctor\_Consultation = consultation  
 self.\_\_nursing\_Services = n\_service  
 self.\_\_hospital\_Charges = hospital\_charges  
 self.\_\_pharmacy = pharmacy  
 self.\_\_room\_charges = room\_charges  
 self.\_\_tax = tax\_charge  
 self.\_\_insurance\_Cover = self.get\_insurance\_cover()  
  
 def display\_medicalbilling\_info(self):  
 *"""  
 Generate a formatted string representing a patient's medical billing information.  
  
 Returns:  
 str: A formatted string containing the patient's medical billing information.  
 """* patient\_info = super().display\_patient\_info()  
 doctor\_info = str(self.\_doctor)  
 nurse\_info = str(self.\_nurse)  
 consultation\_charge = self.\_doctor\_Consultation  
 nursing\_services\_charge = self.\_nursing\_Services  
 hospital\_charge = self.\_hospital\_Charges  
 pharmacy\_charge = self.\_pharmacy  
 room\_charge = self.\_room\_charges  
 tax\_rate = self.\_tax  
  
 total\_charges = (  
 consultation\_charge + nursing\_services\_charge + hospital\_charge +  
 pharmacy\_charge + room\_charge  
 )  
  
 # Determine coverage based on insurance type  
 if self.\_insurance\_Cover == "Platinum":  
 total\_charges = 0 # Full coverage, no charges  
 self.\_insurance\_Cover = "Full"  
 elif self.\_insurance\_Cover == "Gold":  
 total\_charges \*= 0.5 # Half coverage, 50% of billing charges  
 self.\_insurance\_Cover = "Half"  
 elif self.\_insurance\_Cover == "None":  
 tax\_amount = total\_charges \* self.\_\_tax  
 total\_charges += tax\_amount  
 self.\_insurance\_Cover = "None"  
  
 doctor\_info = self.\_doctor.display\_doctor\_info()  
 name\_start\_index = doctor\_info.find("Full Name ") + len("Full Name ")  
 name\_end\_index = doctor\_info.find(", Emirates ID:")  
 doctor\_name = doctor\_info[name\_start\_index:name\_end\_index]  
  
 nurse\_info = self.\_nurse.display\_nurse\_info()  
 name\_start\_indexn = nurse\_info.find("Full Name ") + len("Full Name ")  
 name\_end\_indexn = doctor\_info.find(", Emirates ID:")  
 nurse\_name = nurse\_info[name\_start\_indexn:name\_end\_indexn]  
  
 # Create a table with the medical billing information  
 table = [  
 ["Patient Info", "Doctor Info", "Nurse Info", "Consultation Charge (AED)",  
 "Nursing Services Charge (AED)",  
 "Hospital Charge (AED)", "Pharmacy Charge (AED)", "Room Charge (AED)", "Tax", "Insurance Coverage",  
 "Total Charges (AED)"],  
 [super().display\_patient\_info(), doctor\_name, nurse\_name,  
 consultation\_charge, nursing\_services\_charge, hospital\_charge, pharmacy\_charge, room\_charge,  
 self.\_\_tax,  
 self.\_\_insurance\_Cover, total\_charges]  
 ]  
  
 # Format the table using tabulate  
 table\_str = tabulate(table, headers="firstrow", tablefmt="grid")  
  
 return f"{patient\_info}\n{table\_str}"  
  
 def \_\_str\_\_(self):  
 *"""  
 Generate a string representation of a patient's medical billing information.  
  
 Returns:  
 str: A string containing the patient's medical billing information.  
 """* return self.display\_medicalbilling\_info()  
# Create instances of Hospital classes  
hospital = Hospital()  
doctor = Hospital.Doctor()  
nurse = Hospital.Nurse()  
patient = Hospital.Patient()  
medical\_billing = Hospital.MedicalBilling()  
  
# Create instances of Hospital classes  
hospital = Hospital()  
  
# Create a Doctor instance  
doctor.set\_doctor\_info("Dr. John Doe", "1234567890", "555-555-5555", "Gold", "1980-01-01", "Cardiology", "Room 101",  
 "Cardiologist")  
  
# Create a Nurse instance  
nurse.set\_nurse\_info("Nurse Jane Smith", "9876543210", "555-123-4567", "Platinum", "1990-05-15", "Emergency",  
 "Room 201", "Emergency Nurse")  
  
# Create a Patient instance  
patient.set\_patient\_info("Alice Johnson", "111122223333", "555-789-1234", "None", "1995-12-30", "Checkup",  
 "2023-10-08 08:30:00", "Fever", "2023-10-08 10:45:00")  
  
# Create a MedicalBilling instance  
medical\_billing.set\_medicalbilling\_info("Bob Williams", "444455556666", "555-987-6543", "Platinum", "1988-06-20",  
 "Surgery", "2023-10-08 12:00:00", "Appendectomy", "2023-10-08 14:30:00", doctor,  
 nurse, 300.0, 150.0, 1000.0, 200.0, 500.0, 0.05)  
  
# Display information  
print("Doctor Information:")  
print(doctor)  
  
print("\nNurse Information:")  
print(nurse)  
  
print("\nPatient Information:")  
print(patient)  
  
print("\nMedical Billing Information:")  
print(medical\_billing.display\_medicalbilling\_info())  
hospital = Hospital()  
  
# Create a Person object  
person1 = hospital.Person()  
  
# Set personal information for person1 using the set\_person\_info method  
person1.set\_person\_info("John Doe", "1234567890", "555-555-5555", "Gold", "1980-01-01")  
  
# Get personal information for person1 using the get\_person\_info method  
info1 = person1.get\_person\_info()  
  
# Display information about person1 using the display\_person\_info method  
print("Information about Person 1:")  
print(person1.display\_person\_info())  
  
# Display information about person1 using the \_\_str\_\_ method  
print("\nInformation about Person 1 (Using \_\_str\_\_ method):")  
print(person1)  
  
# Create an Employee object  
employee1 = Hospital.Employee()  
  
# Set the information for the employee  
employee1.set\_employee\_info(  
 "John Smith",  
 "1234567890",  
 "555-555-5555",  
 "Gold",  
 "1985-05-15",  
 "Cardiology",  
 "Room 101"  
)  
  
# Get the employee information using the get\_employee\_info method  
employee\_info = employee1.get\_employee\_info()  
print(employee\_info)  
# Create a Doctor object  
doctor1 = Hospital.Doctor()  
  
# Use the input\_doct\_info method to input information about the doctor  
doctor1.input\_doct\_info()  
  
# Use the display\_doctor\_info method to display the doctor's information  
print("\nDoctor Information:")  
print(doctor1.display\_doctor\_info())  
  
# Use the get\_doctor\_info method to get and display the doctor's information  
doctor\_info = doctor1.get\_doctor\_info()  
print("\nDoctor Information (Tuple):")  
print(doctor\_info)  
  
# Use the \_\_str\_\_ method to get a string representation of the doctor  
print("\nDoctor String Representation:")  
print(str(doctor1))  
  
nurse1 = Hospital.Doctor()  
  
# Use the input\_doct\_info method to input information about the doctor  
nurse1.input\_doct\_info()  
  
# Use the display\_doctor\_info method to display the doctor's information  
print("\nDoctor Information:")  
print(doctor1.display\_nurse\_info())  
  
# Use the get\_doctor\_info method to get and display the doctor's information  
doctor\_info = doctor1.get\_nurse\_info()  
print("\nDoctor Information (Tuple):")  
print(doctor\_info)  
  
# Use the \_\_str\_\_ method to get a string representation of the doctor  
print("\nDoctor String Representation:")  
print(str(nurse1))

TESTING EXAMPLES :

A screenshot of a computer program

Description automatically generatedA close-up of a computer screen

Description automatically generated

Summary:

I learned new skill by defining several classes, including Person, Employee, Doctor, Nurse, Patient, and MedicalBilling, each of which inherits and extends the functionality of the preceding one. This framework helped facilitates the creation, storage, and retrieval of exact information about hospital personnel, including physicians and nurses, as well as patients. Notably, the MedicalBilling class computes and presents realistic medical billing data, taking into account characteristics like as consultation prices, insurance coverage, and tax rates. Overall, this code exhibits strong object-oriented programming concepts, allowing for the methodical and modular organization and handling of hospital-related data.

So