1 Hospital Management System

```
from abc import ABC, abstractmethod



class staff(ABC):
    def __init__(self, name, age, department):
        self.name = name
        self.age = age
        self.department = department


    @abstractmethod
    def get_details(self):
        pass



class doctor(staff):
    def __init__(self, name, age, department, specialization,
patients_assigned=None):
        super().__init__(name, age, department)
        self.specialization = specialization
        self.patients_assigned = patients_assigned if
patients_assigned is not None else []


    def get_details(self):
```

```
        print(f"name: {self.name}")
        print(f"age: {self.age}")
        print(f"department: {self.department}")
        print(f"specialization: {self.specialization}")
        if self.patients_assigned:
            print("patients assigned:")
            for patient in self.patients_assigned:
                print(patient)
        else:
            print("no patients assigned.")



class nurse(staff):
    def __init__(self, name, age, department, shift,
wards_assigned=None):
        super().__init__(name, age, department)
        self.shift = shift
        self.wards_assigned = wards_assigned if wards_assigned is
not None else []



    def get_details(self):
        print(f"name: {self.name}")
        print(f"age: {self.age}")
        print(f"department: {self.department}")
        print(f"shift: {self.shift}")
        if self.wards_assigned:
```

```
            print("wards assigned:")
            for ward in self.wards_assigned:
                print(ward)
        else:
            print("no wards assigned.")



class administrative_staff(staff):
    def __init__(self, name, age, department, designation=None):
        super().__init__(name, age, department)
        self.designation = designation if designation is not None
else ""


    def get_details(self):
        print(f"name: {self.name}")
        print(f"age: {self.age}")
        print(f"department: {self.department}")
        print(f"designation: {self.designation}")



doctor = doctor("bishwajit", 22, "medicine", "cardiology")
doctor.get_details()


print("\n")
```

```python
nurse = nurse("urmi", 23, "nursing", "day shift", ["ward a", "ward b"])

nurse.get_details()



print("\n")



administrative_staff = administrative_staff("shah alom", 25, "hr", "manager")

administrative_staff.get_details()
```

```
bk  .../Python  master ?  v3.12.7  23:31
→ python -u "/home/bk/code/Python/lab_report_7/HMS.py"
name: bishwajit
age: 22
department: medicine
specialization: cardiology
no patients assigned.


name: urmi
age: 23
department: nursing
shift: day shift
wards assigned:
ward a
ward b


name: shah alom
age: 25
department: hr
designation: manager

bk  .../Python  master ?  v3.12.7  23:31
→
```

2 Hospital System

```python
class Patient:
    def __init__(self, name, age):
        self.name = name
        self.age = age
        self.admission_date = None
        self.room_number = None
        self.appointment_time = None


    def get_patient_info(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age}")
        if self.admission_date:
            print(f"Admission Date: {self.admission_date}")
        if self.room_number:
            print(f"Room Number: {self.room_number}")
        if self.appointment_time:
            print(f"Appointment Time: {self.appointment_time}")



class InPatient(Patient):
    def __init__(self, name, age, admission_date, room_number):
        super().__init__(name, age)
        self.admission_date = admission_date
```

```python
        self.room_number = room_number


    def get_patient_info(self):
        super().get_patient_info()
        print(f"Room Number: {self.room_number}")



class OutPatient(Patient):
    def __init__(self, name, age, appointment_time):
        super().__init__(name, age)
        self.appointment_time = appointment_time


    def get_patient_info(self):
        super().get_patient_info()
        print(f"Appointment Time: {self.appointment_time}")



inpatient1 = InPatient("Bishwajit", 22, "2022-01-01", "Room 101")
outpatient1 = OutPatient("Urmi", 23, "12:00 AM")


inpatient2 = InPatient("Shah alom", 25, "2022-02-15", "Room 202")
outpatient2 = OutPatient("Likhon", 20, "06:00 PM")
```

```python
inpatient1.get_patient_info()
print("\n")
outpatient1.get_patient_info()



print("\n")



inpatient2.get_patient_info()
print("\n")
outpatient2.get_patient_info()
```

```
bk  .../Python  master ?  v3.12.7  23:31
→ python -u "/home/bk/code/Python/lab_report_7/Patient.py"
Name: Bishwajit
Age: 22
Admission Date: 2022-01-01
Room Number: Room 101
Room Number: Room 101


Name: Urmi
Age: 23
Appointment Time: 12:00 AM
Appointment Time: 12:00 AM


Name: Shah alom
Age: 25
Admission Date: 2022-02-15
Room Number: Room 202
Room Number: Room 202


Name: Likhon
Age: 20
Appointment Time: 06:00 PM
Appointment Time: 06:00 PM

bk  .../Python  master ?  v3.12.7  23:36
→
```

3 Bank Management System

```python
from abc import ABC, abstractmethod



class BankAccount(ABC):
    def __init__(self, account_number, initial_balance=0):
        self.account_number = account_number
        self.balance = initial_balance


    @abstractmethod
    def deposit(self, amount):
        pass


    @abstractmethod
    def withdraw(self, amount):
        pass



class SavingsAccount(BankAccount):
    def __init__(self, account_number, initial_balance=0,
interest_rate=0.05):
        super().__init__(account_number, initial_balance)
        self.interest_rate = interest_rate
```

```
    def deposit(self, amount):
        self.balance += amount
        print(f"Deposited Tk. {amount:.2f} into Savings Account
{self.account_number}")
        return self.balance



    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance in Savings Account")
            return None
        self.balance -= amount
        print(f"Withdrew Tk. {amount:.2f} from Savings Account
{self.account_number}")
        return self.balance



    def apply_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        print(f"Applied interest of Tk. {interest:.2f} to Savings
Account {self.account_number}")



class CheckingAccount(BankAccount):
    def __init__(self, account_number, initial_balance=0,
transaction_limit=1000):
        super().__init__(account_number, initial_balance)
```

```
        self.transaction_limit = transaction_limit



    def deposit(self, amount):
        if amount > 0:
            print(f"Deposited Tk. {amount:.2f} into Checking
Account {self.account_number}")
            return self.balance + amount
        else:
            raise ValueError("Deposit amount must be positive")



    def withdraw(self, amount):
        if amount <= self.transaction_limit:
            if amount > 0:
                print(f"Withdrew Tk. {amount:.2f} from Checking
Account {self.account_number}")
                return self.balance - amount
            elif amount == 0:
                raise ValueError("Withdrawal amount cannot be
zero")
            else:
                print("Transaction limit exceeded for Checking
Account")
                return None
        else:
            raise ValueError(f"Transaction limit exceeded for
Checking Account")
```

```python
savings_account = SavingsAccount(9999, 1111)
print(savings_account.balance)

checking_account = CheckingAccount(1111, 111)
print(checking_account.balance)

checking_account.withdraw(111)
print(checking_account.balance)

savings_account.apply_interest()
print(savings_account.balance)
```

```
Withdrew Tk. 999 from Business Account 999991

bk  .../Python  master ?  v3.12.7  23:41
→ python -u "/home/bk/code/Python/lab_report_7/BMS.py"
1111
111
Withdrew Tk. 111.00 from Checking Account 1111
111
Applied interest of Tk. 55.55 to Savings Account 9999
1166.55

bk  .../Python  master ?  v3.12.7  23:43
→
```

4 Student Bank Management System

```python
from abc import ABC, abstractmethod



class BankAccount(ABC):
    def __init__(self, account_number, initial_balance=0):
        self.account_number = account_number
        self.balance = initial_balance


    @abstractmethod
    def withdraw(self, amount):
        pass



class StudentAccount(BankAccount):
    def __init__(self, account_number, initial_balance=0,
student_id=None):
        super().__init__(account_number, initial_balance)
        self.student_id = student_id


    def withdraw(self, amount):
        if amount > self.balance:
            print("Insufficient balance in Student Account")
            raise ValueError("Insufficient balance")
        elif self.balance < 100:
```

```
            print(f"No fee for withdrawal of Tk. {amount} from
Student Account {self.account_number}")

        else:

            self.balance -= amount

            print(f"Withdrew Tk. {amount} from Student Account
{self.account_number}")




class BusinessAccount(BankAccount):

    def __init__(self, account_number, initial_balance=0,
overdraft_limit=500):

        super().__init__(account_number, initial_balance)

        self.overdraft_limit = overdraft_limit



    def withdraw(self, amount):

        if amount > self.balance + self.overdraft_limit:

            print(f"Overdraft limit exceeded for Business Account
{self.account_number}. Available balance: Tk. {self.balance}")

            raise ValueError("Overdraft limit exceeded")

        elif amount <= 0:

            print(f"No withdrawal allowed for Business Account
{self.account_number}")

        else:

            if self.balance + self.overdraft_limit < amount:

                overdraft_amount = amount - (self.balance +
self.overdraft_limit)

                self.balance -= self.overdraft_limit
```

```
                print(f"Withdrew Tk. {self.overdraft_limit} from
overdraft for Business Account {self.account_number}")
            else:
                self.balance -= amount
                print(f"Withdrew Tk. {amount} from Business Account
{self.account_number}")



student_account = StudentAccount(999990, 9999)
print(student_account.balance)


business_account = BusinessAccount(999991, 9999, 99)
print(business_account.balance)


student_account.withdraw(999)
print(student_account.balance)


business_account.withdraw(999)
print(business_account.balance)


business_account.withdraw(999)
```

```
bk  .../Python  master ?  v3.12.7  23:36
→ python -u "/home/bk/code/Python/lab_report_7/Student_acc.py"
9999
9999
Withdrew Tk. 999 from Student Account 999990
9000
Withdrew Tk. 999 from Business Account 999991
9000
Withdrew Tk. 999 from Business Account 999991

bk  .../Python  master ?  v3.12.7  23:41
→
```

5 Student Management System

```python
from abc import ABC, abstractmethod




class Student(ABC):
    def __init__(self, name, student_id, email):
        self.name = name
        self.student_id = student_id
        self.email = email



    @abstractmethod
    def get_info(self):
        pass



class Undergraduate(Student):
```

```python
    def __init__(self, name, student_id, email, year):
        super().__init__(name, student_id, email)
        self.year = year



    def get_info(self):
        return f"Name: {self.name}\nStudent ID:
{self.student_id}\nEmail: {self.email}\nYear: {self.year}"




class Graduate(Student):
    def __init__(self, name, student_id, email, research_topic):
        super().__init__(name, student_id, email)
        self.research_topic = research_topic



    def get_info(self):
        return f"Name: {self.name}\nStudent ID:
{self.student_id}\nEmail: {self.email}\nResearch Topic:
{self.research_topic}"



def print_student_info(students):
    for student in students:
        if isinstance(student, Undergraduate):
            print(f"Undergraduate Student\n{student.get_info()}")
        elif isinstance(student, Graduate):
```

```
            print(f"Graduate Student\n{student.get_info()}")
```

```
undergraduate = Undergraduate("Bishwajit", "1414",
"bishwajit@gmail.com","2026")

print(undergraduate.get_info())
```

```
graduate = Graduate("Shah Alom", "1409", "shahalam@gmail.com",
"Machine Learning for Natural Language Processing")

print(graduate.get_info())
```

```
students = [undergraduate, graduate]

print_student_info(students)
```

```
bk  .../Python  master ?  v3.12.7  23:43
→ python -u "/home/bk/code/Python/lab_report_7/SMS.py"
Name: Bishwajit
Student ID: 1414
Email: bishwajit@gmail.com
Year: 2026
Name: Shah Alom
Student ID: 1409
Email: shahalam@gmail.com
Research Topic: Machine Learning for Natural Language Processing
Undergraduate Student
Name: Bishwajit
Student ID: 1414
Email: bishwajit@gmail.com
Year: 2026
Graduate Student
Name: Shah Alom
Student ID: 1409
Email: shahalam@gmail.com
Research Topic: Machine Learning for Natural Language Processing

bk  .../Python  master ?  v3.12.7  23:44
→
```

6 University Management System

```python
class Student:

    def __init__(self, name, student_id, email, major):

        self.name = name

        self.student_id = student_id

        self.email = email

        self.major = major



    def get_details(self):

        details = f"Name: {self.name}\nStudent ID:
{self.student_id}\nEmail: {self.email}\nMajor: {self.major}"

        return details
```

```python
class RegularStudent(Student):
    def __init__(self, name, student_id, email, major):
        super().__init__(name, student_id, email, major)


    def get_details1(self):
        return self.get_details()


class ExchangeStudent(Student):
    def __init__(self, name, student_id, email, major,
home_university):
        super().__init__(name, student_id, email, major)
        self.home_university = home_university


    def get_details1(self):
        details = f"{super().get_details()}\nHome University:
{self.home_university}"
        return details


def print_student_details(students):
    for student in students:
        print(student.get_details())
```

```python
        print("-" * 30)




regular_student = RegularStudent("Bishwajit", "1414",
"bishwajit@gmail.com", "Computer Science")

exchange_student = ExchangeStudent("Shah Alom", "1409",
"shahalam@gmail.com", "Mechanical Engineering", "University of
California, Berkeley")



students = [regular_student, exchange_student]

print_student_details(students)
```

```
bk  .../Python  master ?  v3.12.7  23:44
→ python -u "/home/bk/code/Python/lab_report_7/UMS.py"
Name: Bishwajit
Student ID: 1414
Email: bishwajit@gmail.com
Major: Computer Science
------------------------------
Name: Shah Alom
Student ID: 1409
Email: shahalam@gmail.com
Major: Mechanical Engineering
------------------------------

bk  .../Python  master ?  v3.12.7  23:46
→
```

7 Library Management System

```python
from abc import ABC, abstractmethod



class LibraryItem(ABC):
    def __init__(self, title, author, publication_year):
        self.title = title
        self.author = author
        self.publication_year = publication_year


    @abstractmethod
    def display_info(self):
        pass



class Book(LibraryItem):
    def __init__(self, title, author, publication_year, genre):
        super().__init__(title, author, publication_year)
        self.genre = genre


    def display_info(self):
        info = f"Title: {self.title}\nAuthor:
{self.author}\nPublication Year: {self.publication_year}\nGenre:
{self.genre}"
        return info
```

```python
class Magazine(LibraryItem):
    def __init__(self, title, author, publication_year, issue_number):
        super().__init__(title, author, publication_year)
        self.issue_number = issue_number


    def display_info(self):
        info = f"Title: {self.title}\nAuthor: {self.author}\nPublication Year: {self.publication_year}\nIssue Number: {self.issue_number}"
        return info


class DVD(LibraryItem):
    def __init__(self, title, author, publication_year, duration):
        super().__init__(title, author, publication_year)
        self.duration = duration


    def display_info(self):
        info = f"Title: {self.title}\nAuthor: {self.author}\nPublication Year: {self.publication_year}\nDuration: {self.duration} minutes"
        return info
```

```python
class Library:
    def __init__(self):
        self.items = []


    def add_item(self, item):
        self.items.append(item)


    def display_all_items(self):
        for item in self.items:
            print(f"--- {type(item).__name__}:")
            print(item.display_info())
            print()





library = Library()



book = Book("The Silent Patient", "Alex Michaelides", 2019,
"Psychological Thriller")
magazine = Magazine("Nelolian: The Investigation", "Sharlok Homes",
2022, 3)
dvd = DVD("Crime Scene: The Movie", "Frank Darabont", 1994, 132)
```

```python
library.add_item(book)
library.add_item(magazine)
library.add_item(dvd)


library.display_all_items()
```

```
bk  .../Python  master ?  v3.12.7  23:46
→ python -u "/home/bk/code/Python/lab_report_7/LMS.py"
--- Book:
Title: The Silent Patient
Author: Alex Michaelides
Publication Year: 2019
Genre: Psychological Thriller

--- Magazine:
Title: Nelolian: The Investigation
Author: Sharlok Homes
Publication Year: 2022
Issue Number: 3

--- DVD:
Title: Crime Scene: The Movie
Author: Frank Darabont
Publication Year: 1994
Duration: 132 minutes

bk  .../Python  master ?  v3.12.7  23:47
→
```

8 Banking Account System

```python
class BankAccount:
    def __init__(self, account_number, balance=0):
        self.account_number = account_number
        self.balance = balance


    def deposit(self, amount):
        if amount > 0:
            self.balance += amount
            return f"Deposited Tk. {amount}. New Balance: Tk. {self.balance}"
        else:
            return "Invalid deposit amount."


    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            return f"Withdrew Tk. {amount}. New Balance: Tk. {self.balance}"
        elif amount <= 0:
            return "Invalid withdrawal amount."
        else:
            return "Insufficient funds."


class LoanAccount:
```

```python
    def __init__(self, loan_number, balance=0, interest_rate=0.05):
        self.loan_number = loan_number
        self.balance = balance
        self.interest_rate = interest_rate


    def make_payment(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            return f"Paid Tk. {amount}. Remaining Balance: Tk. {self.balance}"
        elif amount <= 0:
            return "Invalid payment amount."
        else:
            return "Payment exceeds loan balance."


    def calculate_interest(self):
        interest = self.balance * self.interest_rate
        self.balance += interest
        return f"Interest calculated: +Tk. {interest}. New Balance: Tk. {self.balance}"


class CustomerAccount(BankAccount, LoanAccount):
    def __init__(self, account_number, customer_name, loan_number=None, balance=0, interest_rate=0.05):
        super().__init__(account_number)
```

```
        self.customer_name = customer_name
        if loan_number is not None:
            self.loan = LoanAccount(loan_number, balance,
interest_rate)



    def display_customer_info(self):
        return f"Customer Name: {self.customer_name}\nAccount
Number: {self.account_number}"



    def make_transaction(self, transaction_type, amount):
        if transaction_type == "deposit":
            return self.deposit(amount)
        elif transaction_type == "withdrawal":
            return self.withdraw(amount)
        else:
            raise ValueError("Invalid transaction type.")



    def make_loan_payment(self, payment_amount):
        if isinstance(self.loan, LoanAccount):
            return self.loan.make_payment(payment_amount)
        else:
            raise ValueError("No loan associated with this
account.")
```

```python
customer = CustomerAccount("1414", "Bishwajit")


print(customer.display_customer_info())


balance_transaction = customer.make_transaction("deposit", 9999)
print(balance_transaction)


withdrawal_transaction = customer.make_transaction("withdrawal",
999)
print(withdrawal_transaction)


loan_number = "LN123"
customer.loan = LoanAccount(loan_number, balance=99999)


interest_calculation = customer.loan.calculate_interest()
print(interest_calculation)


payment_amount = 999
payment_result = customer.make_loan_payment(payment_amount)
print(payment_result)
```

```
balance_after_payment = customer.balance + (customer.loan.balance -
payment_amount)

print(f"Balance after loan payment: Tk.
{balance_after_payment:.2f}")
```

```
bk  .../Python  master ?  v3.12.7  23:47
→ python -u "/home/bk/code/Python/lab_report_7/Bank_account.py"
Customer Name: Bishwajit
Account Number: 1414
Deposited Tk. 9999. New Balance: Tk. 9999
Withdrew Tk. 999. New Balance: Tk. 9000
Interest calculated: +Tk. 4999.950000000001. New Balance: Tk. 104998.95
Paid Tk. 999. Remaining Balance: Tk. 103999.95
Balance after loan payment: Tk. 112000.95

bk  .../Python  master ?  v3.12.7  23:49
→
```

9 Bank Transaction System

```
class bank_account:

    def __init__(self, account_number, customer_name, balance=0,
email=""):

        self.account_number = account_number

        self.customer_name = customer_name

        self.balance = balance

        self.email = email



    def deposit(self, amount):
```

```
        if amount > 0:
            self.balance += amount
            return f"Deposited Tk. {amount}. New Balance: Tk. {self.balance}"
        else:
            return "Invalid deposit amount."



    def withdraw(self, amount):
        if amount > 0 and amount <= self.balance:
            self.balance -= amount
            return f"Withdrew Tk. {amount}. New Balance: Tk. {self.balance}"
        elif amount <= 0:
            return "Invalid withdrawal amount."
        else:
            return "Insufficient funds."



class savings_account(bank_account):
    def __init__(self, account_number, customer_name, balance=0, interest_rate=0.05, email=""):
        super().__init__(account_number, customer_name, balance, email)
        self.interest_rate = interest_rate



    def calculate_interest(self):
```

```
        interest = self.balance * self.interest_rate
        self.balance += interest
        return f"Interest calculated: +Tk. {interest}. New Balance: Tk. {self.balance}"

    def display_details(self):
        print(f"Account Number: {self.account_number}")
        print(f"Customer Name: {self.customer_name}")
        print(f"Balance: Tk. {self.balance}")
        print(f"Interest Rate: {self.interest_rate * 100}%")


class current_account(bank_account):


    def display_details(self):
        print(f"Account Number: {self.account_number}")
        print(f"Customer Name: {self.customer_name}")
        print(f"Balance: Tk. {self.balance}")
        print(f"Email: {self.email}")


class fixed_deposit_account(savings_account):
    def __init__(self, account_number, customer_name, balance=0, interest_rate=0.05, term_years=0, email=""):
        self.term_years = term_years
        super().__init__(account_number, customer_name, balance, interest_rate, email)
```

```python
    def display_details(self):
        print(f"Account Number: {self.account_number}")
        print(f"Customer Name: {self.customer_name}")
        print(f"Balance: Tk. {self.balance}")
        print(f"Interest Rate: {self.interest_rate * 100}%")
        print(f"Term (Years): {self.term_years}")


savings_account = savings_account(
    "Savings123",
    "Bishwajit",
    balance=10000,
    interest_rate=0.05
)
print(savings_account.deposit(500))
print(savings_account.withdraw(200))
print(savings_account.calculate_interest())
savings_account.display_details()
print(f"Email: {savings_account.email}")


current_account = current_account(
    "Current456",
```

```
    "Shah Alom"
)
current_account.display_details()


fixed_deposit_account = fixed_deposit_account(
    "Fixed Deposit789",
    "Bishwajit",
    balance=10000,
    interest_rate=0.10
)
print(fixed_deposit_account.deposit(500))
print(fixed_deposit_account.withdraw(200))
print(fixed_deposit_account.calculate_interest())
fixed_deposit_account.display_details()
```

```
bk  .../Python  master ?  v3.12.7  23:49
→ python -u "/home/bk/code/Python/lab_report_7/another_bank.py"
Deposited Tk. 500. New Balance: Tk. 10500
Withdrew Tk. 200. New Balance: Tk. 10300
Interest calculated: +Tk. 515.0. New Balance: Tk. 10815.0
Account Number: Savings123
Customer Name: Bishwajit
Balance: Tk. 10815.0
Interest Rate: 5.0%
Email:
Account Number: Current456
Customer Name: Shah Alom
Balance: Tk. 0
Email:
Deposited Tk. 500. New Balance: Tk. 10500
Withdrew Tk. 200. New Balance: Tk. 10300
Interest calculated: +Tk. 1030.0. New Balance: Tk. 11330.0
Account Number: Fixed Deposit789
Customer Name: Bishwajit
Balance: Tk. 11330.0
Interest Rate: 10.0%
Term (Years): 0

bk  .../Python  master ?  v3.12.7  23:50
→
```

10 Another Hospital

```python
class Person:
    def __init__(self, name, age, gender):
        self.name = name
        self.age = age
        self.gender = gender


    def display_details(self):
        print(f"Name: {self.name}")
        print(f"Age: {self.age} years")
        print(f"Gender: {self.gender}")
```

```python
class Patient(Person):
    def __init__(self, patient_id, name, age, gender, diagnosis, email=""):
        super().__init__(name, age, gender)
        self.patient_id = patient_id
        self.diagnosis = diagnosis
        self.email = email


    def display_details(self):
        super().display_details()
        print(f"Patient ID: {self.patient_id}")
        print(f"Diagnosis: {self.diagnosis}")
        if self.email:
            print(f"Email: {self.email}")


class InPatient(Patient):
    def __init__(self, patient_id, name, age, gender, diagnosis, room_number, admission_date, email=""):
        super().__init__(patient_id, name, age, gender, diagnosis, email)
        self.room_number = room_number
        self.admission_date = admission_date
```

```python
    def display_details(self):
        super().display_details()
        print(f"Room Number: {self.room_number}")
        print(f"Admission Date: {self.admission_date}")



patient1 = Patient("P001", "Bishwajit", 22, "Male", "cold")
in_patient1 = InPatient("I001", "Shah Alom", 25, "Female",
"Diabetes", "Room 101", "2022-01-01")


print("\nPatient Details:")
patient1.display_details()
print("\nIn Patient Details:")
in_patient1.display_details()
```

```
bk  .../Python  master ?  v3.12.7  23:50
→ python -u "/home/bk/code/Python/lab_report_7/another_hospital.py"

Patient Details:
Name: Bishwajit
Age: 22 years
Gender: Male
Patient ID: P001
Diagnosis: cold

In Patient Details:
Name: Shah Alom
Age: 25 years
Gender: Female
Patient ID: I001
Diagnosis: Diabetes
Room Number: Room 101
Admission Date: 2022-01-01

bk  .../Python  master ?  v3.12.7  23:52
→
```

11 Another Doctor

```python
class doctor:

    def __init__(self, name):

        self.name = name



    def diagnose(self, patient):

        print(f"{self.name} is diagnosing the patient.")



    def prescribe_medicine(self, patient, medicine):

        print(f"{self.name} has prescribed {medicine} to the
patient.")
```

```python
class cardiologist(doctor):
    def __init__(self, name, years_of_experience):
        super().__init__(name)
        self.years_of_experience = years_of_experience


    def perform_ecg(self, patient):
        print(f"{self.name} is performing ECG on {patient.name}.")
        import time
        time.sleep(2)
        print(f"ECG done. Results will be analyzed.")


    def analyze_ecg_results(self, patient, results):
        print(f"{self.name} is analyzing the ECG results for
{patient.name}.")
        import time
        time.sleep(1)
        print(f"ECG results: {results}")



class neurologist(doctor):
    def __init__(self, name, specialization_area):
        super().__init__(name)
        self.specialization_area = specialization_area
```

```python
    def perform_neurological_exam(self, patient):
        print(f"{self.name} is performing the neurological exam for {patient.name}.")
        import time
        time.sleep(1)
        print("Neurological exam done.")


    def diagnose_neurological_condition(self, patient, condition):
        print(f"{self.name} is diagnosing the {condition} in {patient.name}.")
        import time
        time.sleep(2)
        print(f"Diagnosis confirmed: {condition}")


class Patient:
    def __init__(self, name, email):
        self.name = name
        self.email = email


    def get_patient_info(self):
        return f"Name: {self.name}, Email: {self.email}"
```

```python
doctor = doctor("bishwajit@gmail.com")
cardiologist = cardiologist("shah alom@yahoo.com", 10)


patient1 = Patient("John Doe", "john.doe@example.com")
print(patient1.get_patient_info())


doctor.diagnose(patient1)
cardiologist.prescribe_medicine(patient1, "aspirin")


cardiologist.perform_ecg(patient1)
cardiologist.analyze_ecg_results(patient1, "normal ecg results")


neurologist = neurologist("shah alom@hotmail.com", "neurology")


patient2 = Patient("Shafi", "jane.doe@gmail.com")
print(patient2.get_patient_info())


neurologist.diagnose_neurological_condition(patient2, "migraine")
```

```
bk  .../Python  master ?  v3.12.7  23:52
→ python -u "/home/bk/code/Python/lab_report_7/another_doctor.py"
Name: John Doe, Email: john.doe@example.com
bishwajit@gmail.com is diagnosing the patient.
shah alom@yahoo.com has prescribed aspirin to the patient.
shah alom@yahoo.com is performing ECG on John Doe.
ECG done. Results will be analyzed.
shah alom@yahoo.com is analyzing the ECG results for John Doe.
ECG results: normal ecg results
Name: Shafi, Email: jane.doe@gmail.com
shah alom@hotmail.com is diagnosing the migraine in Shafi.
Diagnosis confirmed: migraine

bk  .../Python  master ?  v3.12.7  23:53
→
```