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")

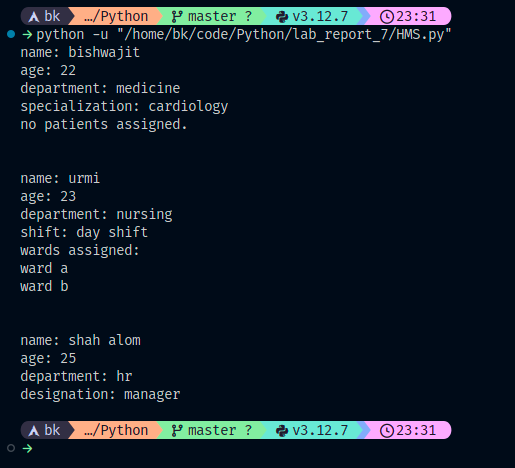
nurse = nurse("urmi", 23, "nursing", "day shift", ["ward a", "ward b"])

nurse.get\_details()

print("\n")

administrative\_staff = administrative\_staff("shah alom", 25, "hr", "manager")

administrative\_staff.get\_details()



2 Hospital System

*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*

*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")

inpatient1.get\_patient\_info()

print("\n")

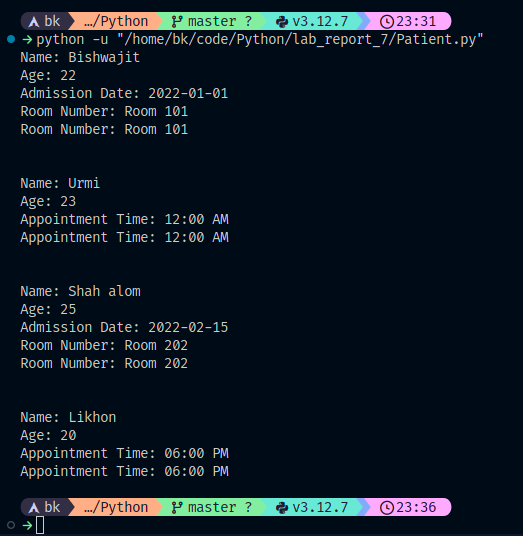
outpatient1.get\_patient\_info()

print("\n")

inpatient2.get\_patient\_info()

print("\n")

outpatient2.get\_patient\_info()



3 Bank Management System

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")

savings\_account = SavingsAccount(9999, 1111)

print(savings\_account.balance)

checking\_account = CheckingAccount(1111, 111)

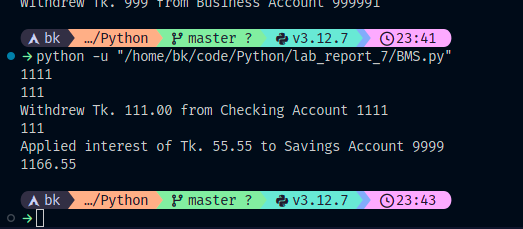
print(checking\_account.balance)

checking\_account.withdraw(111)

print(checking\_account.balance)

savings\_account.apply\_interest()

print(savings\_account.balance)



4 Student Bank Management System

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)

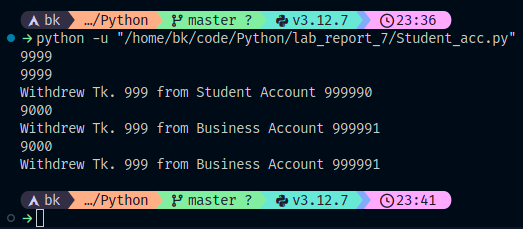
student\_account.withdraw(999)

print(student\_account.balance)

business\_account.withdraw(999)

print(business\_account.balance)

business\_account.withdraw(999)



5 Student Management System

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):

*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")

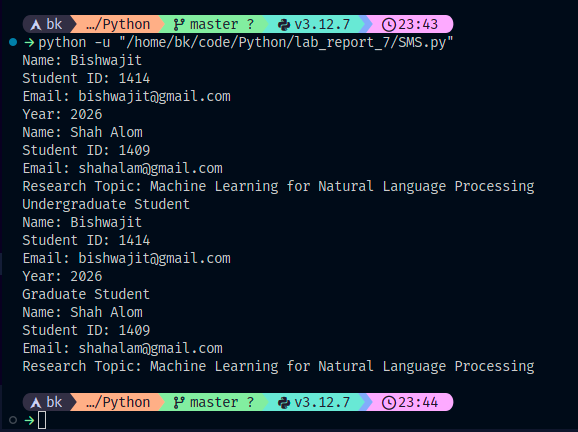
print(undergraduate.get\_info())

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)



6 University Management System

*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

*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())

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)



7 Library Management System

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

*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

*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)

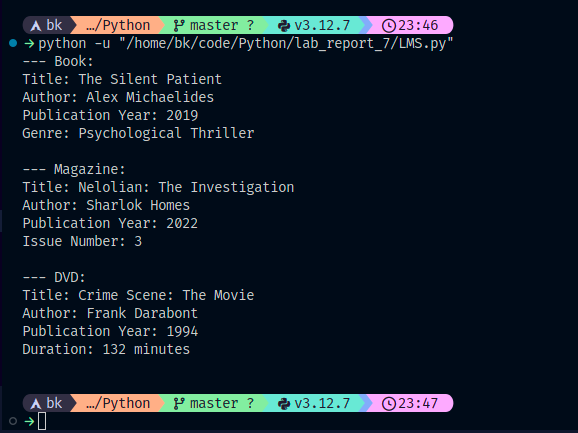
dvd = DVD("Crime Scene: The Movie", "Frank Darabont", 1994, 132)

library.add\_item(book)

library.add\_item(magazine)

library.add\_item(dvd)

library.display\_all\_items()



8 Banking Account System

*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:

*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.")

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)

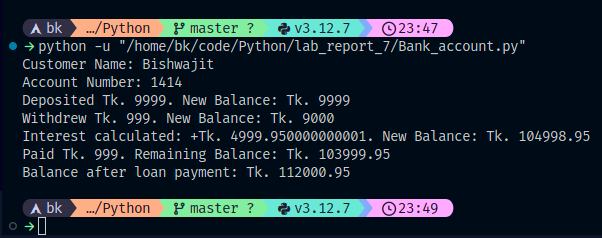
payment\_amount = 999

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*}")



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*)

*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

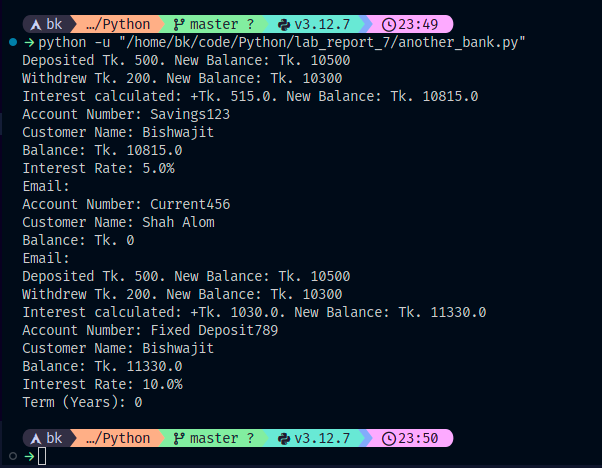
)

print(fixed\_deposit\_account.deposit(500))

print(fixed\_deposit\_account.withdraw(200))

print(fixed\_deposit\_account.calculate\_interest())

fixed\_deposit\_account.display\_details()



10 Another Hospital

*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}")

*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*

*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")

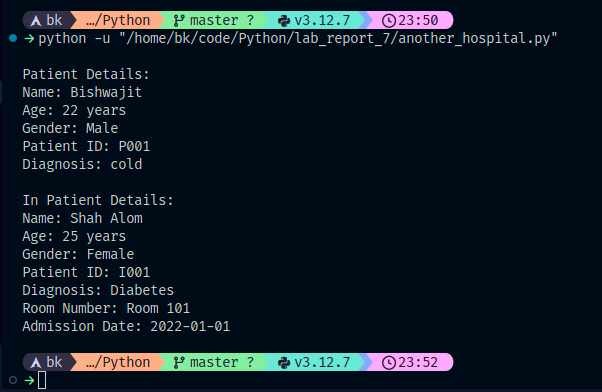
in\_patient1 = InPatient("I001", "Shah Alom", 25, "Female", "Diabetes", "Room 101", "2022-01-01")

print("\nPatient Details:")

patient1.display\_details()

print("\nIn Patient Details:")

in\_patient1.display\_details()



11 Another Doctor

*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.")

*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*

*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}"

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")

