Programs to demonstrate inheritance

A.

Imagine a publishing company that markets both book and audio cassette versions of its works. Create a class **Publication** that stores the title and prize of a publication. From this class derive two classes: **Book**, which adds the page count and **Tape** which adds the playing time in minutes. Each of these three classes should have a get() function to get its data from the user at the keyboard, and a set() function to display its data. Write the main() program to test the Book and Tape classes by creating instances of them, asking the user to fill in data with get(), and then displaying the data with set().

B.

Create a class called **Account** that has the protected data members accnumber , balance and constructors for initialization. Derive a class called **SBAccount** and define methods deposit(amount), withdraw(amount), calc_interest(). Provide a parameterized constructor with two parameters account number, and init_balance and assign these values to the appropriate data members of the class using super class constructor. The deposit method should take one double type argument amount and adds the amount to the balance if the amount is positive. The withdraw method, should take one double type argument amount and checks if balance - amount is greater than Rs.1000/-. If so, it should subtract the amount from balance. Otherwise it should display error message. The calc_interest() method should compute interest for the balance amount available @ 4% for one year and the interest amount should be credited to the balance.

Derive another class called **FDAccount** that has a data member period. Provide parameterized constructor with parameters for accno, period and deposit amount for initializing them. Provide a method called calc_interest() for calculating interest for the deposit amount for the given period @ 8.25% p.a. and returns the calculated interest, and another method called close() which calls calc_interest() add it to the balance.

Create a class called **Customer**. The data members of the class are cust_id type, name, and address, and objects of SBAccount, and FDAccount. Provide parameterized constructor with cust_id, name and address as parameters for initializing. Provide a method called createAccount(type). Based on the value of type (SB, or FD) create a new account of given type. Provide a method transaction (type) {type may be withdrawn, deposit or calc_interest on SBAccount, or closing of FDAccount} to perform the requested transaction on the requested Account object.

To demonstrate all the functionalities of above classes, create a class called **BankDemo** and declare a main method to create list of Customer objects, and perform manipulations on the objects by creating SB and FD accounts, depositing and withdrawing amount from SB account, and closing FD account.

C.

Define a class called **Politician**. Define getLIMIT() method to return 5 crores as the limit. Define doITRaid() method that calls evaluateProperty() if the person has property, otherwise returns zero.

Define another class called **Professional**. Define getLIMIT() method to return 10 times of his annual income. Define doITRaid() method that calls getDeposit() if the person has deposit, otherwise returns zero.

Define class called **Person** that holds name and status (private member). Define constructor for initializing its members, getStatus() to return status, and ITRaid() that calls doITRaid().

Define a class called **Minister** derived from Person and Politician. It has two members called party_name to hold name of the party he is belonging to and asset_value, a dictionary having name & value of the asset for each of his properties. Provide constructor to initialize its super class attributes by calling the super class constructor and along with initialization of its own members. Define a method called hasProperty() to check if asset_value is empty or not. Define a method called evaluateProperty() to return the total asset value.

Define a class called **Officer** derived from Person and Professional. It has three members called qualification, monthly_income, and deposit_value a dictionary having account_number & deposit amount for each of his deposit and. Provide constructor to initialize its super class attributes by calling the super class constructor along with initialization of its own members. Define a method called hasDeposit() to check if deposit_value is empty or not. Define a method called getDeposit() to return the total deposit amount.

Define an exception class called **ITRaid**. Provide a member called errmsg to hold error message. Provide __str__() method to return the string "ITRaid Exception: " along with the error message.

Create a list of Persons of type both Minister and Officer and call ITRaid() for each person. If the total value of the Raid exceeds the corresponding limit according to their status (Minister/Officer) raise the ITRaid exception. If the person is a Minister and the limit exceeds, the raid should display "You have Disproportionate Assets". If the person is an Officer and the limit exceeds the raid should display "Your Deposits are Locked". If the limit is not exceeded, display the message "Good:".