OBJECT ORIENTED PROGRAMMING

[THURSDAY OCTOBER 5, 2023: 08:45 AM – 12:00 NOON]

ASSIGNMENTS – 06 CODE: ASSIGN06

NOTES:

i) Create a main class with the following file naming conventions: If your roll number ends with **abc**, year of admission is 2022 and assignment code is **Assign06** then, use the wrapper class (having main method) file name as follows: Assign062022abc.java (use the extension .java in lowercase)

   For example, if the roll number ends with 127; year of admission is 2022 & the assignment code is Assign06, then the file name should be **Assign062022127.java**

ii) Strictly follow the file naming convention. Otherwise, it would attract a penalty up to 20%.

PROBLEM: package name: `iiits.oop.m2023.inherit`

Use different types of inheritances to solve the given problems and use random numbers as necessary.

a) Define a superclass: **`SpareBank`** with suitable branch related information.

   States: Use suitable data structures for these member variables

   Define **`Customer`** class that consists of the basic profile details of a customer having at least any two of the following identity related information, apart from the name, dob (MM/DD/YYYY), designation, address, CIF number (Customer Information Folio) and balance.

   i. **`Passport Number`** format – [a-zA-Z]{1}[0-9]{7} – For example, H3845776 is a valid passport number
   ii. **`Personal Number`** – comprising of 12 digits with two spaces – for separating a group of 4 digits, for example: 0234 1245 8779 is a valid personal number
   iii. **`Driving Licence`** format: [0-9]{6}/[1-9]{4}. For example, 135647/1998 is a valid driving licence number. Here the last four digits after '/' represent a year in [1920, 2018]
   iv. **`Employee ID card`** number format: [a-zA-Z]{3}[0-9]{2}[0-9]{6}. For example, SEC94011074 is a valid ID card number where last 6 digits refers to date of birth

   Behaviour: Define different behaviours of the superclass

   i. **`getName()`**: This returns the name of the customer with his / her title (Mr/Ms/Dr)
   ii. **`getAge()`**: Take dob as the input and return the age in years and months
   iii. **`isValid()`**: This method should take any of the above 4 identify related information and return true if it is in the valid format; false, otherwise.
   iv. **`showAccounts()`**: This method should take two arguments: CIF number and any one of the identity related information and show the details of all accounts including the balance in each account. There may be more than one account under the CIF number of a customer.

Wrapper class with the file name as given in the instruction will only have the main method. So every solution to be written for the following problems should be invoked from this main method only. Apply Inheritance wherever possible.

b) Let there be n customers in the SpareBank where n $\in$ [10, 20]. Each customer should have at least one of the following account types: (i) Savings (ii) Current (iii) Deposit (in the worst case, one customer may have multiple accounts that span over all three account types). Apply Hierarchical Inheritance to handle the above task with a minimum of Rs. 5000/- for savings and current account and 10,000/- for the fixed deposit account with minimum of 1-year duration. Define suitable methods to get the balance of each account.

c) Apply Multiple Inheritance on any of the above account types in such a way that the Balance (as on a specific date) will be computed based on credits and debits made at a particular duration. Derive the following:

$$\begin{tikzcd}
& & \text{Credits} \arrow[dl] & \\
\text{Customer} & \text{Account Type} \arrow[l] & & \text{Balance} \arrow[ul] \arrow[dl] \\
& & \text{Debits} \arrow[ul] &
\end{tikzcd}$$

d) [Marks: 5] Apply the following interest rates using single level inheritance:

Account Type ← Interest

and define suitable methods to set and get the accumulated interest as on a particular date.
Saving Account: 2.75% for every 3 months; Current Account: 2.25% for every 3 months and
Fixed Deposit: 7.1% for every 12 months

e) Apply multilevel inheritance to compute Taxation - The bank calculates the tax deducted at source (TDS) on the interest accumulating every month in each account. Derive the following: Customer ← Account Type ← Taxation and define suitable methods in the derived class to compute the TDS. You may use a suitable Virtual Method to print the TDS of each account under different account types.

Amount of Interest (Each Month)	**TDS**
amount <= 1,000	No TDS Deduction
amount > 1,000 & amount <= 5,000	10 %
amount > 5,000 & amount <= 10,000	20 %
amount > 10,000 & amount <= 20,000	25 %
amount > 20,000 & amount <= 50,000	30 %
amount > 50,000 or above	35%

f) Write an interface PrintDetails having two methods PrintProfile and PrintSummary. Implement these two methods from the member inner class / or in an appropriate class to print the customer profile details (using PrintProfile) and the summary of all accounts (using PrintSummary).