Assignment 04 – Inheritance and Polymorphism World

Average expected estimated time to complete this assignment is 12 hours

- [1 HOUR] software requirement understanding
  - reading of the assignment document
  - understanding of problem statement
  - understanding of software inputs and outputs
- [2 HOURS] software design
  - identification of user-software interaction sequences
    - software menus
    - user inputs and
    - software flows
  - identification of required variables and data structures to be used
    - for input, processing, and output
    - identification of storage class specifications of required variables
  - modularization of software into required functions
    - proper naming of functions
    - identification of tasks to be performed into functions
    - identification of parameters and return types of functions
  - identification of user defined header files
    - names and functions to be placed in
  - flow sequence of main function and call of various user defined functions from it
- [6 HOURS] software coding
  - coding of user defined functions
  - placement of user defined functions into header file/s
  - coding of main function
- [2 HOURS] software testing
  - running software on various inputs
  - identification of software error and bugs
  - removal of software error and bugs
- [1 HOUR] software documentation
  - proper indentation of code
  - use of proper naming conventions
  - commenting of code

Objectives:

- This assignment will provide experience with concepts of Inheritance and Polymorphism

Prerequisite Skills:

- Basic knowledge of mathematical calculations
- Understanding of Class design
- Understanding of Inheritance
- Understanding of Polymorphism

Instructions before you start:

- Understanding of the assignment is part of the assignment.
- For each class, you will have a header file and a cpp file with class declarations in header file and definitions in .cpp file.
- Handle corner cases so that your program never crashes or produces wrong output.
- First understand the whole Assignment structure and then start to implement it.
- Do not use the prohibited means for implementing this assignment.
- Good luck!!!.

Task 01: Vehicles- Show

Overview:

Polymorphism is the third pillar of object-oriented programming (OOP) after encapsulation and inheritance. Polymorphism allows an interface to be expressed in different ways. Real-world examples are commonplace; for example, the taxonomy used in biology to classify organisms is polymorphic.

In this assignment, you will create a program that models a vehicle inventory using Polymorphism. The inventory will contain several types of vehicles such as cars, trucks, and motorcycles, each of which will have different properties and methods.

Problem Statement:

Instructions:

- Create a base class called "***Vehicle"*** that has the following data members.
    - make (string),
    - model (string),
    - year (integer),
    - price (double).

The class should also have a virtual method called ***"display Info"*** that takes no parameters and returns nothing.

- Create three derived classes: **"Car", "Truck", and "Motorcycle"** that inherit from the "Vehicle" base class.

    - The **"Car"** class should have an additional data member called **"numDoors"** (integer).
    - The **"Truck"** class should have an additional data member called **"bedSize"** (string).
    - The **"Motorcycle"** class should have an additional attribute called **"engineSize"** (double).

  All derived classes should override the ***"display Info"*** to display its specific vehicle information.

  The "**Car"** class should output the make, model, year, price, and number of doors.
  The **"Truck"** class should output the make, model, year, price, and bed size.
  The **"Motorcycle"** class should output the make, model, year, price, and engine size.

- Create a main program that creates several instances of each vehicle type, sets their properties, and calls their ***"display Info"*** method to display them. Use a polymorphic approach to call the "display Info" method of each vehicle object. The output of the program should be a visual display of all the vehicles in the inventory.

Specifications:

Implement the following features to enhance the program:

- Implement a way to add new vehicles to the inventory.
- Implement a way to remove vehicles from the inventory.
- Implement a way to search for a specific vehicle by make, model, year, or price.
- Implement a way to sort the vehicles by make, model, year, or price.

Task -02: Bank System

In this task you will demonstrate working of a Bank System. You see, customers at bank can have accounts of various types such as:

- **Business account** is for the individuals that are involved in major business activities and have to make large and many transactions on daily basis. Special thing about business account is that you can make withdrawals and transfers even you don't have balance in your account. In that case that particular amount is added as negative balance to your account and account holders pay that amount at the end of the month.

- **Student Account** is for students who are not earning yet on their own but are supported by some guardian and they need to make daily life money transactions.

- **Savings account** is for the individuals who want to safeguard and store their side money at the bank for the purpose of safe keeping and to earn interest on it after a specified time.

- **Credit Account** is for the general purpose banking life of the citizen. It is the most commonly form of account type used by the customers. Users can withdraw and deposit money in specified ranges in daily routine.

- **Loan Account** is for the customers who wants to obtain loan from Bank for a period of time. The customers must return the amount in addition to some decided interest amount in multiple payments within this time period.

- **Special Citizen Account** is for elderly or disable citizens which receive payments/ retirement pension or other support from Government or any other organization.

Bank Account

No matter what is type of the account, a bank account in simple terms has the information stored such as Account holder name, his/her other personal information and Balance.

Now lets start to form up our Banking system using the concepts of inheritance and Polymorphism we have just learned.

First create a base class **Person**. We will use this class as parent class to represent any Personals in our system.

1. Person

- **id** (uniquely self generated)
- **First name**
- **Last name**
- **CNIC** (must be exactly of 13 digits, input must be taken in format (34***-*******-*) (must be unique as well. Although the CNIC will also be unique for each person, Bank wants to maintain a unique id of its own for identification, for performance gains.)

- **Address** (Object of address class. We will need an address class to store address information for persons.)
- **Date of Birth** (Object of Date class. We will need an Date class to store Date of Birth information for persons.).
- **Father** (Pointer to an other Person object).
- **Mother** (Pointer to another Person object).

Methods

- Constructors (all three), Destructor and setter getters.
- Display function

Now lets move on to Account class. This account class of ours will be the Base class for all the Account types class we will make onwards. Lets get straight into it.

2. Transaction class

Transaction class will be used to store information about a transaction made between two users. We will store the following information in it:

- transaction id
- sender id
- sender name
- receiver id
- receiver name
- Amount transferred
- Date of transaction (Date type)

Methods:

Create the required methods that you will need for this class.

3. Account

- **Account_number** (unique string)
- **Account holder** (Person)
- **Balance** (We want to be able to be able to store a large number but we also can't afford to waste memory space as well. So keeping this in mind, we will store the Balance amount in a char array. In char array, store each digit of Balance amount at each index of this array. Let say our balance is 1250 rupees. Store the digits like this:
- 1 2 5 0).
- **Transactions history**: It will be a vector of type ***transaction***.

Methods

- Constructors (all three), Destructor and setter getters.
- Check Balance
- View transaction history

Create following Pure virtual functions in this class:

- Display function
- Withdraw(amount)
- Deposit (amount)
- Transfer (amount)

Now lets start making Different types of Account classes for our Bank which will be derived from our **Account class**.

4. Business Account

We have already discussed what a Business Account is. Now lets discuss what specifications it will have. Following are some specifications for Business account:

Data members:

- **Negative_balance**. (It will be monthly based and must be payed back at the end of each month. If at the end it is not cleared then add 10 percent additional amount for the next month as charge.

Functionalities:

- Business accounts are not charged for any kind of Transaction fee for Withdrawal or Transfer.
- Business Accounts are not limited for amount of withdrawal or deposit or Transfer in a month, however if the total transactions amount (all withdrawal + deposit + transfer) exceeds 20 million, then charge 3 percent fee.
- Provide functionalities to withdraw, transfer or deposit amount from one Account to another. Store the information of each transaction in transaction vector.
- You will write definitions for all virtual functions.
- Provide functionality to clear, increment, addition for next month, show for negative balance.

5. Student Account

We have already discussed what a Student Account is. Now lets discuss what specifications it will have. Following are some specifications for Student account:

Data members:

- **Deposit_limit:** total balance limit.
- **Withdrawal_limit:** withdrawal limit for a month
- **Transfer_limit:** transfer limit for a month.
- **Withdrawal_charge:**
- **Transfer_charge:**
- **Number of transactions made:** This many transactions can only be made maximum in a single day.

Functionalities:

- Student account will be charged only once in a day for all the withdrawals (25 rupees) and for transfer (5 rupees) respectively.
- Student Account has limit of 50,000 for Balance.
- In a single day, student can only withdraw 90 percent of the balance. For transfer the limit is 20,000 for a single day.
- Provide functionalities to withdraw, transfer or deposit amount from one Account to another. Store the information of each transaction in transaction vector.

- You will write definitions for all virtual functions.

6. Savings Account

Keeping in mind the purpose of savings account, following will be the specifications.

Data members:

- **Time Period:** Time period after which user will be given returned his/her amount in addition with the interest amount.
- **Interest rate:** Decided interest rate.
- **Package start date:** Date object.

Functionalities:

- Provide functionalities to withdraw, transfer or deposit amount from one Account to another. Store the information of each transaction in transaction vector.
- Withdrawal is not allowed is the Time period is not complete yet.
- Transfer functionality is not allowed for Savings account.
- You will write definitions for all virtual functions.
- Provide the function to provide the amount that will be returned for current interest rate and balance.

7. Credit Account

Credit account has following functionalities:

Data members:

- **Withdrawal_charge:**
- **Transfer_charge:**
- **Deposit_limit:** total balance limit.
- **Withdrawal_limit:** withdrawal limit for a month
- **Transfer_limit:** transfer limit for a month.

Functionalities:

- Provide functionalities to withdraw, transfer or deposit amount from one Account to another. Store the information of each transaction in transaction vector.
- For each withdrawal, 25 rupees will be deducted. In withdrawal ask if the bank is same. If yes, then don't deduct withdrawal amount.
- Transfer charge is 10 rupees. Ask If the bank is same or not. If same then do not deduct this amount.
- Implement all virtual functions from base class.

8. Loan Account

Keeping in mind the purpose of Loan account, following will be the specifications.

Data members:

- **Interest rate:** Decided interest rate with which the user will return the amount.
- **Total loan:** total loan obtained.
- **Time Period:** Time period after which user will return his/her amount in addition with the interest amount.
- **Charge_rate:** If the user is not able to return the loan in the time period, he/she will be charge this much rate of the loan in addition to the loan amount.

Functionalities:

- Provide functionalities to get Loan (withdrawal), and check and return Loan (deposit) in payments (loan). In transactions history, sender will be user and receiver will be the bank.
- Transfer functionality is not allowed for this account.
- You will write definitions for all virtual functions.
- User cannot get more loan, once the already obtained loan is not completely returned.

9. Special Citizen Account

Keeping in mind the purpose of Special Citizen account, following will be the specifications.

Data members:

Supporter: info of the supporter, Govt/ organization.
Fixed amount per month: fixed amount users will receive at the start of each month.

Functionalities:

- Provide functionalities to withdraw, transfer or deposit amount from one Account to another. Store the information of each transaction in transaction vector.
- Do not allow to deposit amount more than specified limit.
- Transfer functionality is not allowed for this account.
- You will write definitions for all virtual functions.
- No deduction at time of withdrawal.

Finally all the classes are done which were required for our Bank class. Well done!. Have a little clap on the back of yourself. Have a break for few mins, have some air and water and Daylight. You deserve it now.

Now that you are back, Let work on our Bank class.

13. Bank Class

Bank class will be the major class in which we will use the functionality of all the classes created earlier. Following are the Specifications:

Data members:

- **Bank name**
- **Number of accounts**
- **Total balance:** Total amount the bank has at the moment.
- **Total interest to pay:** Total interest the bank has to pay to his customers
- **Total profit:** Total interest the bank has to receive from his customers.
- **Vector of type Account * ……**

Functionalities:

- Provide functionality to add account of any type. Accounts will be added to the respective vector of their type.
- Provide functionality to display all accounts of any type. Also provide functionality to see the detail of a specific account given the account number.
- Provide functionality to see all the account of a person by providing the person's CNIC.
- Allow to make withdraw, deposit and transfer transactions by any account of any type.
- Allow to see the account history of any account given account number.
- Allow functionality to rollback any transaction. When a transaction is to be roll backed, we do its reverse operation, means if the balance was increased, we decreased the balance with same amount and so on…..
- Provide functions to provide Loans get get Loans back.
- Maintain the bank's attribute's data consistent.

Main function

In main function, you will provide functionality to select and run any of the functions of Bank class. Main function should be user friendly and Menu-based.

Exception Handling:

Use Exceptional Handling throughout your code to handle corner cases. Your program should not crash.

That's It. Well Done!!