**Assignment 02**

**Object Oriented Programming, spring 2023**

**Department of Software Engineering,**

**Faculty of Computer Science & Information Technology,**

**The Superior University Gold Campus, Lahore.**

**Submission Deadline: Monday 11:59 PM, May 15, 2022**

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

|  |  |  |  |
| --- | --- | --- | --- |
| Question | CLO | Domain/BT Level | Total Marks |
| 1-4 | 3 | C2 | 85 |

**Advice and Submission Guidelines for Assignment**

* Make sure that you read and understand each and every instruction. If you have any questions or comments you are encouraged to discuss (only) with your colleagues and instructor.
* In case of coding assignment, paste all codes and screenshots of output on word file, later you can convert this word file into PDF file as well. **Upload PDF file on the LMS** and keep code files with you (better in your own email) as it will be used for evaluation and viva.
* All the submitted evaluation instruments (quizzes, assignments, lab work, exams, and the project) will be checked for plagiarism.
* Later Submission and Plagiarism will be dealt as per university Policy
* Start early otherwise you will struggle with the assignment.

*Note: Keep all your code files. It will be required at any time of evaluation.*

**Question 1 (Inheritance) Marks : 20**

(Account Inheritance Hierarchy) Create an inheritance hierarchy that a bank might use to represent customers’ bank accounts. All customers at this bank can deposit (i.e., credit) money into their accounts and withdraw (i.e., debit) money from their accounts. Morespecific types of accounts also exist. Savings accounts, for instance, earn interest on the money they hold. Checking accounts, on the other hand, charge a fee per transaction (i.e., credit or debit). Create an inheritance hierarchy containing base class Account and derived classes SavingsAccount and CheckingAccount that inherit from class Account.

Base class Account should include one data member of type double to represent the account balance. The class should provide a constructor that receives an initial balance and uses it to initialize the data member. The constructor should validate the initial balance to ensure that it’s greater than or equal to 0.0. If not, the balance should be set to 0.0 and the constructor should display an error message, indicating that the initial balance was invalid. The class should provide three member functions. Member function credit should add an amount to the current balance. Member function debit should withdraw money from the Account and ensure that the debit amount does not exceed the Account’s balance. If it does, the balance should be left unchanged and the function should print the message "Debit amount exceeded account balance." Member function getBalance should return the current balance. Derived class SavingsAccount should inherit the functionality of an Account, but also include a data member of type double indicating the interest rate (percentage) assigned to the Account. SavingsAccount’s constructor should receive the initial balance, as well as an initial value for the SavingsAccount’s interest rate. SavingsAccount should provide a public member function calculateInterest that returns a double indicating the amount of interest earned by an account. Member function calculateInterest should determine this amount by multiplying the interest rate by the account balance. [Note: SavingsAccount should inherit member functions credit and debit as is without redefining them.] Derived class CheckingAccount should inherit from base class Account and include an additional data member of type double that represents the fee charged per transaction. CheckingAccount’s constructor should receive the initial balance, as well as a parameter indicating a fee amount. Class CheckingAccount should redefine member functions credit and debit so that they subtract the fee from the account balance whenever either transaction is performed successfully. CheckingAccount’s versions of these functions should invoke the base-class Account version to perform the updates to an account balance. CheckingAccount’s debit function should charge a fee only if money is actually withdrawn (i.e., the debit amount does not exceed the account balance). [Hint: Define Account’s debit function so that it returns a bool indicating whether money was withdrawn. Then use the return value to determine whether a fee should be charged.] After defining the classes in this hierarchy, write a program that creates objects of each class and tests their member functions. Add interest to the SavingsAccount object by first invoking its calculateInterest function, then passing the returned interest amount to the object’s credit function.

**Note: Please decide here which type of inheritance can exist here**

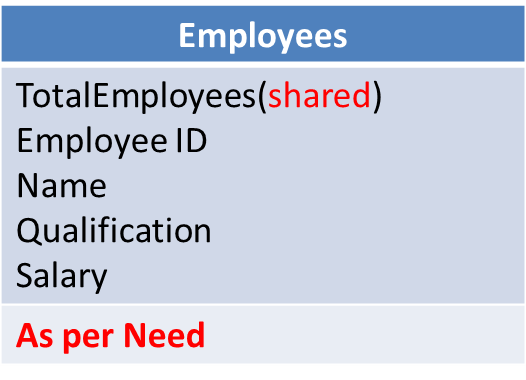
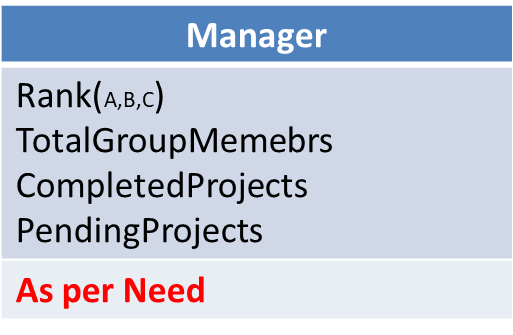
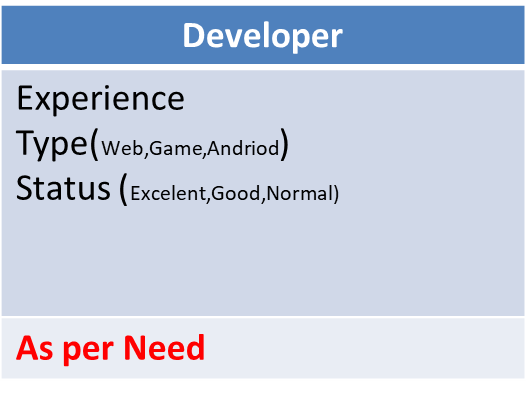
**Question 2(Inheritance) Marks : 20**

Package-delivery services, such as FedEx ®, DHL® and UPS®, offer a number of different shipping options, each with specific costs associated. Create an inheritance hierarchy to represent various types of packages. Use class Package as the base class of the hierarchy, then include classes TwoDayPackage and OvernightPackage that derive from Package. Base class Package should include data members representing the name, address, city, state and ZIP code for both the sender and the recipient of the package, in addition to data members that store the weight (in ounces) and cost per ounce to ship the package. Package’s constructor should initialize these data members. Ensure that the weight and cost per ounce contain positive values. Package should provide a public member function calculateCost that returns a double indicating the cost associated with shipping the package. Package’s calculateCost function should determine the cost by multiplying the weight by the cost per ounce. Derived class TwoDayPackage should inherit the functionality of base class Package, but also include a data member that represents a flat fee that the shipping company charges for two-day-delivery service. TwoDayPackage’s constructor should receive a value to initialize this data member. TwoDayPackage should redefine member function calculateCost so that it computes the shipping cost by adding the flat fee to the weight-based cost calculated by base class Package’s calculateCost function. Class OvernightPackage should inherit directly from class Package and contain an additional data member representing an additional fee per ounce charged for overnight-delivery service. OvernightPackage should redefine member function calculateCost so that it adds the additional fee per ounce to the standard cost per ounce before calculating the shipping cost. Write a test program that creates objects of each type of Package and tests member function calculateCost.

**Task 3 (Inheritance) Marks: 15**

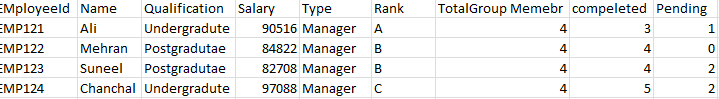
Create a SavingsAccount class. Use a static data member annualInterestRate to store the annual interest rate for each of the savers. Each member of the class contains a private data member savingsBalance indicating the amount the saver currently has on deposit. Provide member function calculateMonthlyInterest that calculates the monthly interest by multiplying the balance by annualInterestRate divided by 12; this interest should be added to savingsBalance. Provide a static member function modifyInterestRate that sets the static annualInterestRate to a new value. Write a driver program to test class SavingsAccount. Instantiate two different objects of class SavingsAccount, saver1 and saver2, with balances of $2000.00 and $3000.00, respectively. Set the annualInterestRate to 3 percent. Then calculate the monthly interest and print the new balances for each of the savers. Then set the annualInterestRate to 4 percent, calculate the next month’s interest and print the new balances for each of the savers

**Question 4 (Inheritance, Array of Objects and Static) Marks : 30**

****Below figures shows Inheritance relationship

**Following task are needed to be done**

1. Create each class, attributes (choose data type and function yourself). Inherit these classes as shown above, select inheritance type as required by task below.
2. You are given a file which contains data of Manager and developer of a company.
3. First number shows total employees, second show total managers and third number shows total Developer.
4. Read file in array of objects of two different classes. This should be dynamic array and program should be generalized. Manager data is always placed at first place and then developer data will be placed.
5. Attribute in file are in same flow as describe above for example





****

1. Create following Menu(These functions should be in main() file)
   1. Add Record of a new employee(Manager or Developer)
   2. Delete Record of a employee(Manager or Developer)
   3. Update Record of existing employee
   4. Display total number of employee
   5. Show developers data on the basis of experience i.e. developer with high experience should be displayed first.(**Caution: you have to sort developer data on basis of experience, please verify rest of employee’s data are correct displayed)**
   6. Print the name of Manager who has completed highest projects

**Note: If we add or remove employee, TotalEmployees Will be changed. All changes should also be made in file.**