National University of Computer and Emerging Sciences



Lab Exercise 9

For

Object Oriented Programming Lab

|  |  |
| --- | --- |
| Course Instructor(s) | Dr. Danish |
| Lab Instructor(s) | Mr. Mughees Ismail |
| Semester | Spring 2020 |

**FAST School of Computing**

# Instructions:

1. Make a word document with the naming convention “SECTION\_ LAB#\_ROLLNO” and put all your source code and snapshots of its output in it. Make sure your word file is formatted properly.
2. Plagiarism is strictly prohibited.
3. Do not discuss solutions with one another.

# Useful links

|  |
| --- |
| **Question#1** |

Design a class named PersonData with the following member variables:

* FirstName
* LastName
* Address
* City
* State
* Zip
* PhoneNo

Write the appropriate accessor (getter) and mutator (setter) functions for these member variables.

Next, design a class named CustomerData, which is derived from the PersonData class. The CustomerData class should have the following member variables:

* CustomerNumber
* MailingList

The CustomerNumber variable will be used to hold a unique integer for each customer. The MailingList variable should be a bool. It will be set to true if the customer wishes to be on a mailing list, or false if the customer does not wish to be on a mailing list. Write appropriate Accessor and Mutator functions for these member variables. CustomerData class will have the

* InputCustomerData member function which will Input all the data for customer. (use function over riding).
* DisplayCustomerData member function which will display all the data for customer. (use function over riding).

Demonstrate an object of the CustomerData class in a simple program.

|  |
| --- |
| **Question#2** |

A retail store has a preferred customer plan where customers may earn discounts on all their purchases.

The amount of a customer’s discount is determined by the amount of the customer’s cumulative purchases in the store.

* When a preferred customer spends $500, he or she gets a 5% discount on all future purchases.
* When a preferred customer spends $1,000, he or she gets a 6% discount on all future purchases.
* When a preferred customer spends $1,500, he or she gets a 7% discount on all future purchases.
* When a preferred customer spends $2,000 or more, he or she gets a 10% discount on all future purchases.

Design a class named PreferredCustomer, which is derived from the CustomerData class you created in **problem 1**. ThePreferredCustomerclass should have the following member variables:

* purchasesAmount (a double)
* discountLevel (a double)

The purchasesAmount variable holds the total of a customer’s purchases to date. The discountLevel variable should be set to the correct discount percentage, according to the store’s preferred customer plan. Write appropriate member functions for this class and demonstrate it in a simple program.

|  |
| --- |
| **Question#3** |

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. More specific 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. Easy?