Week 4: Constructor, Destructor, Default Parameters, const and static Members, Pass object and return object

Learning Materials: Chapter 6

Home Practice Task

For all lab tasks in Lab 3 (copy the code of Lab 3 to Lab 4), modify the class definition to add **constructors** that assign the member variables to an appropriate initial value. Write a zero-argumented constructor as well as an argumented constructor that initializes the member variable to the passed arguments' value.

Add a destructor function that displays a message: **"The destructor for the object(<name of the class>) is called.**" This demonstrates that for each object, when the lifetime ends, the destructor is called.

*** Implement all the appropriate constructors, destructors, and get/set methods. Only make the necessary member functions public; otherwise, keep them private.

Task 1

Define a class “**Product**” with following description:

- addToInventory(int addedQuantity) - Adds a specified quantity to the current product inventory.
- isAvailable() - Checks if the product is available (i.e., if the quantity is greater than 0).
- purchase(int purchasedQuantity) - If the product is available, reduces the quantity by the purchased amount.
- updatePrice(int discountPercent) - Adds a discount to the current price.
- displayInventoryValue() - Displays the total value of a particular product in the inventory (quantity * price).
- displayDetails() - Prints the details of the product (name, id, price, quantity, availability status).
- displayTotalInventoryValue() - Displays the total value of all the products available in the inventory.

The main() function should reflect:

First, initialize a product using the product name, its unique ID, unit price, the initial quantity, and the maximum amount of that product. Then add an extra quantity of that product to the inventory. Make a purchase of that product. After that, provide a 5% discount and update the price accordingly. Also, calculate the inventory value of that product. Finally, display all the details.
Create 2 more products and display the total inventory value of all 3 products.

Task 2

Define a class “**BankAccount**” with the following description.
Each account will have the following information:

Private members

- The account number.
- The account holder name.
- The account type (current/savings) (assume the data type)
- The current balance.
- The minimum balance (An account has to maintain the minimum amount; it cannot withdraw. It can only be initialized at the time of creating object. **USE const modifier**)

The class will have the following criteria:.

Public members

- The member variable value of the object can be assigned **during** object **creation** or **after** the object has been created.
- A function to show all the information of a **BankAcccout** object.
- Function **showBalance() (**for displaying current balance**),**
- Functions **deposit()** and **withdrawal()** of money from an account. Show appropriate messages for invalid amount.
- Function **giveInterst()** will deposit net interest to the account. Default interest is **3 percent** of current balance but it might be different. A fixed **10%** Source Tax will be deducted from the incurred interest.
- When the **BankAccount** object is destroyed display a message like : *Account of Mr. X with account no 1234 is destroyed with a balance BDT 5000*

Non-member functions

Define a non-member function display_stat()(this is not a member function of the BankAccount class) that displays the following information:
Total number of BankAccount objects created and total number of BankAccount objects currently present in the program. It also shows the total amount of source tax collected from all the bank accounts.

Define a non-members function with the prototype
BankAccount Larger(const BankAccount A, const BankAccount B): it returns the BankAccount Object that has a higher current balance.

Task 3

Define a class in C++ with following description:

Private Members:

- A data member EmpName of type String
- A data member ID of type Integer
- A data member BaseSalary of type float
- A data member JoinigYear of type integer
- A member function calculateTotalSalary() to calculate the total salary using the formula:
    - Base Salary + 10% transport allowance + 30% housing allowance + 10% miscellaneous allowance.
    - Every year, the base salary increases by 3%.
- A member function calculateBonus() to calculate the employee's bonus based on their status, years of service, and the updated base salary.
    - 5% for "Low" status.
    - 10% for "Moderate" status.
    - 15% for "High" status.
- A member function getStatus() to **return** the status of an employee on the basis of the following criteria

Age	**Total Salary**	**Status**
<=25	<=20000	Low
	>20000	Moderate
>25	<=21000	Low
	>21000 and <=60000	Moderate
	>60000	High

Public Members:

A function **FeedInfo**(), to set the information of an employee like **EmpName, ID, Base Salary, Joining year.**

A function **ShowInfo**(), to display all the information of an employee.