LAB 04 – TASKS

Instructor: Shafique Rehman

TASK # 01

You are building a student management system. Create a class `Student` to manage student records.

Requirements:

1. Attributes:

- `studentID` (int)
- `name` (string)
- `age` (int)
- `grade` (char)

2. Define a default constructor that initializes `grade` to `'N'` (Not Assigned).

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `promoteStudent()`: Upgrades the student's grade (e.g., from 'A' to 'B').
- `isEligibleForScholarship()`: Returns `true` if the student's grade is 'A'.
- `displayDetails()`: Displays the student's details.

5. Create a few `Student` objects and test the methods.

TASK # 02

Scenario:

A **library system** allows users to borrow and return books. The system should:

1. Add new books to the collection.
2. Borrow books (check availability).
3. Return books.
4. Display all available books.

Requirements:

- Implement a `Book` class with attributes: `ID`, `title`, `author`, `availability`.
- Implement a `Library` class with:
    1. Method to **add a book** (with unique ID).
    2. Method to **borrow a book** (updates availability).
    3. Method to **return a book** (marks it available).
    4. Method to **display all available books**.
- Store book records dynamically using **pointers and DMA**.
- Use **constructor overloading** to initialize books with or without availability status.

TASK # 03

You are building a bank account management system. Create a class `Account` to manage bank accounts.

Requirements:

1. Attributes:

- `accountNumber` (string)
- `accountHolderName` (string)
- `balance` (double)

2. Define a default constructor that initializes `balance` to `0.0`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `deposit(double amount)`: Adds the amount to the balance.
- `withdraw(double amount)`: Deducts the amount from the balance (if sufficient funds are available).
- `checkBalance()`: Displays the current balance.

5. Create a few `Account` objects and test the methods.

TASK # 04

You are building a car rental system. Create a class `Car` to manage cars available for rent.

Requirements:

1. Attributes:

- `carID` (int)
- `model` (string)
- `year` (int)
- `isRented` (bool)

2. Define a default constructor that initializes `isRented` to `false`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `rentCar()`: Marks the car as rented.
- `returnCar()`: Marks the car as available.
- `isVintage()`: Returns `true` if the car's year is before 2000.

5. Create a few `Car` objects and test the methods.

TASK # 05

You are building an employee management system. Create a class `Employee` to manage employee records.

Requirements:

1. Attributes:

- `employeeID` (int)
- `name` (string)
- `department` (string)
- `salary` (double)

2. Define a default constructor that initializes `salary` to `0.0`.

3. Define a parameterized constructor to initialize all attributes.

4. Add methods:

- `giveBonus(double amount)`: Adds the bonus amount to the employee's salary.
- `isManager()`: Returns `true` if the employee's department is "Management".
- `displayDetails()`: Displays the employee's details.

5. Create a few `Employee` objects and test the methods.

TASK # 06

Scenario: A bank wants to develop a system for managing customer accounts. The system should allow customers to:

1. **Create a new account** with an **account number, owner's name, and initial balance** (default balance is 0 if not provided).
2. **Deposit money** into their account.
3. **Withdraw money** from their account, ensuring they cannot withdraw more than the available balance.
4. **Display account details** including account number, owner's name, and current balance.

Your task is to **implement a C++ program** that fulfills these requirements.