Lab 3 – Object Oriented Programming

November, 2018

Document Control

Version	1.0
Updated by	Karthikeyan Jayaraman
Reviewed by	Ajit Jog
Date	November 2018

Table of Contents

1. Getting Started

1.1. Overview

In this lab, you will use Object-Oriented Programming concepts.

1.2 Pre-requisites

- Object-Oriented Concepts
- C++ programming

1.3 Software Requirement

- Visual Studio 2010 or above (Professional or Ultimate edition)

OR

- Visual Studio 2010 or above (Express for Web and Desktop editons)

  Downloadable from

  http://www.visualstudio.com/en-us/products/visual-studio-express-vs.aspx

1.4 Instructions

- Create a folder named **CSharp_Labs** on the local drive of your system.
- The labs for each day must be stored as a Visual Studio solution inside this folder.

2. Object Oriented Programming

2.1. TODO List

- Create an empty solution named **Lab3** inside the **CSharp_Labs** folder created previously.
- Create a new project named **OOP** in the current solution.
- Create a class library project named **LitwareLib** in the current solution.
- Create a hierarchy of **Employee**, **Manager** and **MarketingExecutive**. Create the following private data members in each class with suitable parameterized constructors:

Employee class

empcode	int
empname	string
salary	double
pf	double
tds	double
grosssalary	double
netsalary	double

Manager class (inherited from Employee)

petrolallowance	double
foodallowance	double
otherallowances	double

MarketingExecutive class (inherited from Employee)

kilometertravel	double
tourallowance	double
telephoneallowance	double

- Calculate the allowances for Manager & MarketingExecutive classes based on the following table:

Manager class (calculation is based on % of salary)

PF	TDS	PetrolAllowance	FoodAllowance	OtherAllowances
15%	16%	8%	13%	3%

MarketingExecutive class(PF and TDS is based on % of salary)

PF	TDS	KilometerTravel	TourAllowance	TelephoneAllowance
10%	12%	-	Rs. 5/- per km.	Rs. 1000/-

- Create a method named **ShowDetails()** in the Employee class which must display the following details for each employee type:
  - Employee id
  - Name
  - Salary
  - Provident Fund (as percentage of the salary)
  - TDS(as percentage of the salary)

- Overridde the **CalculateSalary()** method in each derived class, to calculate the GrossSalary and NetSalary based on the following formula:

  Manager class
  grosssalary = (salary + petrolallowance + foodallowance + otherallowance)
  netsalary = grosssalary – (pf + tds)

  MarketingExecutive class
  grosssalary = (salary + kilometertravel + tourallowance + telephoneallowance)
  netsalary = grosssalary – (pf + tds)

- Overridde the **ShowDetails()** method in each derived class to display the basic details and allowances details for each type of employee:

- From the Main() method, create an array of Employee objects. Store the details of Manager and MarketingExecutive in the array. Using a foreach loop, call the **Show()** and **CalculateSalary()** methods of each class to display the details of Manager and MarketingExecutive respectively.

  The client code must be as follows:

  ```
  Employee []allemployees = new Employee[2];
  allemployees[0] = new Manager(100, "ABC", 43000);
  allemployees[1] = new MarketingExecutive(200, "XYZ", 30000, 15);

  foreach(Employee emp in allemployees)
  {
          emp.ShowDetails();
          emp.CalculateSalary();
  }
  ```

  Use this class library in the **InheritancePolymorphism** project by adding a reference.

- Create two interfaces named **ISquare** and **ICube**. Create a method named **Compute()** which takes an integer as an argument and returns an integer, in each interface respectively. Create a class named **SimpleMath** which implements both the interfaces. Create two explicit implementations of the Compute() method and a default implementation of the Compute() method in the **SimpleMath** class. The first explicit implementation must return the square of the number passed and the second explicit implementation must return the cube of the number passed. The default implementation must return the sum of square and cube of the number passed.
  - Create an instance of the **SimpleMath** class from the **Main()** method and call the **Compute()** method to display the square, cube and the sum of square asnd cube of a number. THE NUMBER MUST BE PASSED AS A COMMAND LINE ARGUMENT.