# LAB ASSIGNMENT #2

**Due Date:** **Week 05 (12.30 pm, Friday, 5th October)** **Marks/Weightage: 30/10%**

**Purpose:** The purpose of this Lab Assignment is to:

- Practice the use of instance data members, constructors, methods in classes and objects

**References:** Read the course's text book **chapter 09 – Classes and Objects** and the lecture notes/ppts. This material provides the necessary information that you need to complete the exercises.

**Instructions**: Be sure to read the following general instructions carefully:

This lab should be completed individually by all the students. You will have to demonstrate your solution in a scheduled lab session and submitting the project **through drop box link on e-Centennial**.

You must name your solution according to the following rule:
**FirstName-LastName_SectionNumber_COMP123_Labnumber**
For Example: **John-Smith_Sec002_COMP123_Lab02**

Each exercise should be added as a separate project - named as firstname-last-name_*exercise1*, firstname-last-name_*exercise2* etc. For example – John-Smith_Ex-01, John-Smith_Ex-02 so on.

Submit your assignment in a **zip file** that is named according to the following rule:
**FirstName-LastName_SectionNumber_COMP123_Labnumber.zip**
Example: **John-Smith_Sec002_COMP123_Lab02.zip** *(if your section is 001..)*

Apply the naming conventions for variables, methods, classes, and packages:
- *variable names* start with a *lowercase* character for the first word and uppercase for every other word
- *classes* start with an *uppercase* character of every word
- namespaces use only *lowercase* characters
- *methods* start with an *uppercase* character for the first word and uppercase for every other word

## Exercise #1: *[15 marks]*

Write a C# application using VS 2017 as IDE, that implements the following class(es) as per business requirements mentioned below:

Create a **BasePlusCommissionEmployee** class (***BasePlusCommissionEmployee.cs***) that has the following instance variables:

- Employee ID, First name, last name, base salary, gross sales ( amount in dollars) and commission rate. Define their data types appropriately.
- Define read only property for employee ID, first name, last name and base salary. Ensure the proper ( no negative and null values ) data values by implementing data validations.
- Use default value of 200.00 dollars for base salary for all the employees.
- Define property getters and setters for gross sales and commission rate. Ensure the values for them should never be negative or zero.
- Commission rate should be between 0.1 and 1.0%. Set default value to 0.1.

- Class should have defined two overloaded constructors:
  - One for initializing all the instance data members
  - Second for initializing employee ID, first name, base salary only.
- Define a public method - **double earnings()** which calculates employee's commission ( commission rate * gross sales + base salary )
- Define a public method – **String toString()** which is used to display the object's data

Create a test class – **BasePlusCommissionEmployeeTest** (***BasePlusCommissionEmployeeTest.cs***) which tests above class by at least creating two objects of the BasePlusCommissionEmployee class.

## Exercise #2: *[15 marks]*

Write a C# application that implements the following class(es) as per business requirements mentioned below:

Create a **CheckingAccount** class (*CheckingAccount.cs*) that has the following instance variables:

- Account number , customer name, account balance
- Define read only property for account number and customer name
- Define property getter and setter for account balance. Balance should be positive and minimum 50.00 dollars all the time.
- Class should have defined a constructor:
  - For initializing all the instance data members
- Define one public method - **double withdraw (double amount)** which is used for taking out money. With every withdrawal, there is transaction fee of 3.00 dollars.
- Define a public method – **String toString()** which is used to display the object's data

Create a test class – **CheckingAccountTest** (*CheckingAccountTest.cs*) which tests above class by at least creating two objects of the CheckingAccount class with different set of data values.

**Evaluation:**

| **Functionality** | |
|---|---|
| Correct implementation of classes (instance variable declarations, constructors, getter and setter methods etc.) | 70% |
| Correct implementation of driver classes (declaring and creating objects, calling their methods, interacting with user, displaying results) | 20% |
| Comments, correct naming of variables, methods, classes, etc. | 5% |
| **Friendly input/output** | 5% |
| **Total** | 100% |