Mid-term Exam (TEST #1) – Section: 004

Date: Wednesday 7th March, 2018
Time: 6.30 pm to 8.30 pm
Room Number: A3-15
Marks/Weightage: 50/25%

Student ID: _____

Name of Student: _____

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

This test should be completed individually by all the students. At the start of Visual Studio 2017, you must name your solution according to the following rule:
firstName-lastName_SectionNumber_COMP123_Mid-term.
For Example: **Joh-Smith_Sec004_COMP123_Mid-term**
And project name should be - **Joh-Smith _Exercise01**

You need to zip up your above solution folder application after completion. Submit your test in a **zip file** (**Joh-Smith_Sec004_COMP123_Mid-term.zip**) and upload on to drop box link in e-centennial,

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
- **namespace** use only *lowercase* characters
- *methods* start with an *uppercase* character of every word

Note: Academic dishonesty in any form is not allowed. You are not allowed to talk, share, e-mail, and communicate during the entire duration of the test. You are required to keep your cell phones switched off.

Exercise 1: *[50 marks]*

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

Create an abstract **Mortgage** class (Mortgage.cs) that has the following instance variables:

- **Mortgage number**, **customer name**, **customer address, Mortgage amount**, and **yearly rate of interest**.
- Define properties along with validations for all the above instance data members.
- Mortgage number should only have read only property and should be declared of the type final
- Mortgage amount and interest rate cannot be negative or zero. Interest rate should not be more than 5.0% in any given situation.
- Mortgage class should have defined two overloaded constructors:
    - One for initializing all the instance data members
    - Second for initializing only Mortgage number, customer name, and address
- Declare an abstract public method **double CalculateMonthlyMortgageInstallment()** which is used for calculating monthly Mortgage installment amount.
- Define ToString() to display the object data

Create following **two child classes** of **Mortgage** class:

a) HouseMortgage (HouseMortgage.cs)
b) BusinessMortgage (**BusinessMortgage** .cs)

For **HouseMortgage** class, implement the following:

- Define an instance variable – **property tax** for yearly property tax.
- Define another instance variable –**infrastructure tax** for municipal infrastructure which is a fixed monthly amount (100.00 dollars) added to while calculating monthly **mortgage** installment amount
- Define properties for the above and validations. Property tax and infrastructure tax should not be negative and zero.
- A constructor for initializing all the instance variables
- Overriding the method - **double CalculateMonthlyMortgageInstallment()** which calculates (use formula: **mortgage** amount * rate of interest / 12) monthly mortgage installment and to this installment amount, you need to add property tax and infrastructure tax.
- You need to override ToString() method to display the object's data.

For **BusinessMortgage** class, implement the following:

- Define an instance variable – **business** insurance amount and implement properties with validations for positive value for insurance amount.
- Constructor for initializing all the instance variables
- Overriding the method - **double CalculateMonthlyMortgageInstallment()** which calculates monthly **business mortgage** installment and to this installment amount, you need to add **business** insurance amount
- You need to override ToString() method to display the object's data.

Create a test class – **MortgageTest** (MortgageTest.cs) which tests above classes by at least creating one object each of the HouseMortgage and BusinessMortgage classes and then processing them.

Evaluation:

Functionality	
Correct implementation of classes (instance variable declarations, validations, constructors, properties, methods, class methods etc.)	70%
Correct implementation of test classes (declaring and creating objects, calling their methods, interacting with user, displaying results)	20%
Comments, correct naming of variables, methods, classes, etc.	5%
User Friendly input/output	5%
Total	100%