**Part A**.

1. Use class Time2 from Fig. 10.5 to create a C# class library project that can be used in a console project application (you can cut and paste the code from the book example).
2. Modify the class to include two overloaded methods that can add time to the time objects as follows:
3. one method will add a value of time received as three integers representing number of hours, minutes and seconds to be added.

public void addtime(int h, int m, int s) { ... }

1. the other one will add a value of time received as another Time2 object.

Public void addtime( Time2 atime ) { … }

1. Use the console application from Fig 10.6 to create a console project that will use the class library (will need to create a reference to the class library dll) created in a). and modify the test program to show that time is added correctly. What should your classes do when adding time takes the time over the 23:59;59 max time in a day?

**Part B.**

1. Create a new class Time2 based on Fig. 10.5 but representing the time internally as an array of chars (**char []**) object, two characters for the hours, two characters for the minutes and two characters for the seconds (you do not need a separator for the **internal** representation of time) and create a second class library project with your new implementation of class Time2. (All you need to do is rework the 3 Properties to use the new data representation).
2. Show that the application created in A.c) can use either class library (the one created in A.b)) or the one created in B.a) **without** any change to the test application. Turn in source code and schedule a demo with our TA the week the program is due, to show a test of your program running once using the original class and again using the new class, **both runs should produce the same results** except for maybe the toString and toUniversal string adding a short message indicating which library file is being used (for information purposes). You should run **THE SAME TEST PROGRAM** created in A.c) twice, once using each implementations of Time2 (the old and the new).

**Part C**. Write a derived class **time2ss (time2-stop watch)** based on **either** one of your time2 classes (the original or the new implementation) a value of time that includes milliseconds (0 to 999). You will need to **override the toString and the toUniversalString** methods to return the time including the milliseconds. Your class should include **at least 3 constructors** (a fully parameterized constructor that expects H, M, S and mS; one that only expects H, M, S and defaults to 0 mSec; and one that expects a time2ss object). Turn in your source code and do a demo showing your testing of the class (with a modified test program (from part A.c) where you create several time2 and time2ss objects and shows the polymorphic behavior of the toString and the toUniversalString methods.

**Part D**. Create a new test console application which should do the following:

**The data for the entered time objects should be kept in a list of time2 objects ( List<time2>) so you can use LINQ to run the queries that will produce the reports**

Start with a loop to ask the user to enter data for a number of time2 and time2tz objects. The user should be given the choice to keep entering data or stop after data for each object is entered, something like this:

Which type of object you wish to enter?:

1 – time2

2 – time2ss

3 – Stop entering data

1

Enter Hours:

6

Enter Minutes:

25

Enter Seconds:

50

Which type of object you wish to enter?:

1 – time2

2 – time2tz

3 – Stop entering data

2

Enter Hours:

15

Enter Minutes:

12

Enter Seconds:

24

Enter milliSeconds:

240

Which type of object you wish to enter?:

1 – time2

2 – time2tz

3 – Stop entering data

3

What report do you want:

1 – All objects

2 – All objects with AM times

3 – All objects with PM times

4 - QUIT

Display the list of requested objects in increasing order…

After the report is produced, the user should have the choice to produce another report or quit.

**Extra credit**: (10 Points) Have your classes throw an exception when adding time takes the time value to more than 23:59:59:999 and modify the Test program to handle the exception.

Your programs should follow good software engineering practices, i.e., **all constructors should call the fully parameterized constructor, constructors in the derived class constructor should call the base class constructor appropriately to initialize the base class state, use of properties, if your derived class overrides a method, it should call the base class version of the method to do partial work and OF COURSE use of private variables AND ONLY private variables in your classes**!

Develop all your libraries and main program projects in the same solution. Zip that Solution folder and upload it to blackboard on the due dates before midnight and then schedule a demo with our Teaching Assistant the week after the assignment is due.