Practical Task 2.3

(Credit Task)

Submission deadline: 10:00am Monday, April 6
Discussion deadline: 10:00am Saturday, April 25

General Instructions

In this practical task, you need to implement a class called `MyTime`, which models a time instance. The class must contain three private instance variables:

- **`hour`**, with the domain of values between 0 to 23.
- **`minute`**, with the domain of values between 0 to 59.
- **`second`**, with the domain of values between 0 to 59.

For the three variables you are required to perform input validation. The class must provide the following public methods to a user:

- **MyTime( )**

  Constructor. Initializes a new instance of the `MyTime` class.
- **MyTime (int hour, int minute, int second)**

  Constructor. Initializes a new instance of the `MyTime` class and sets the three instance variables to the respective values given in the input.
- **void SetTime(int hour, int minute, int second)**

  Sets the three instance variables to the respective values given in the input.
- **void SetHour(int hour)**

  Mutator method. Gives the specified value to the instance variable `hour`.
- **void SetMinute(int minute)**

  Mutator method. Gives the specified value to the instance variable `minute`.
- **void SetSecond(int second)**

  Mutator method. Gives the specified value to the instance variable `second`.
- **int GetHour( )**

  Accessor method. Gets the value of the instance variable `hour`.
- **int GetMinute( )**

  Accessor method. Gets the value of the instance variable `minute`.
- **int GetSecond( )**

  Accessor method. Gets the value of the instance variable `second`.
- **String ToString( )**

  Returns a string that represents the current object. `ToString()` is the major formatting method in the .NET Framework. It converts an object to its string representation so that it is suitable for display. This method must return the time stored in the `MyTime` as a string formatted to "HH:MM:SS".
- **MyTime NextSecond( )**

  Updates this instance of `MyTime` to the next second and returns this instance.
- **MyTime NextMinute( )**

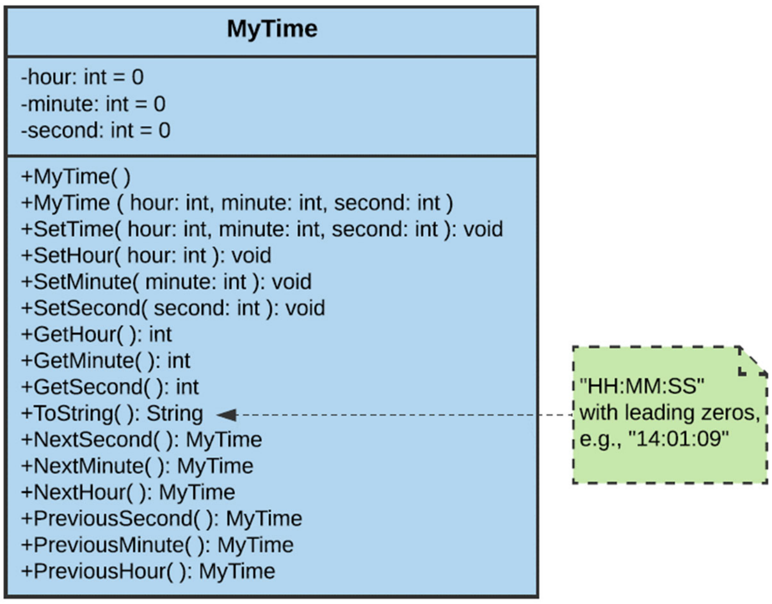
  Updates this instance of `MyTime` to the next minute and returns this instance.
- **MyTime NextHour( )**

  Updates this instance of `MyTime` to the next hour and returns this instance.

- **MyTime PreviousSecond( )**
  Updates this instance of `MyTime` to the previous second and returns this instance.
- **MyTime PreviousMinute( )**
  Updates this instance of `MyTime` to the previous minute and returns this instance.
- **MyTime PreviousHour( )**
  Updates this instance of `MyTime` to the previous hour and returns this instance.

Take note that the mutator methods must check if the given hour, minute and/or second are valid before changing the respective instance variables. If the provided information is invalid, no changes are applied. (Indeed, in practice, it should also throw an `ArgumentException` with the message "Invalid hour, minute, or second!".) You must ensure the correct time after each change you apply, e.g. the `NextSecond()` of 23:59:59 is 00:00:00.

The following UML diagram should help you to understand the design of the class.


MyTime
-hour: int = 0
-minute: int = 0
-second: int = 0
+MyTime( )
+MyTime (hour: int, minute: int, second: int)
+SetTime(hour: int, minute: int, second: int): void
+SetHour(hour: int): void
+SetMinute(minute: int): void
+SetSecond(second: int): void
+GetHour( ): int
+GetMinute( ): int
+GetSecond( ): int
+ToString( ): String
+NextSecond( ): MyTime
+NextMinute( ): MyTime
+NextHour( ): MyTime
+PreviousSecond( ): MyTime
+PreviousMinute( ): MyTime
+PreviousHour( ): MyTime
"HH:MM:SS" with leading zeros, e.g., "14:01:09"

Create a new C# Console Application project and write your code for the `MyTime` class. Also write a test driver, a class called `TestMyTime`, to test all the public methods defined in the `MyTime` class. Indeed, `TestMyTime` is to be the new name of the default `Program` class. Test your program and make sure that your methods perform as expected. Note that you are not allowed to use any native .NET libraries to work with time, e.g. `DateTime`. That is, you must implement the `MyTime` class completely from scratch.

Further Notes

- Study the way to program and use classes and objects in C# by reading Sections 2.3-2.6 of SIT232 Workbook available in CloudDeakin → Resources.
- The following links will give you more insights on classes/objects and relevant topics:
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/classes
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/concepts/object-oriented-programming
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/members
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/methods
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/constructors
  - https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/classes-and-structs/objects

  Because this information is quite detailed and addresses some advanced concepts in OOP, you may mainly focus on the parts you feel relevant to this practical task.

- It is worth to explore the functionality of the native .NET class to represent an instant in time, typically expressed as a date and time of day. Spend time to learn its methods and properties as you will likely need it later in your practice.
    - https://docs.microsoft.com/en-us/dotnet/api/system.datetime
- If you seriously struggle with the remaining concepts used in this practical task, we recommend you to start reading the entire Sections 1 and 2 of SIT232 Workbook.
- In this unit, we will use Microsoft Visual Studio 2017 to develop C# programs. Find the instructions to install the community version of Microsoft Visual Studio 2017 available on the SIT232 unit web-page in CloudDeakin in Resources → Software → Visual Studio Community 2017. You however are free to use another IDE, e.g. Visual Studio Code, if you prefer that.

Marking Process and Discussion

To get your task completed, you must finish the following steps strictly on time.

- Make sure that your program implements the required functionality. It must compile and have no runtime errors. Programs causing compilation or runtime errors will not be accepted as a solution. You need to test your program thoroughly before submission. Think about potential errors where your program might fail.
- Submit the expected code files as an answer to the task via OnTrack submission system. Cloud students must record a short video explaining their work and solution to the task. Upload the video to one of accessible resources, and refer to it for the purpose of marking. You must provide a working link to the video to your marking tutor in OnTrack.
- On-campus students must meet with their marking tutor to demonstrate and discuss their solution in one of the dedicated practical sessions. Be on time with respect to the specified discussion deadline.
- Answer all additional questions that your tutor may ask you. Questions are likely to cover lecture notes, so attending (or watching) lectures should help you with this **compulsory** interview part. Please, come prepared so that the class time is used efficiently and fairly for all students in it. You should start your interview as soon as possible as if your answers are wrong, you may have to pass another interview, still before the deadline. Use available attempts properly.

Note that we will not accept your solution after the submission deadline and will not discuss it after the discussion deadline. If you fail the deadline, you also fail the task and this may impact your performance and your final grade in the unit. Unless extended for all students, the deadlines are strict to guarantee smooth and on-time work throughout the unit.

Remember that this is your responsibility to keep track of your progress in the unit that includes checking which tasks have been marked as completed in the OnTrack system by your marking tutor, and which are still to be finalised. When marking you at the end of the unit, we will solely rely on the records of the OnTrack system and feedback provided by your tutor about your overall progress and quality of your solutions.