Mid-Term Exam

Term: Summer 2023 **Professor:** Gursharan Singh Tatla
Course: PROG32356 – .NET Tech. using C# **Email:** gursharan.tatla@sheridancollege.ca

Please Be Advised That:

1. Exam must be completed as an individual effort. Do not collaborate with anyone or share it with any individual, party or entity.
2. Do not share this exam with anyone or any 3rd party without the written consent of the professor.
3. ZIP the project and upload it to SLATE by due date/time, mentioned in SLATE.
4. All online submissions will be done via SLATE (Email submissions will NOT be accepted).
5. Corrupt/incorrect submissions will be graded as zero.
6. This is a 3-hour exam which will be attempted over the period of 3-days. Therefore:
    a. No late submissions will be accepted as it is an exam.
    b. No 3-day grace period is allowed.
    c. No extensions will be provided to students with accommodations.
7. Make sure your laptops are in good working condition. If something does not work, or any application or system crashes, you will be responsible to fix it and submit your work on time. No excuses will be accepted during the exam.
8. Make sure your Visual Studio is in good working condition. No excuses will be entertained regarding issues with Visual Studio.
9. Please refer to the Academic Integrity Policy.
10. Refer to the School of Applied Computing's Academic Procedures for Evaluations for more details regarding missed work: Procedures for Evaluations.
11. If you take code-snippets from external sources such as StackOverflow, make sure to provide references to the source's webpage as comments in your code.

Copyright Disclaimer:

The materials provided in class and in SLATE are protected by copyright. They are intended for the personal, educational uses of students in this course and should not be shared externally or on websites such as Chegg, Course Hero or OneClass. Unauthorized distribution may result in copyright infringement and violation of Sheridan policies.

Gursharan Singh Tatla (Summer 2023)

Instructions:

1. Make a WPF Application (or WPF App .NET Framework) using C# in Visual Studio and name it as **MTYourFirstnameLastname**.
2. Design and implement an application that keeps track of the employees in a company and their earnings information. The application calculates an employee's wages and allow users to review employee information of any previously added employee.
3. The employees belong to three different categories: hourly, commission-based and weekly salary.
4. The app maintains a generic collection of employees.
5. Use LINQ whenever you need to read or fetch data.
6. ***NOTE:*** At no point should the application save any information to disk. All information is stored and managed in memory using a generic collection. The use of collections and the object-oriented design of the solution are an important part of the evaluation of your submission.

7. Design a window that looks something like the one below. Feel free to change the layout and colors:

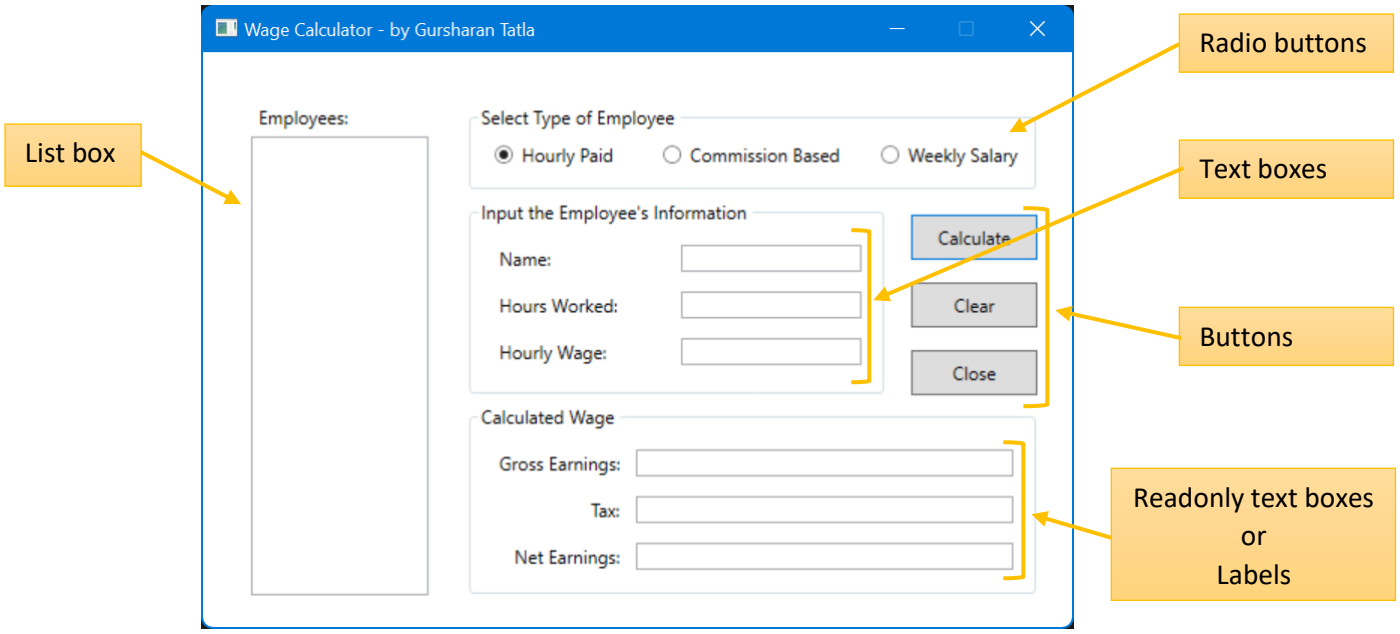

Fig. 1: Window layout

8. User selects a type of employee from the `RadioButtons`.
    - You can set one of the `RadioButtons` as default selected.
9. On selecting "Commission Based" `RadioButton`, the `Labels` "Hours Worked" and "Hourly Wage" should change to "Gross Sales" and "Commission Rate", respectively. (*See sample output at the end of this document*).
10. On selecting "Weekly Salary" `RadioButton`, the `Label` "Hours Worked" should change to "Weekly Salary", and "Hourly Wage" `Label` and `TextBox` should hide.

Gursharan Singh Tatla (Summer 2023)

11. Then, the user can enter the name of an employee, and the earning details.
12. Upon clicking the "Calculate" `Button`, the application shall calculate and display the gross earnings, taxes and net earnings.
    - Assume a federal tax of 20%.
    - An hourly-based employee is paid overtime wages which are calculated as 1.5 times the hourly wage for any hours over 40 hours a week.
    - A commission-based employee is paid the commission based on the gross sales and commission rate.
    - A weekly-salaried employee just earns the fixed weekly salary.
13. After doing the above calculations, add the employee names to the `ListBox`. The `ListBox` shall ONLY display the employee names and no other information.
    - When adding a new employee, do not ask for Employee ID, rather generate a new one. The Employee ID must be unique for every employee, just like a primary key in database.
14. Upon selecting an existing employee, display the wage information associated with it including earnings.
    - The `RadioButton` should be updated according to the type of employee.
    - The name and the following two `TextBoxes` should display the appropriate data.
    - The gross earning, tax and net earning should be displayed in the readonly `TextBoxes`, or `Labels`.

15. The "Clear" `Button` should clear all the fields of the form with the exception of the Employee `ListBox`.
16. The "Close" `Button` should close the window.
17. Use whatever layout panels you want to use: `Grid`, `StackPanel`, `Canvas` or any other layout.
18. Use your own imagination to layout the controls and design the app.
19. Format all the message boxes with appropriate caption, buttons, and image.

Gursharan Singh Tatla (Summer 2023)

Classes:

1. Create a parent class `Employee` which must be declared as `abstract`.
2. Then derive three child classes from the `Employee` class:
   - `HourlyEmployee`
   - `CommissionEmployee`
   - `SalariedEmployee`
3. Use `enum` to represent different types of employees. If you ever need to find what type of employee it is, do not compare it with `string`, rather make use of `enum`.
   - **Do:**
     ```
     if (employeeType == EmployeeType.HourlyEmployee)
     ```
     Where, `EmployeeType` is declared as an `enum`.
   - **Don't:**
     ```
     if (employeeType == "HourlyEmployee")
     ```
4. The parent class `Employee` has fields which are common to all employees, such as:
   `EmployeeType`, `EmployeeId`, `EmployeeName`
5. The parent class `Employee` also declares an abstract property `GrossEarnings`, which will be overridden in the derived classes.
   - *GrossEarnings property must only have `get` accessor.*
   - *It only returns the gross earnings. There is no point setting the value for this property.*
6. A property `Tax` (with only `get` accessor), that calculates and returns the 20% tax.
7. A property `NetEarnings` (with only `get` accessor), that returns the net earnings after tax deductions.

Gursharan Singh Tatla (Summer 2023)

8. Class `HourlyEmployee` inherits from `Employee`:
   - This class stores hours worked and hourly wage.
   - Implements the `GrossEarnings` property that returns the earnings of this employee, considering any overtime at the rate of 1.5.
     Gross Earnings:
     *if hours <= 40 then wage * hours*
     *if hours > 40 then 40 * wage + (hours - 40) * wage * 1.5*
9. Class `CommissionEmployee` inherits from `Employee`:
   - This class stores the gross sales and commission rate.
   - Implements the `GrossEarnings` property that returns the earnings of this employee.
     *Gross Earnings = gross sales * commission rate*
10. Class `SalariedEmployee` inherits from `Employee`:
    - This class stores the fixed weekly salary.
    - Implements the `GrossEarnings` property that returns the earnings of this employee.
      Gross Earnings = fixed weekly salary

11. Use your Java and object-oriented programming knowledge to create this hierarchy.
    - **NOTE:** Use C# properties to get/set the fields. Do not use getter/setter methods like you would do in Java.
    - You will get higher grades if:
        i. Proper use of inheritance.
        ii. User-Interface is user-friendly.
        iii. App runs smoothly without crashing.
    - Otherwise, there will be grade deductions.

Gursharan Singh Tatla (Summer 2023)

Exception Handling:

1. There should be no unhandled exceptions of any kind at run-time.
2. Display all errors resulting from the input validation using `MessageBox`es. Validate the input as follows:
    a. All fields are required (radio buttons, name, hours worked and hourly wage).
    b. The hours worked has to be a positive real number and cannot exceed 168.
    c. The hourly wage, gross sales, commission rate and weekly salary fields have to be a positive real number.
3. The error messages should be clear and understandable.
    a. Something like, “Name field can not be left blank”.
    b. Not like “Invalid input”, or “Input was not in a correct format”.

Gursharan Singh Tatla (Summer 2023)

Submission:

1. Once done, ***ZIP the solution folder*** and upload it to **Assignments** on SLATE.
    a. Make sure to ZIP the whole folder, not just .SLN file.
    b. If only .SLN file is zipped and submitted, Grade 0 will be given.
    c. ***Double-check your submission by downloading it and running it.***
2. You are to submit .ZIP and .TXT files, separately:
    a. Upload the **.ZIP** file of your assignment to SLATE.
    b. Upload the **.TXT** files to SLATE.
3. You must copy and paste all of your source code from your C# files into separate plain text files.
    a. You can copy and paste the source code into Notepad.
    b. Create a separate .TXT file for each source code file.
    c. You do not need to copy the XAML source code. Only copy/paste the .CS source code.
4. You don't have to format this code - it's used by TurnItIn (the originality checker in SLATE, which is a piece of software that checks your submission for plagiarism against other submissions in the college, in other colleges, from the web, and various other sources).
5. Submit these text files in addition to your assignment ZIP file.
    a. DO NOT add them inside your zip/rar file – they must be a separate file.
    b. These are used for TurnItIn (it won't examine the contents of zip/rar files).
6. **Note:**
    a. If these submission instructions are not followed, Grade 0 will be granted.
    b. If TXT files are not provided, Grade 0 will be granted.

Sample Output:

https://www.loom.com/share/8533fe0a41c64815903676cd62f15460