Sheridan | Get Creative

Assignment #1
PROG32356 – .Net Technologies Using C#
Instructor: Sukhbir Tatla

Instructions:

1. The assignment must be completed as an individual effort.
2. Submit the assignment on SLATE before the due date which is specified on SLATE.
3. All online submissions will be done via SLATE (Email submissions will NOT be accepted).
4. Late assignments will be penalized 10% each day for up to 3 days. After that, it is worth zero.
5. This assignment is of 20%.
6. Create a solution file in neat, clean format and properly explained.
7. Corrupt/incorrect submissions will be graded as zero.
8. Please refer to the Academic Integrity Policy

Assignment Instructions:

1. Make a .NET Console App using C# in Visual Studio and name it as **A1YourFirstnameLastname**.
2. Implement the wage calculator for employees which can do the following:
   - Add, edit, and delete employees.
   - View and search employees.
3. The application must be menu-based app:

   1 - Add Employee
   2 - Edit Employee
   3 - Delete Employee
   4 - View Employees
   5 - Search Employee
   6 - Exit
4. When selecting Add, Edit or Delete Employee option from the above menu, display a sub-menu which asks which type of employee is being added, edited, or deleted.
5. For example, if user selects *1 - Add Employee*, then display the following sub-menu:

   1 - Add Hourly Employee
   2 - Add Commission Employee
   3 - Add Salaried Employee
   4 - Add Salary Plus Commission Employee
   5 - Back to Main Menu

6. The menu should run continuously until the user quits the application.
   - Validate the inputs. Any invalid input should not crash the app.
   - Display user-friendly messages wherever necessary.
   - Always display the employee info in a tabular form.
   - Always display column headers when displaying employee info.
   - Format the numeric values with appropriate symbols, such as currency amount with $ symbol, commission rate with % symbol.

7. ***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.

8. **Add Employee:**
   - When adding a new employee, ask the user for all the required info and then save it to the collection.
   - Then display a success message and print all the employees from that type of employee category in a tabular form to verify that the new employee is added.
   - **NOTE:** 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.

9. **Edit Employee:**
   - When editing an employee info, first display a list of all employees from the selected category.
   - Then ask the user for the Employee ID of the employee that user wants to edit and fetch that employee from the collection.
   - And ask the user for all the required info and then display a success message.
   - Print all the employees from that category in a tabular form to verify that the employee is edited.
   - **NOTE:** Do not ask the user to edit the Employee ID. It must be unique for every employee and must not be edited.

10. **Delete Employee:**
    - When deleting an employee info, first display a list of all employees from the selected category.
    - Then ask the user for the Employee ID of the employee that user wants to delete and fetch that employee from the collection.
    - And remove the employee from the collection and then display a success message.
    - Print all the employees from that category in a tabular form to verify that the employee is deleted.

11. **View Employees:**
    - Display all the employees in a tabular form.
    - Display column headers.
    - Format currency and percentage amounts with appropriate symbols.
    - Categorize the employees based on their employment type.

12. **Search Employees:**
    - Ask the user to enter the employee's name and display all the employees that match the search keyword.
    - Partial match must fetch the results.
    - Match should be case-insensitive, meaning if user enters lowercase "john" and the employee's name is "John" (with uppercase J), it must fetch the employee.

13. The employees belong to four different categories: hourly, commission based, fixed weekly salary, and fixed weekly salary plus commission based.

14. Create an employee class Employeewhich must be declared as abstract.
15. Then derive three child classes from the Employeeclass:
    - HourlyEmployee
    - CommissionEmployee
    - SalariedEmployee
16. Inherit the SalaryPlusCommissionEmployeefrom the CommissionEmployeeclass.
17. 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, EmployeeTypeis declared as an enum.
    - **Don't:**

      ```
      if (employeeType == "HourlyEmployee")
      ```

18. The parent class Employeehas fields which are common to all employees, such as:
    EmployeeType, EmployeeId, EmployeeName
19. The parent class Employeealso declares an abstract property GrossEarnings, which will be overridden in the derived classes.
    - *GrossEarnings property must only have getaccessor.*
    - *It only returns the gross earnings. There is no point setting the value for this property.*
20. A property Tax(with only getaccessor), that calculates and returns the 20% tax.
21. A property NetEarnings(with only getaccessor), that returns the net earnings after tax deductions.

22. Class HourlyEmployeeinherits from Employee:
    - This class stores hours worked and hourly wage.
    - Implements the GrossEarningsproperty 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*

23. Class CommissionEmployeeinherits from Employee:
    - This class stores the gross sales and commission rate.
    - Implements the GrossEarningsproperty that returns the earnings of this employee.
        *Gross Earnings = gross sales * commission rate*

24. Class SalariedEmployeeinherits from Employee:
    - This class stores the fixed weekly salary.
    - Implements the GrossEarningsproperty that returns the earnings of this employee.
        Gross Earnings = fixed weekly salary

25. Class SalaryPlusCommissionEmployeeinherits from CommissionEmployee:
    - This class stores the fixed weekly salary.
    - Implements the GrossEarningsproperty that returns the earnings of this employee.
        *Gross Earnings = weekly salary + gross sales * commission rate*

26. 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 menu runs smoothly without crashing.
    - Otherwise, there will be grade deductions.

27. In the Program.cs file, create and maintain a generic collection of all employees.
    - The generic collection must be of type parent class Employeeand it must store objects of derived classes.
    - ***Populate this collection with some sample data when the program runs.***

Submission:

Once done, ZIP the entire solution folder and upload it to Assignments on SLATE. **Double-check your submission by downloading it and running it.**

1. You are to submit 2 files, separately:
    a. Upload the .ZIP file of your assignment to SLATE.
    b. Upload the .TXT file to SLATE.
2. You must copy and paste all your source code from your C# file into a plain text file.
    a. You can copy and paste the source code into Notepad.
    b. You don't have to format this code - it's used by TurnItIn (the originality checker in SLATE,
    c. 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).
3. Submit this document in addition to your assignment ZIP file.
    a. DO NOT add it inside your zip/rar file - it must be a separate file.
    b. This is used for TurnItIn (it won't examine the contents of zip/rar files).

Reference:

- I've used NuGet package ***ConsoleTables*** to print the output in the tabular form.
- Link: https://github.com/khalidabuhakmeh/ConsoleTables