# Setup:

You will be creating a new Visual Studio project named "FirstnameLastname_CE07" using the C# Console Application template.

# Grading:

Please refer to the CE:07 tab of this rubric to understand the grading procedures for this assignment.

# Deliverables:

You will compress and upload a file named "FirstnameLastname_CE07.zip" with the following contents with the following names:

- Project - Folder containing your entire project and all files necessary to build the project(csproj, .cs files, App.config, and Properties folder).

Be sure to upload the correct project and all required files the first time as only one submission will be allowed. No extra time or consideration will be given if the wrong files are uploaded.

# Instructions:

For today's lab you will be creating a program that will act as basic employee tracking software. The program will consist of multiple connected classes. Employee will serve as a base class that contains fields for name and address. Hourly will inherit from employee and contains decimal fields for payPerHour and hoursPerWeek. FullTime will inherit from Hourly. PartTime will inherit from Hourly. Contractor will inherit from hourly and contains a float field for noBenefitsBonus. Salaried will inherit from Employee and contains a decimal field for salary. Manager will inherit from Salaried with a decimal field for bonus.

Main will contain a list of employees and present a menu to the user that offers the following options: Add employee - lets the user to choose to create an employee of type (FullTime, PartTime, Contractor, Salaried, or Manager) and prompts the user for the appropriate values, Remove Employee - Allows the user to select an existing employee and remove her/him from the list, Display Payroll - displays all of the employees (1 per line, using properties to access name / address from

employee and the CalculatePay method for the employee's yearly pay), Exit - allows the user to exit the program. The program will continue running until the user chooses to exit.

Input must be validated. The user must not be able to crash your program with invalid input. For all methods the user should be able to make a selection either by number 1,2,3… or by typing out the option "add employee". String values should be case insensitive so that "AdD EMPloyEE" works just like "add employee".

Use the following guidelines to complete this application:

**Classes**

- Create a class called Employee
    - Contains the following fields
        - name of type string
        - address of type string
    - Implements IComparable interface
- Create a class called Hourly
    - Inherits from Employee
    - Contains the following fields
        - payPerHour of type decimal
        - hoursPerWeek of type decimal
- Create a class called FullTime
    - Inherits from Hourly
- Create a class called PartTime
    - Inherits from Hourly
- Create a class called Contractor
    - Inherits from Hourly
    - Contains the following fields
        - noBenefitsBonus of type decimal - contractors tend to be paid 10 to 15 % more than normal employees because they do not receive benefits.
- Create a class called Salaried
    - Inherits from Employee
    - Contains the following fields

        - salary of type decimal - salaried employees get a lump sum each year no matter the number of hours worked.
- Create a class called Manager
    - Inherits from Salaried
    - Contains the following fields
        - bonus of type decimal - managers are salaried but usually receive a yearly bonus in addition to their normal pay.

**Constructors**

- Employee needs a constructor that takes parameters for name and address
- Hourly needs a constructor that takes parameters for name, address, payPerHour, and hoursPerWeek.
    - Name and address need to be passed to the Hourly's parent constructor.
- FullTime needs a constructor that takes parameters for name, address, and payPerHour.
    - Name, address, payPerHour, and a constant 40 for hours need to be passed to FullTime's parent constructor.
- PartTime needs a constructor that takes parameters for name, address, payPerHour, and HoursPerWeek.
    - All of the parameter values need to be passed to PartTime's parent constructor.
- Contractor needs a constructor that takes parameter for name, address, payPerHour, hoursPerWeek, and noBenefitsBonus.
    - All of the parameters except noBenefitsBonus need to be passed to Contractor's parent constructor.
- Salaried needs a constructor that takes name, address, and salary
    - Name and address need to be passed to Salaried's parent constructor.
- Manager needs a constructor that takes name, address, salary, and bonus.
    - Name, address, and salary need to be passed to Manager's parent constructor.

**Menu Options**

- The menu must have the following options:
    - Add Employee - this option presents the user with the option of creating an employee of the following types: FullTime, PartTime, Contractor, Salaried, and Manager. The last thing this option should do is call .Sort() on the list of employees.
    - Remove Employee - this option presents the user with a list of all employees in the list and allows the user to select 1 to remove.

    - Display Payroll - this option displays all of the employees in the list showing name, address, and yearly pay (Calculated using CalculatePay method).
    - Exit - stop the program.

**Other Items**

- At the top of main you will need to create a list of Employees
- Employee must implement the IComparable interface
    - CompareTo method implemented to sort employees by name
- CalculatePay
    - Employee must contain a virtual method CalculatePay
    - Other classes must override CalculatePay so that it returns the correct value for what that employee would make in a year.
        - Contractor - income is base amount + base amount * noBenefitBonus
        - Manager - income is base amount + bonus

**Input Validation**

- Input must be validated
    - The user should not be able to crash your program
    - Selections can be make using an index 1,2,3 or by typing the associated string like “add employee”.
    - Input should be handled in a case insensitive manner.  ex “ADD EMploYee” and “add employee” work the same way.

**Extra Information**

Go back through your code and check for the following:

- All variables and methods are named appropriately.
- Any information being output to the user should be clear and concise.
- The user should be clearly informed of what is occurring throughout the application. When values change or objects are instantiated information about this occurrence should be displayed.
- Make sure nothing accesses an object that doesn’t exist.