# CS3160 Midterm Exam Supplement

# DUE: Tuesday, March 9, 2021, by 11:59 pm

## Submitter's Information

Student Name: Sidney Sanders

## Purpose and Description

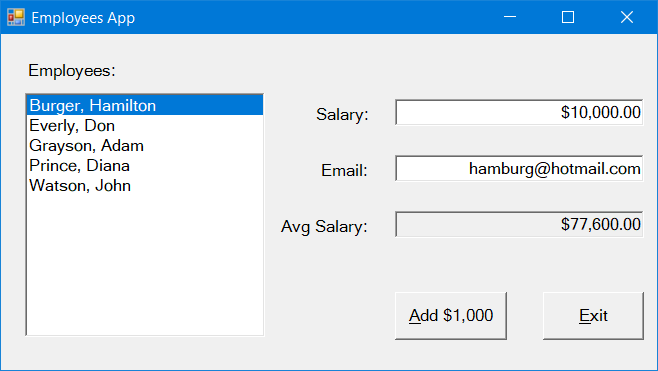
This assignment is a supplement to the midterm exam, and its purpose is to provide an additional opportunity for students to demonstrate mastery of the material. Submission is optional. However, if submitted, the following must be provided:

1. This document with answers to questions 1-5.
2. A working WinForms application solution (compressed .zip file). If the solution does not compile or work correctly, some answers to questions 1-5 may not be accepted since they must be coming from the working solution.

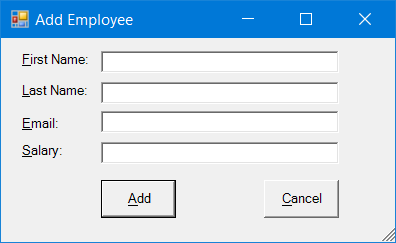
### Instructions for the WinForms application

Implement a WinForm application using C# and Visual Studio 2019 as described below.

1. The application's main form should be designed to contain the controls pictured below (ListBox, TextBoxes, Buttons, and Labels~~). Note that users cannot type into the average salary text box, but they can provide new salary and email~~. When that happens, Employee objects' internal storage must be updated with the new information via Employee class properties Salary and Email. These properties should be validating the user's input and throwing ApplicationExceptions if invalid input is provided.



1. ~~Implement the Employee class that stores employee's first, last names, salary, and email. Provide an overloaded constructor to set internal data members corresponding to these pieces of information~~. This class should implement Salary and Email properties both with set/get components. ~~This class should provide a human-readable version of an employee object, which is their Last, First name string (see the ListBox in the form).~~
2. ~~The main application Form1 class should include a private data member, which should be an Array storing employee objects~~. ~~Allocate memory for the Array in the form constructor and be sure the Array size is 50~~. ~~Also, hard-code five employee objects with data of your choice~~. ~~These should be added to the internal data member array at that time~~. ~~In the form load event, traverse this array and add objects to the ListBox~~. ~~Be sure to use~~ ***~~foreach~~*** ~~construct here~~. Call form’s helper method calculates the average salary of all employees in the internal array data member.
3. When the user click’s on the “Add $1,000” button, add 1,000 to the current salary of the employee selected in the ListBox. Similarly, if the user enters a new salary or email, update the corresponding employee’s data. Be sure to use Employee’s class properties to do the updates. Be sure you are updating the employees stored in the form’s array of Employees. When salary changes, the average salary needs to reflect that change.
4. Use exception handling to recover gracefully when invalid data is provided.
5. ~~Use “C” formatting to show currency values~~.
6. ~~Use decimal data type to handle the salary~~.
7. ~~Incorporate an Employee Add button on the main form. Design Form2 as shown below. Display the form to gather new employee data~~. When user clicks on Add, set this.DialogResult = DialogResult.OK, and DialogResult.Cancel otherwise. Store new employee object in a private data member. Provide a property to retrieve this member from the main form class.



### Questions and Answers

1. Give a statement that will clear out the txtEmail text box.

txtEmail.Clear();

1. In the form above, the average salary text box is “grayed” to prevent someone from changing the value displayed. Write C# code to do that programmatically?  (That is, how do you make a text box UN-editable?)

txtAvg.Enabled() = false;

1. Write statements that would (1) add $1,000 to the salary of the currently-selected employee (Burger, Hamilton) and (2) update both the associated Employee object and the salary textbox.

Don’t worry about updating the average salary.  Remember that the salary should be displayed in the textbox as a currency value.

Clearly mark your answers to (a) and (b).

1. This question has two parts (a) and (b). Clearly indicate your answers for (a) and (b).

Declare a private array data member in Form1 class called mEmployeeList that will hold the Employee objects.  Allow up to 50 employee objects to be saved in the array. Do not allocate memory for this array.

(a) \_\_\_\_\_Employee[] mEmployeeList;\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [ 2 points ]

Write a complete private helper function for Form1 class called calculateAverage( ).  This function should calculate the average salary by iterating through the Employee objects stored in the mEmployeeList array.  Return the new average salary (decimal type) as the value of the function. You **MUST** use foreach in your solution.

(b) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ [8 points ]

private decimal calculateAverage()

{

decimal total = 0;

decimal avg = 0;

int count= 0;

foreach(Employee em in mEmployeeList)

{

decimal num = decimal.Parse(em.Salary);

total += num;

count++;

}

avg = total / count;

return avg;

}

1. The form shown above, Form2, is displayed when a new Employee object is to be created.  Part of the event handler **btnOK\_Click** is given below.  Complete the handler implementation to create a new Employee object using the data entered into Form2 by the user (Assume txtFirst, txtLast, txtSalary, txtEmail are the names of controls) and save the resulting object in this.mNewEmployee array of Employees.  If an exception is thrown when creating the new Employee object, just display the error contained in the exception and return from the function. Assume Employee class has an overloaded constructor accepting ***first, last, salary, email*** as its string parameters. **You must use Exception handling in your solution to receive credit.**

private void btnOK\_Click(object sender, System.EventArgs e){

**// provide code to be written here**

    this.DialogResult = DialogResult.OK;

    this.Close();

}

try

{

newEmp = new Employee(txtFirst.Text, txtLast.Text, txtEmail.Text, Decimal.Parse(txtSalary.Text));

}

catch(Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

1. Write C# code to add **Salary** property for the Employee class.  Throw an *ApplicationException*if there is an attempt to make the salary negative.  Be sure to show the declaration of any other variables you use in defining the property. Assume private data member storing salary is named *mSalary*.

private void txtSalary\_Leave(object sender, EventArgs e)

{

if (decimal.Parse(txtSalary.Text) < 0)

{

throw new ApplicationException("Invalid Sallary");

}

else

{

mEmployeeList[lstEmployee.SelectedIndex].Salary = txtSalary.Text;

}

}

1. Give statement(s) that should be put into the Click event handler for **btnExit** to close the app.

this.Close();

1. The employee names in the list box are shown in format "last, first."  Describe what causes those names to appear that way.

In the class file, you can override the To\_String format to display the format you desire. In my ovveridded To\_string I returned this.lName + ", " + this.fNam which will print the last name comma first name to the list box.

1. This question has two parts: (a) and (b). Clearly indicate your answers for these two parts.

(a) Declare an array variable for an array capable of holding ***Employee*** objects.

(b) Allocate memory for 100 ***Employee*** objects.

(a)Employee[]emp;

(b)emp = new Employee[100];