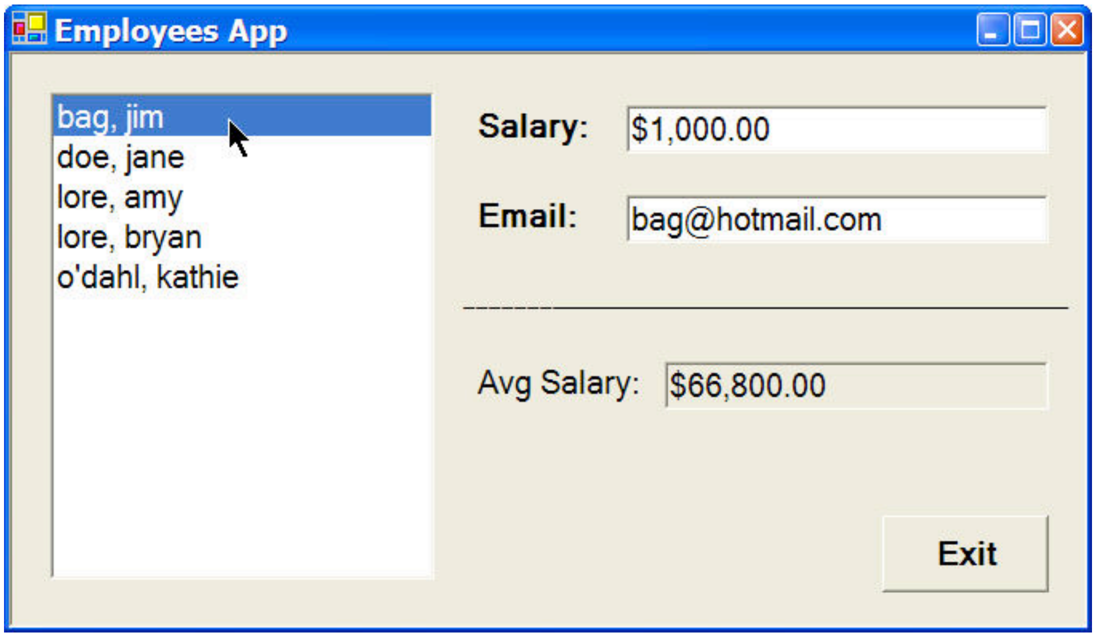
WinForms Lab

(developed by Joe Hummel, modified by Shereen Khoja)

Today you'll work in Visual Studio .NET to create your first form-based, Windows GUI application. Your GUI app will display information about imaginary employees.

Part 1: Designing the GUI



1. Create a new Visual C# Project using the Windows Application template. NOT a console application.
2. The first step is to design the GUI. The empty form should be visible in Visual Studio. Click anywhere on the form to select it, and then press F4 to view the Properties window. Set the form's Text property (i.e. what is displayed in the title bar) to "Employees App". Then set the FormBorderStyle property to "Fixed3D" so the user cannot resize the form, and set the StartPosition property to "CenterScreen" so the form is centered when first shown. Here's a nice trick: if you set the form's Font property, the controls will inherit this setting when they are dropped on the form! Go ahead and set the form's Font property to 12pt, and perhaps Bold as well.
3. Next, drag-and-drop the necessary controls from the Toolbox onto the form as shown above: a list box, 3 labels, 3 text boxes, and a command button. The line you see is actually another label with its Text property

set to lots of underscores. Set the list box's name to lstEmployees (that's lower case L); a control's name property is at the top of the Properties window list displayed as (Name). Then set the button's name to cmdExit, and set the names of the text boxes to txtSalary, txtEmail, and txtAvgSalary, respectively. For each of the label's, set their Text property appropriately. For each of text boxes, set its Text property to empty so nothing is displayed in the text box when the form first appears. For the txtAvgSalary text box in particular, also set its ReadOnly property to True so that the user will be unable to modify this value. Finally, change the Text property of the command button to "Exit".

4. Click on the form's background to select the form, and then drop-down the View menu and select "Tab Order". Click on the controls in the order you want the focus to move when the user presses the tab key; when you're satisfied, drop-down the View menu again and deselect "Tab Order".
5. Save your work. Now run (F5). While the app doesn't do anything yet, it should appear centered, and as you repeatedly press the tab key, the focus should move around the form appropriately. Make sure you cannot resize the form.

Part 2: Programming the GUI to Display Employee Names in List Box

1. First, code the Exit button to end the application. Double-click on the button to reveal its Click event method, and then code this method to end the application by programmatically closing the form via this.Close(); . Run and test the button.
2. Next, let's create an Employee class for holding info about each employee. Add a new class file to your project via the Project menu; Visual Studio will generate some boilerplate code for the class which you can either delete, or just ignore. Define each employee to have a first name, last name, email address, and salary; these are all strings except for the salary, which should be a decimal field (C# has a data type called "decimal"). There is some code below that you can copy and paste into your program, and the code assumes you have implemented a constructor with parameters in that same order. Also implement a ToString() method that returns the employee's full name in the format "lastname, firstname". For part 3, you will need three additional methods, GetEmail() to return the email address, GetSalary() to return

the salary as a decimal, and GetFormattedSalary() to return the employee's salary as a nicely-formatted string like this:

```
public string GetFormattedSalary()
{
   return this.Salary.ToString("$#,##0.00");
}
```

Select "Build Solution" from the Build menu and fix any syntax errors.

3. Now let's create some employee objects and display them in the list box. Since we haven't talked about file I/O yet, we'll simply create the employee objects by hand with predetermined values. Double-click on the form to reveal the form's Load event method. Outside and just above this method, define a private member field employees of type Employee[]. Keep in mind that the form is a class, so it is perfectly legal for us to define fields and methods within the form; the employees field will be a reference to an array that keeps track of the employees for the duration of the application's execution. Now, inside the form's Load event method, just copy and paste this code:

```
this.employees = new Employee[5];
employees[0] = new
Employee("jim","bag","bag@hotmail.com",1000.00M);
employees[1] = new
Employee("jane","doe","doe@aol.com",5000.00M);
employees[2] = new
Employee("amy","lore","amy@hotmail.com",123000.00M);
employees[3] = new
Employee("brian","lore","brian@hotmail.com",122999.99M);
employees[4] = new
Employee("kathie","o'dahl","kathy@hotmail.com",82000.00M);
```

4. Build and fix any syntax errors. Once you have the array built, at the bottom of the Load event method simply iterate through the array and add each employee to the list box using foreach. Run and behold! You should now see all your employees listed in the list box.

```
foreach( Employee emp in employees )
{
 lstEmployees.Items.Add( emp );
}
```

5. The list box display is not sorted. Stop the application, click on the list box, press F4 to view the Properties window, and set the list box's Sorted property to True. Run again, and the attendees should now be displayed in sorted order.

Part 3: Displaying Employee Information

1. The next step is to view employee information when an employee is selected in the list box.
2. Double-click on the list box to reveal its SelectedIndexChanged event method. This event is triggered whenever the user selects an item in the list box (in other words, whenever the underlying index of the selected item has changed). The list box lstEmployees has a property (field) called SelectedItem, which is either (a) null if nothing is selected, or (b) a reference to the selected object. Type-cast the contents of this property into a local variable of type Employee. If the value is null, clear the contents of each text box, otherwise fill the text boxes with the employee's info. When filling a text box, recall that you want to assign to the text box's Text property; in the case of an employee's salary, use the employee's GetFormattedSalary() method so the text box displays their salary in a formatted way.

```
Employee tempEmp = (Employee) lstEmployees.SelectedItem;
if( tempEmp == null )
{
     txtSalary.Text = "";
     txtEmail.Text = "";
}
else
{
     txtSalary.Text = tempEmp.GetFormattedSalary();
     txtEmail.Text = tempEmp.GetEmail();
}
```

3. Run and test, and you should now be able to click on different employees and view their information in the text boxes.
4. Finally, in the Form class, write a private helper method that loops through the array of employees, calculates the average salary, formats it as a string, and displays the string in the txtAvgSalary text box. Call this method from the Load event method, run and test.