Clock in – Clock out machine

Windows forms app project

I am making this program to improve my C# skills as it's an idea which I believe I can execute with my skillset while still being able to learn new skills and aspects of C# from.

Since I am doing this project to better myself, I do not have a client. Therefore I am setting requirements for myself and assuming the program will be used by myself (it won't)

I'm going to be setting all the requirements from the start and obviously there will be no conversing with a client as the client is essentially myself for the purpose of practice.

So… I've decided to use a waterfall methodology for this project

Like mentioned prior, I will be starting out with the requirements

Requirements

There must be a clock which displays 24hr time within the form

There must be buttons to allow a clock in/out

If an employee has been clocked in for over 8 hours, they are then payed overtime pay (*1.2)

There must be variables for an employee which has attributes of age and payRise

There must be a keypad/keyboard so the form can identify/validate an employee

Validation should take place, if invalid clocks are inputted an appropriate error message must be returned.

Employees must have at least 10 hours between each shift, if they attempt to clock in less than 10 hours after their previous shift ended then the clock should return invalid.+

Breaks for employees are unpaid, so there must be buttons to clock in and out of breaks

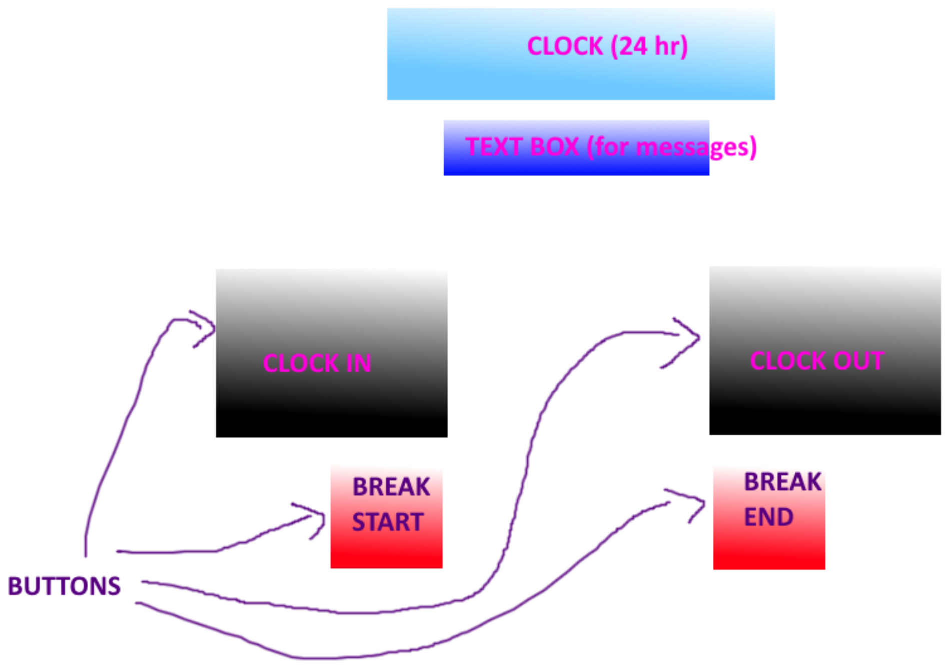
Employee breaks are 30 minutes, if an employee takes a break which exceeds 40 minutes there should be an appropriate message displayed

The UI must be easy to understand; fonts must be readable buttons must be sized appropriately etc

Design

Since windows forms are designed to be interacted with by a user, the form needs a graphical user interface.

FORM WINDOW INITIAL DESIGN



here’s the first idea of the UI which incorporates the features mentioned in the requirements section, this is not how the final UI will look however this give a representation of what it will look like.

Now time for the object orientated section of the project

Making the project object orientated makes the code encapsulated which makes it reusable

There will be an class for employee

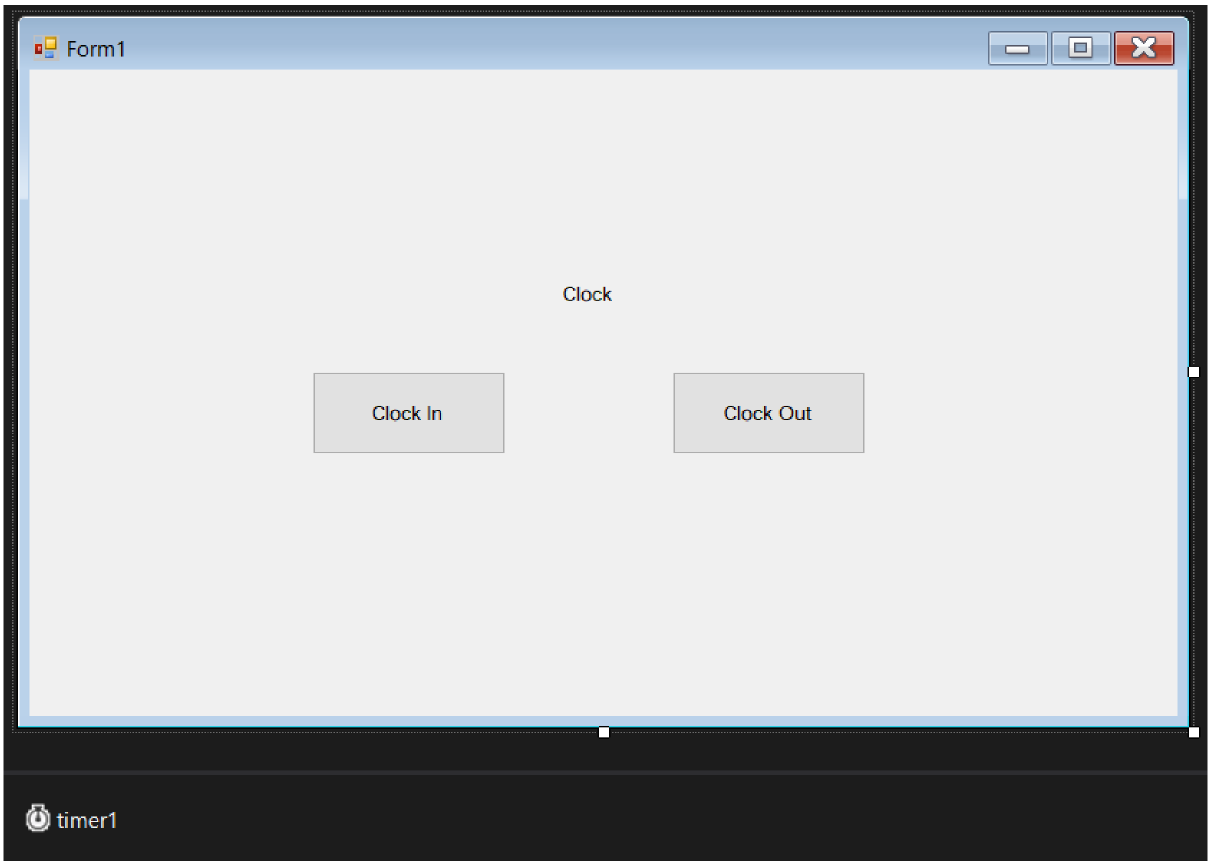
Employee will have the different employees as its objects with their names listed in the format fnameLname

The objects will carry attributes of age and payRise

Age will be an integer value

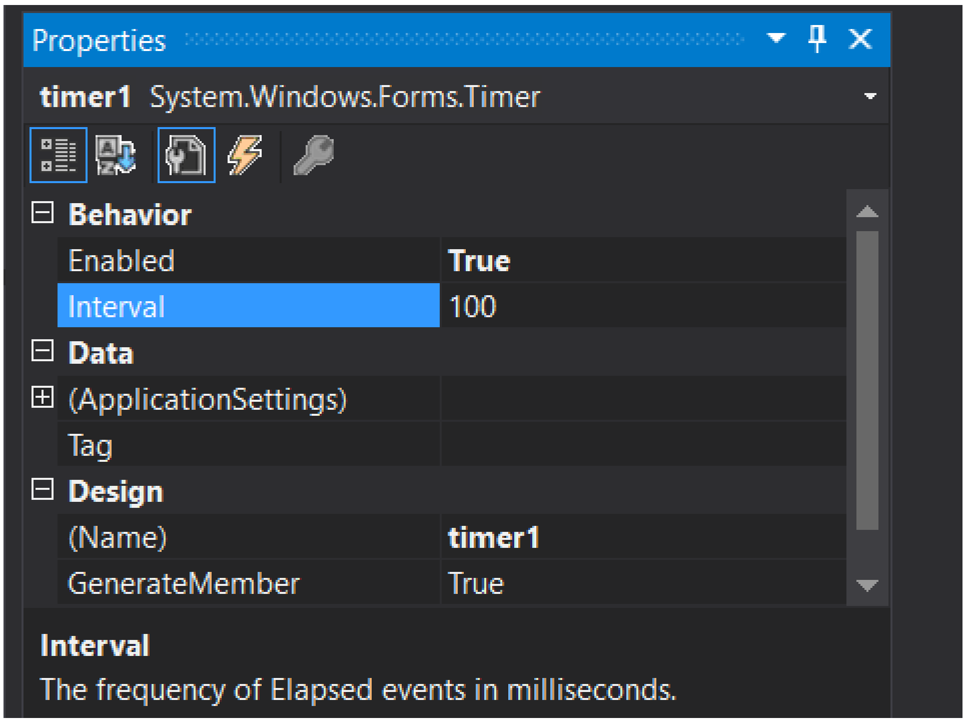
The age attribute will correlate the hourly pay of the employee

Methods for an employee will be all the functions from the buttons such as: clockIn, clockOut, startBreak, endBreak etc



here is the initial design I created for starting the implementation clockIn and clockOut are buttons where the Clock is a label

timer1 is added for ticks, I had to enable the timer to allow it to tick, the ticks are set as 100ms as default – meaning 1 tick per second.


Properties

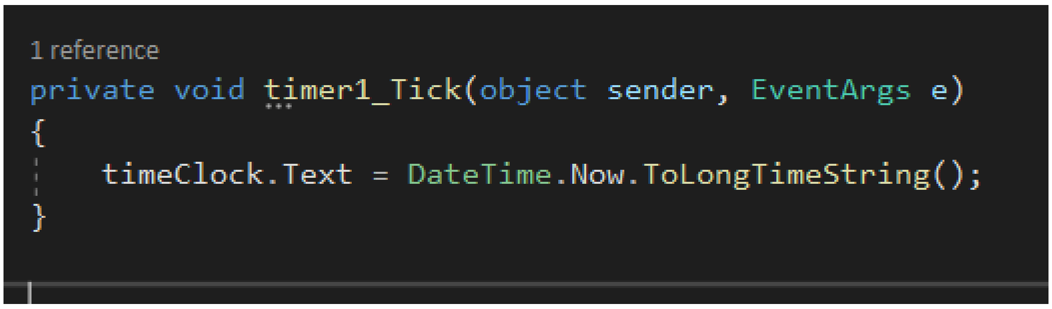
timer1 System.Windows.Forms.Timer

Behavior	
Enabled	**True**
Interval	100
Data	
(ApplicationSettings)	
Tag	
Design	
(Name)	**timer1**
GenerateMember	True

Interval
The frequency of Elapsed events in milliseconds.

The timer is what will allow the clock label to change time every second by itself

Then I double clicked the timer which would open a sub procedure for it so I could write code for a realtime clock.



```
1 reference
private void timer1_Tick(object sender, EventArgs e)
{
    timeClock.Text = DateTime.Now.ToLongTimeString();
}
```

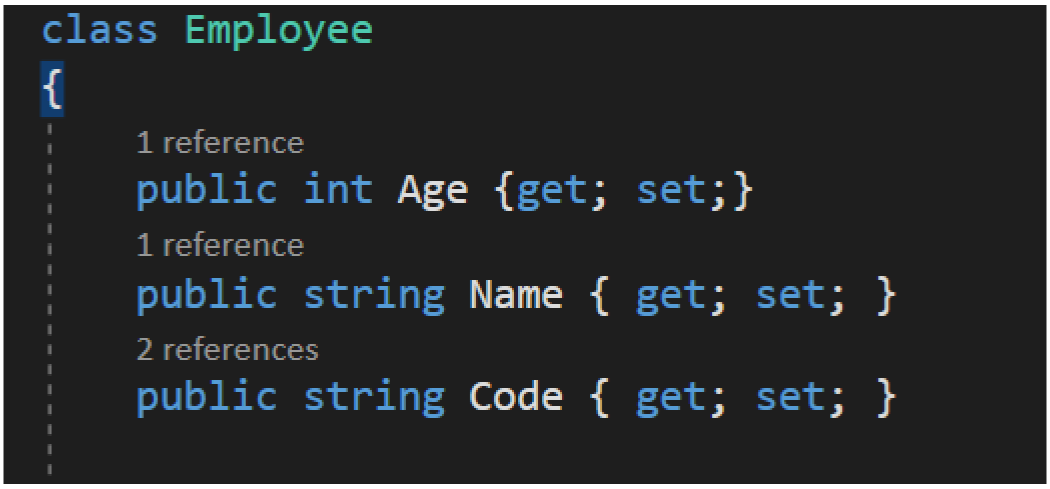
This is the code I wrote for it which sets the labels text to the current long time (clock to the nearest second). Since this is in the subprocedure for the timer tick, it will run the code every time the timer ticks which is every 100ms as that is what the timers tick speed is set to in timer properties.

Upon running the code the label shows up with the current time, updating every second like expected

One “issue” to point out is that the clock is updating every second meaning there is an error interval of 0.9 recurring seconds on either side of the clock. However this software doesn’t need to be that precise so I chose to ignore this “problem”.

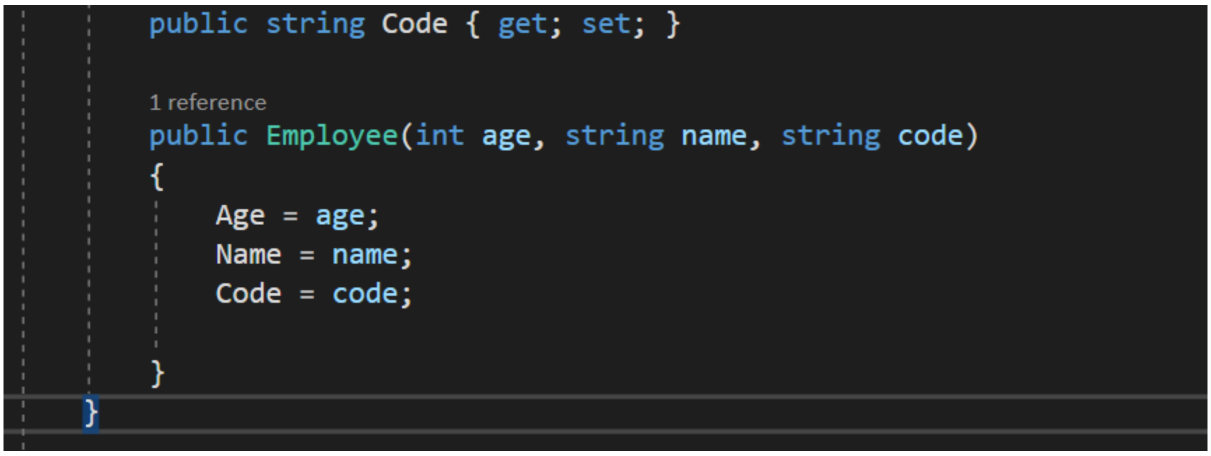
Since this program is not using a database for a source of employees, I needed to hard code them.

So I created a new class and call it Employee. I added a few attributes (with room to add more later on)



```
class Employee
{
    1 reference
    public int Age {get; set;}
    1 reference
    public string Name { get; set; }
    2 references
    public string Code { get; set; }
```

Then, within this class I added another sub procedure which set the variables declared in Employee as the actual attributes which would be attached to the object



```
        public string Code { get; set; }

        1 reference
        public Employee(int age, string name, string code)
        {
            Age = age;
            Name = name;
            Code = code;

        }
    }
```

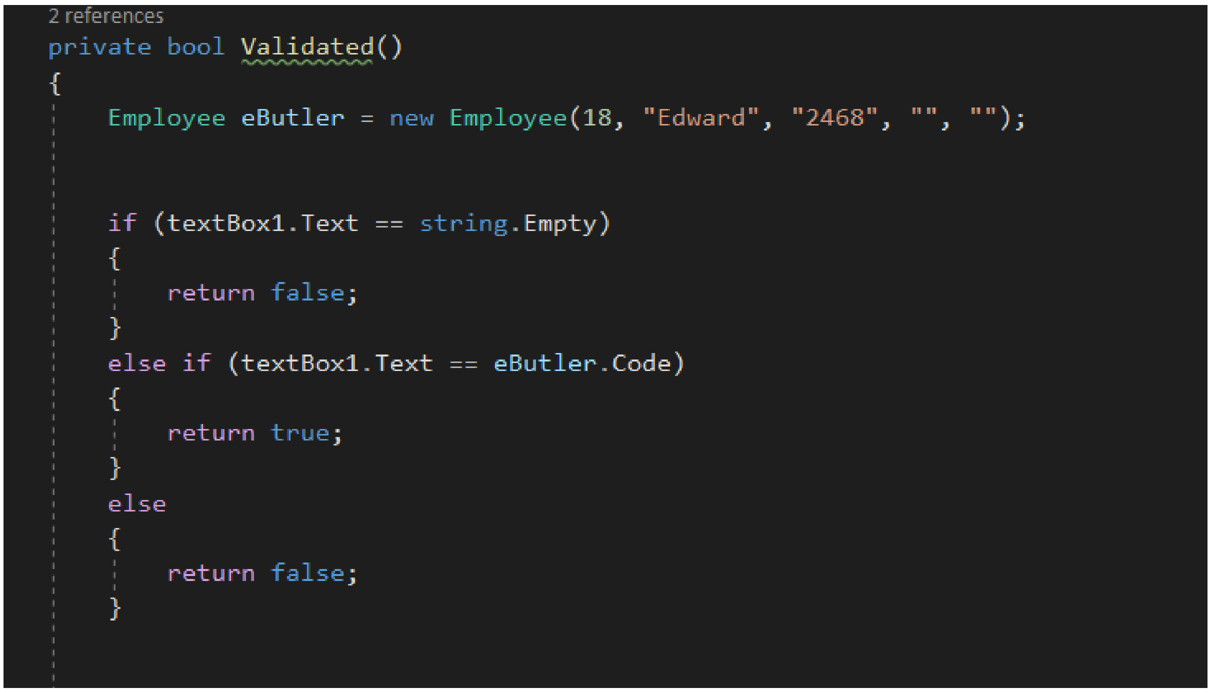
After this I needed to make validation for the first button (the clock in button), obviously I started out by double clicking the button to make the method appear in the script. The button click method, along with any other method created from the form design will break if any other parameters are added so I was having a lot of trouble with getting the validation to work within the button click method, despite me having a fairly good idea of what to do.

After a while I came across a method on youtube on a video for a login form so I applied this method to my own work with a few adjustments since my solution is different from a login form…

So, I created a new method called Validated. This is the validation checker it is a Boolean method so it only returns true or false. The principle of it is…

If it returns true, it means the data entered into the text box on the form matches exactly that of the clock in code.

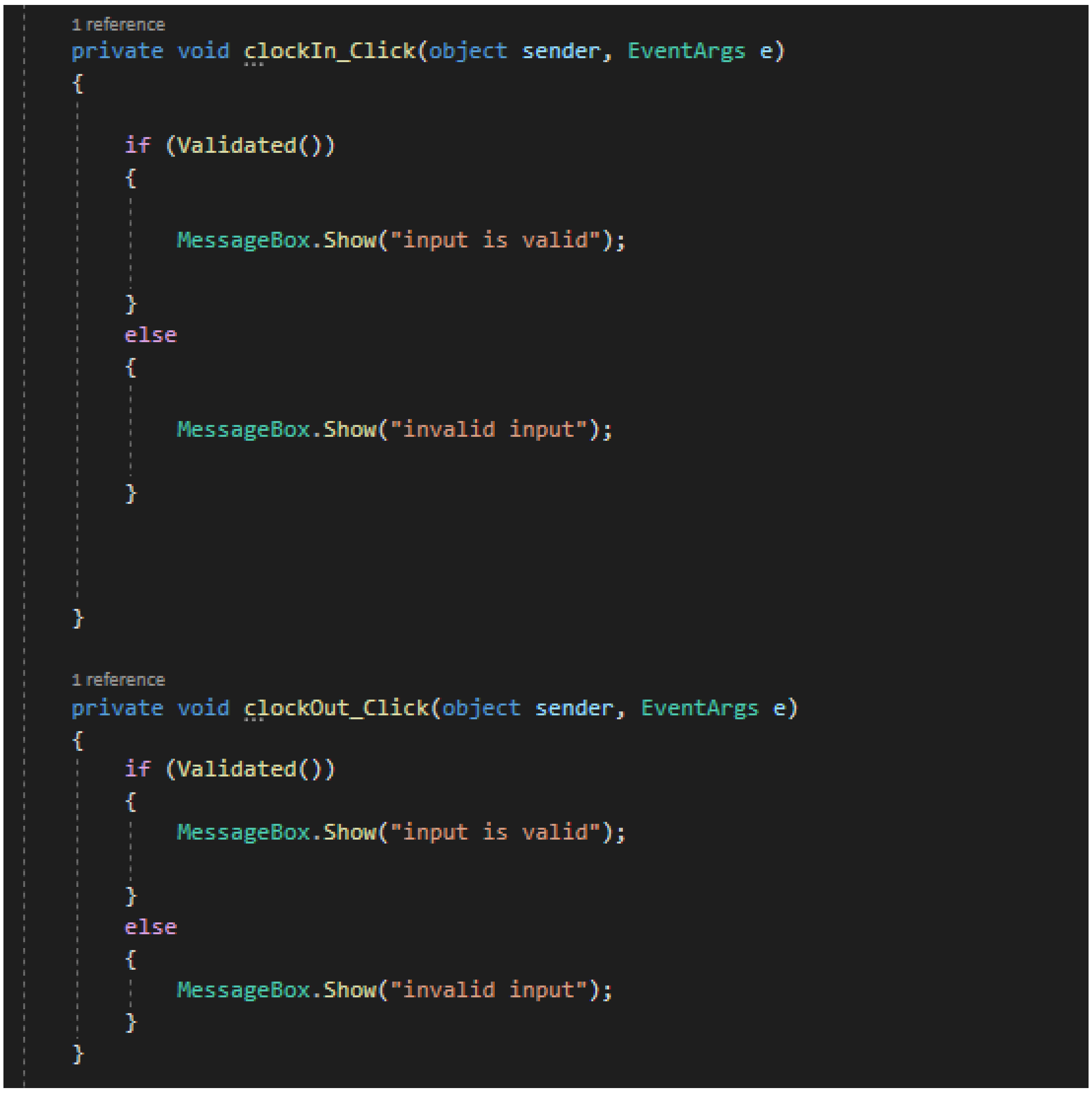
If it returns false, there is either no data entered into the text box or there is invalid data (not a valid clock in code ie not present in the database(in this case the database is hard coded objects for now) entered into the text box.



```
2 references
private bool Validated()
{
    Employee eButler = new Employee(18, "Edward", "2468", "", "");


    if (textBox1.Text == string.Empty)
    {
        return false;
    }
    else if (textBox1.Text == eButler.Code)
    {
        return true;
    }
    else
    {
        return false;
    }

```

here is the method for validation, I also moved the object creation code into this method to make it work.

The method “Validated” would finally allow me to add code to the button click methods

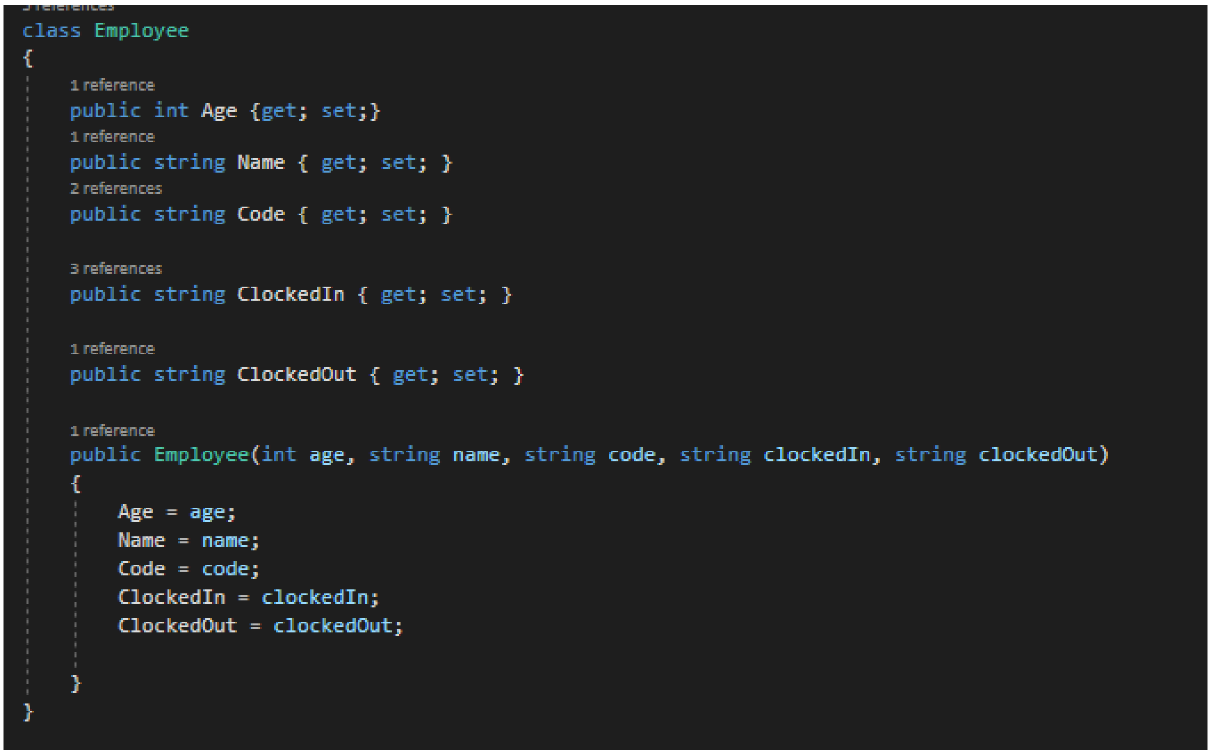


```
1 reference
private void clockIn_Click(object sender, EventArgs e)
{

    if (Validated())
    {

        MessageBox.Show("input is valid");

    }
    else
    {

        MessageBox.Show("invalid input");

    }



}

1 reference
private void clockOut_Click(object sender, EventArgs e)
{
    if (Validated())
    {
        MessageBox.Show("input is valid");

    }
    else
    {
        MessageBox.Show("invalid input");
    }
}
```

This is all the code I added for the time being.

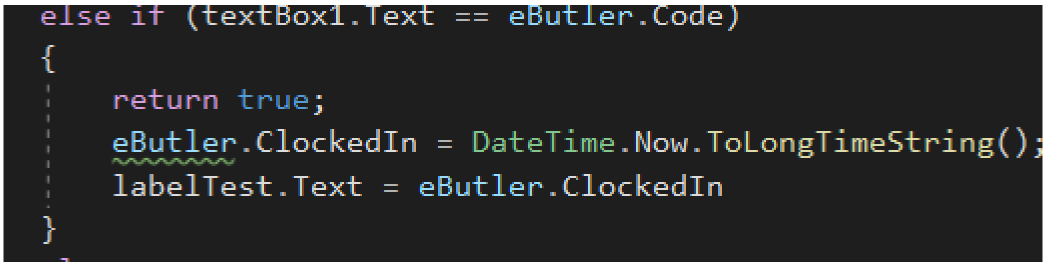
The next thing I needed to program was to assign values for clock in time…

So I created two more attributes for the Employee class, one for clock in time and one for clock out time.



```
class Employee
{
    1 reference
    public int Age {get; set;}
    1 reference
    public string Name { get; set; }
    2 references
    public string Code { get; set; }

    3 references
    public string ClockedIn { get; set; }

    1 reference
    public string ClockedOut { get; set; }

    1 reference
    public Employee(int age, string name, string code, string clockedIn, string clockedOut)
    {
        Age = age;
        Name = name;
        Code = code;
        ClockedIn = clockedIn;
        ClockedOut = clockedOut;

    }
}
```

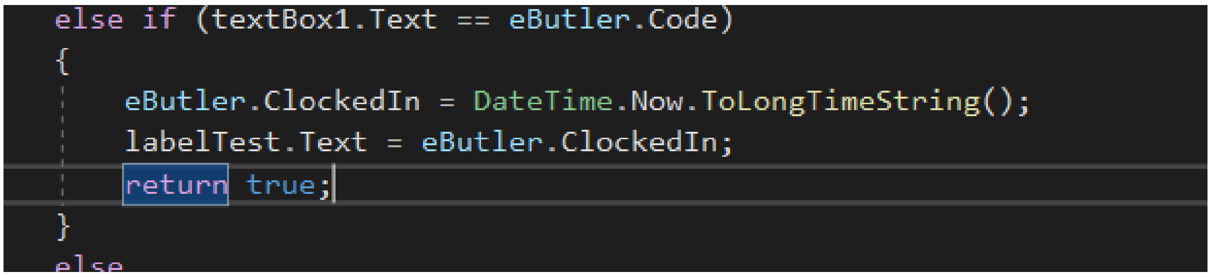
Then I created new code which reassigned the empty string of ClockedIn to the long string of the current time, however…



```
else if (textBox1.Text == eButler.Code)
{
    return true;
    eButler.ClockedIn = DateTime.Now.ToLongTimeString();
    labelTest.Text = eButler.ClockedIn
}
```

I was specifically having troulbe with this part of the code as it hadn't flagged any errors and ran when I pressed f5 however it was not running as intended, the intended way being ClockedIn, would be assigned the time in which the validation on the clock in buttion took place.

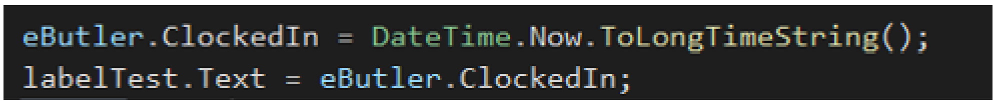
However I realised I had made a stupid mistake as I had put the return before as opposed to after, meaning the sequence didn't get completed during runtime which is why it wasn't working as intended.



```
else if (textBox1.Text == eButler.Code)
{
    eButler.ClockedIn = DateTime.Now.ToLongTimeString();
    labelTest.Text = eButler.ClockedIn;
    return true;
}
else
```

After moving the return to the end it worked as it should and the test label displayed the contents of ClockedIn as I expected.

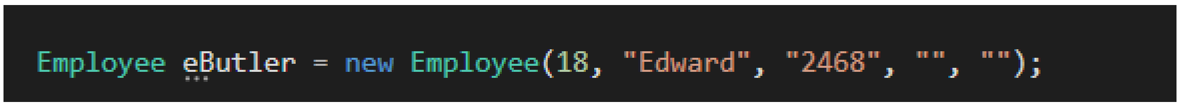
There was quite a big issue with this however, this is that both buttons would function the same way.



```
eButler.ClockedIn = DateTime.Now.ToLongTimeString();
labelTest.Text = eButler.ClockedIn;
```

This is because these two lines were in the validated method, however the validated method is shared by both the clock in and the clock out button.

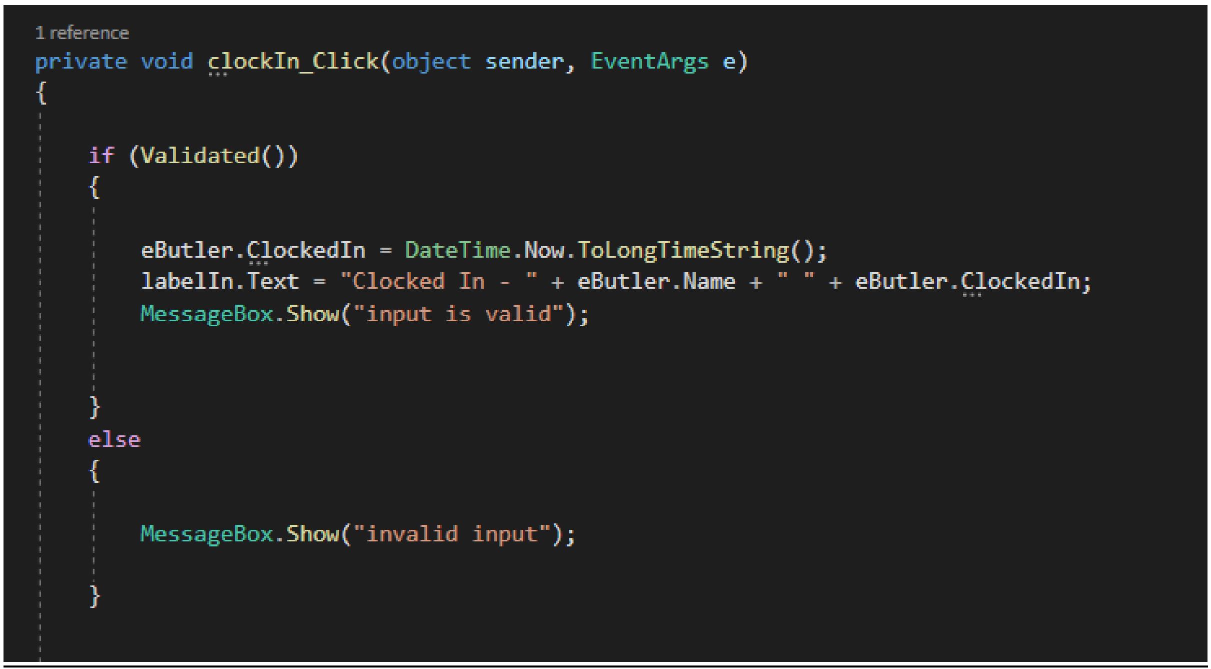
So effectively, I needed to put these two lines into the button click method, this arose an issue initially and that was that the object eButler was not recognized within the methods for the button click, however I was finally able to solve this problem by simply making this line public



```
Employee eButler = new Employee(18, "Edward", "2468", "", "");
```

By moving it outside the validated method.

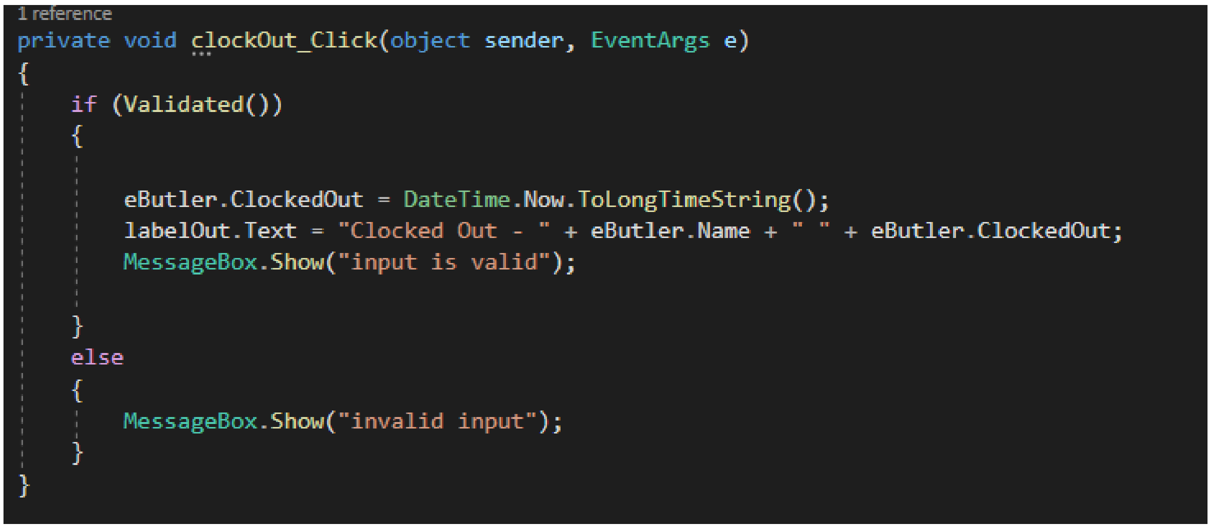
(Making the class public is something which I had tried to do to tackle an earlier roadblock, however I now realise what I was doing wrong there, only the objects need to be public – not the class)



```
1 reference
private void clockIn_Click(object sender, EventArgs e)
{

    if (Validated())
    {

        eButler.ClockedIn = DateTime.Now.ToLongTimeString();
        labelIn.Text = "Clocked In - " + eButler.Name + " " + eButler.ClockedIn;
        MessageBox.Show("input is valid");


    }
    else
    {

        MessageBox.Show("invalid input");

    }

```

I was then able to put this code into the button click method for clocking in.

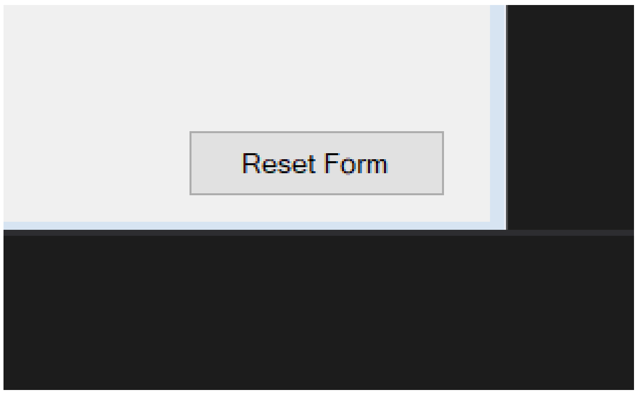
Now that I had done that I was able to put the code into the button click method for clocking out, where I had to make a couple adjustments.



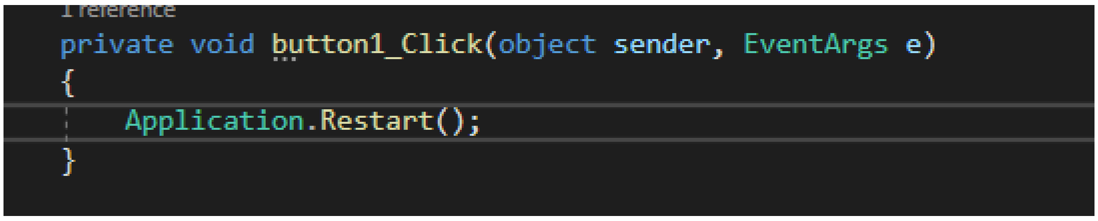
```
1 reference
private void clockOut_Click(object sender, EventArgs e)
{
    if (Validated())
    {

        eButler.ClockedOut = DateTime.Now.ToLongTimeString();
        labelOut.Text = "Clocked Out - " + eButler.Name + " " + eButler.ClockedOut;
        MessageBox.Show("input is valid");

    }
    else
    {
        MessageBox.Show("invalid input");
    }
}
```

(2 new labels: labelIn and labelOut were made for this aswell – to visualise the code working within the form.

I then decided that it would be useful for me to return to my design to just make one slight adjustment and that would be to add a form refresh button to the UI



So I created this button on the form – it is fairly out of the way from the rest of the form so that the user has less of a chance of accidently clicking it.

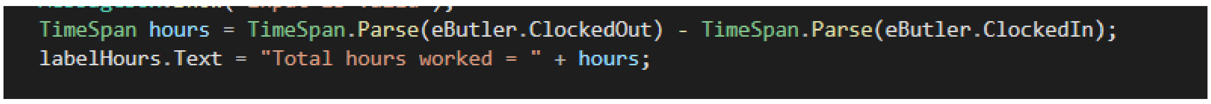


```
1 reference
private void button1_Click(object sender, EventArgs e)
{
    Application.Restart();
}
```

Simply I just added the application.restart event to the button click method.

Next I needed to create a way to subtract the clock in time from the clock out time to hold a new value for the total hours worked.

I wasn't quite sure on how arithmetic worked with times in c# so a few google searches informed me that I need to use TimeSpan.Parse on my string clock in / out attributes.



```
TimeSpan hours = TimeSpan.Parse(eButler.ClockedOut) - TimeSpan.Parse(eButler.ClockedIn);
labelHours.Text = "Total hours worked = " + hours;
```

This is the code I came up with… I first had it in the method for the newly created labelHours, however this didn’t work as I expected so I moved it into the clock out click method in the if statement, this means the label will change upon clock out which is what I wanted.

The next part of the code I need to implement is a method which calculates the amount of money which the user has made within their time clocked in…