INTERMEDIATE PROGRAMMING

CEIS209

PRESENTATION BY
YANETH GONZALEZ

Table Of Content

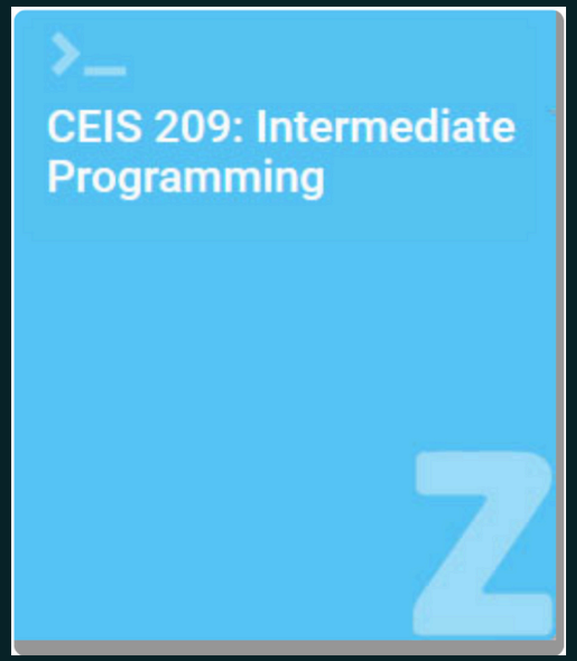
Introduction

In CEIS 209, I explored the world of intermediate programming, focusing on creating functional and user-friendly applications.

- Throughout the course, I learned how to design applications using Windows Forms, validate input, and handle events to make programs interactive. I gained skills in object-oriented programming, including encapsulation, composition, and inheritance, which allowed me to organize and manage data efficiently.
- By building a payroll system project, I applied these concepts to create a complete, real-world application. From saving and loading data using file handling to extending functionality with inheritance, each module added a new layer of complexity and learning.

This course has not only improved my technical skills but also strengthened my confidence in developing software solutions that solve real problems.

Material Used

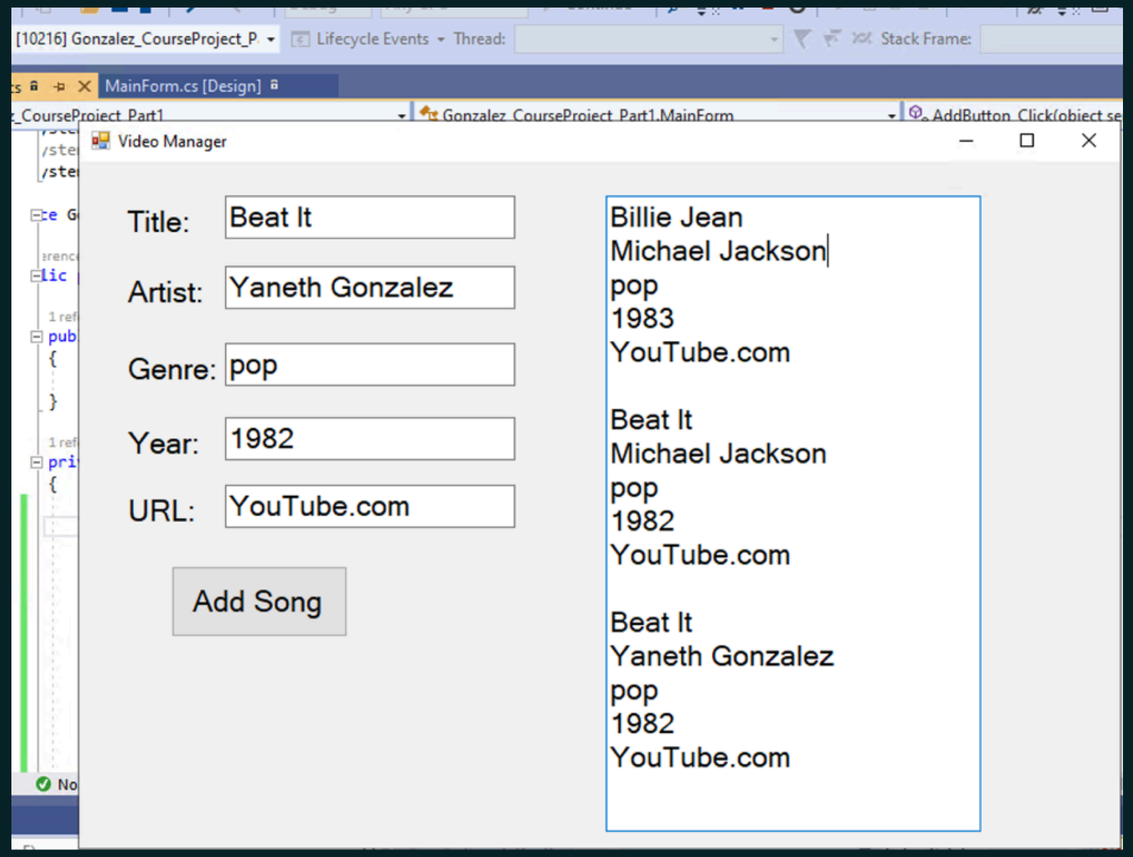


Module 1: Course Project - Create a Windows Form Application With Basic Controls

The Module 1 Project was about building a simple app where users could enter and display song details like the title, artist, genre, year, and a URL.

- It focused on teaching us how to create a user-friendly design, write the right code, and organize everything properly.
- We started by setting up Visual Studio and creating a new project, making sure it was neat and saved in the right place.
- Then, we designed the app by adding labels, text boxes, and buttons, and learned how to customize their properties, like changing the text and size, so everything looked professional.

The coding brought the app to life. We wrote code to grab the song details from the text boxes, format them nicely, and show them in the output section. A big focus was making sure the program worked smoothly.


[10216] Gonzalez_CourseProject_P
Lifecycle Events
Thread:
Stack Frame:
MainForm.cs [Design]
Video Manager
Title:
Beat It
Artist:
Yaneth Gonzalez
Genre:
pop
Year:
1982
URL:
YouTube.com
Add Song
Billie Jean
Michael Jackson
pop
1983
YouTube.com
Beat It
Michael Jackson
pop
1982
YouTube.com
Beat It
Yaneth Gonzalez
pop
1982
YouTube.com

MainForm.cs

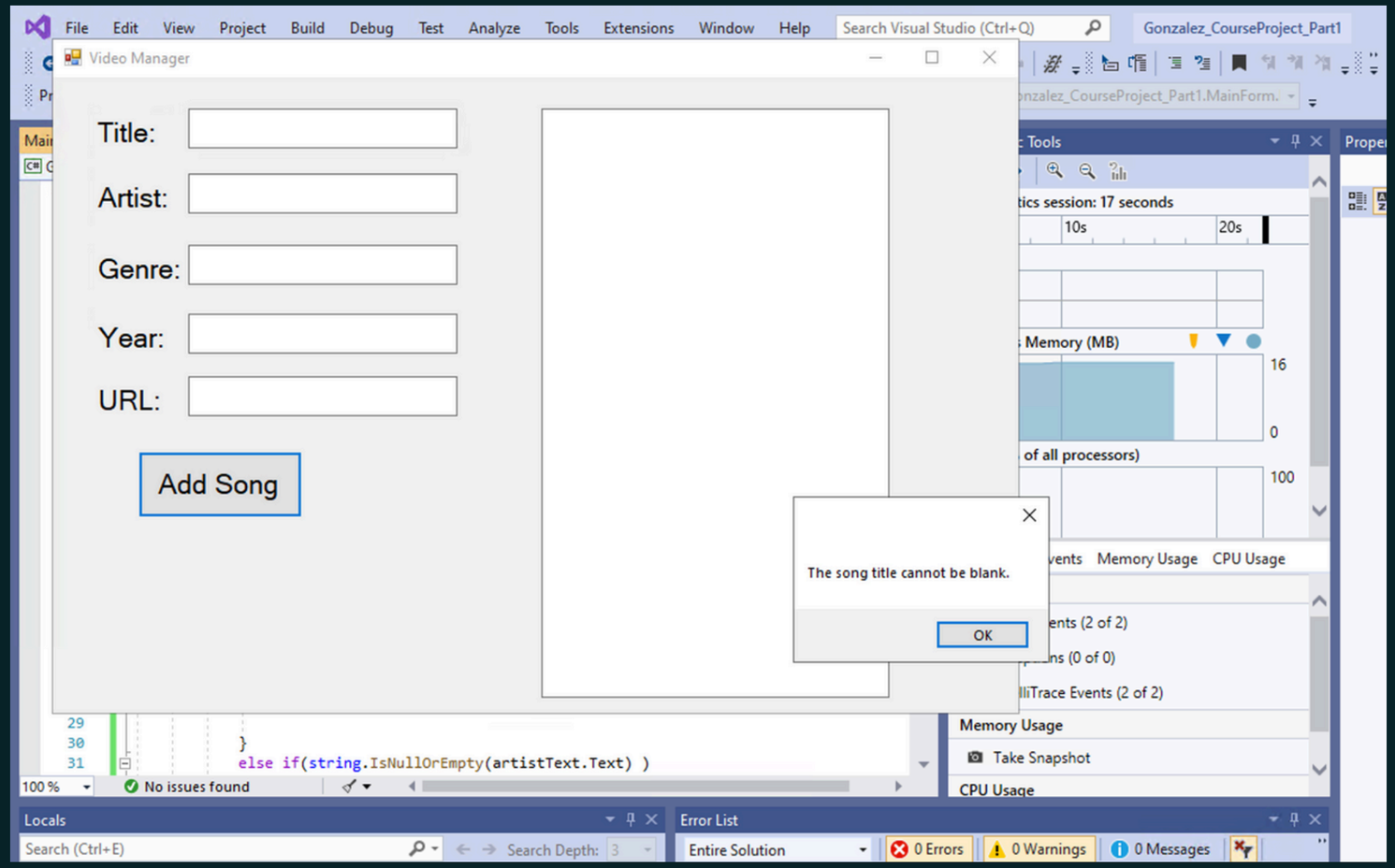
```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part1
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder(outputText.Text);
            string nl = "\r\n";

            sb.Append(titleText.Text);
            sb.Append(nl);
            sb.Append(artistText.Text);
            sb.Append(nl);
            sb.Append(genreText.Text);
            sb.Append(nl);
            sb.Append(yearText.Text);
            sb.Append(nl);
            sb.Append(urlText.Text);
            sb.Append(nl);
            sb.Append(nl);

            outputText.Text = sb.ToString();




        }
    }
}
```

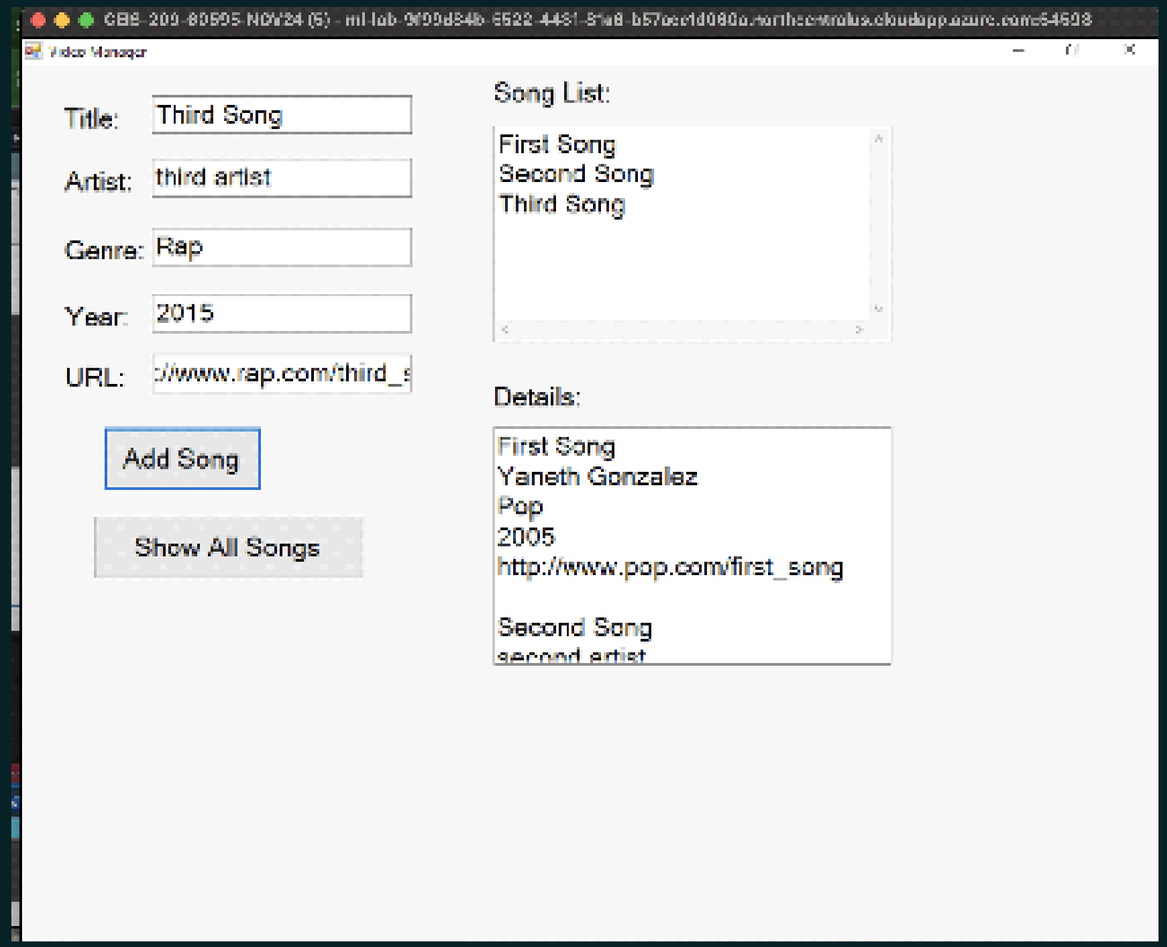
Module 2: Course Project - Validating Input From a Form and Looping Through Songs

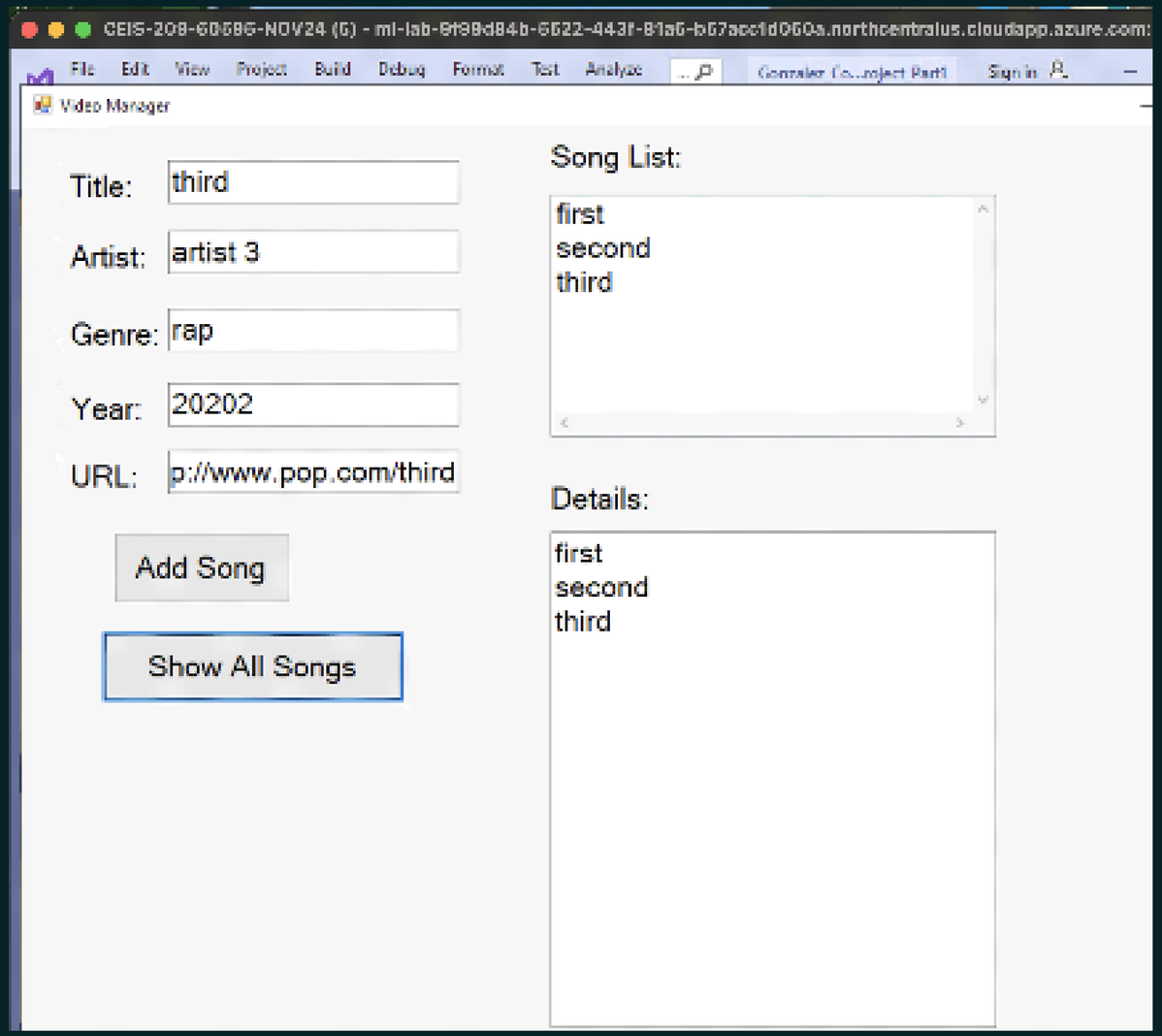
In the Module 2 Project, we improved our app by adding validation to make sure all fields were filled in before adding a song.

- If a field, like Title or Artist, was left empty, the app would show an error message and stop until the problem was fixed.
- We also added a ListBox to show a list of all the songs entered. This made the app more organized and user-friendly.
- The Add Button was updated to add song titles to the ListBox only if all fields were correctly filled.
- Finally, we created a Show All Songs button. This button displayed all the songs from the ListBox in the details section.
- After testing the app and ensuring everything worked smoothly, we took screenshots of the results and submitted the project.

This project taught us how to validate user input, work with a ListBox, and organize data better, making the app more useful and professional.


File Edit View Project Build Debug Test Analyze Tools Extensions Window Help
Search Visual Studio (Ctrl+Q)
Gonzalez_CourseProject_Part1
Video Manager
Title:
Artist:
Genre:
Year:
URL:
Add Song
The song title cannot be blank.
OK
}
else if(string.IsNullOrEmpty(artistText.Text))
No issues found
Memory Usage
Take Snapshot
CPU Usage
Locals
Search (Ctrl+E)
Search Depth: 3
Error List
Entire Solution
0 Errors
0 Warnings
0 Messages


Video Manager
Title:
Third Song
Artist:
third artist
Genre:
Rap
Year:
2015
URL:
://www.rap.com/third_s
Add Song
Show All Songs
Song List:
First Song
Second Song
Third Song
Details:
First Song
Yaneth Gonzalez
Pop
2005
http://www.pop.com/first_song
Second Song


File Edit View Project Build Debug Format Test Analyze
Sign in
Video Manager
Title:
third
Artist:
artist 3
Genre:
rap
Year:
20202
URL:
p://www.pop.com/third
Add Song
Show All Songs
Song List:
first
second
third
Details:
first
second
third

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part1
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder(outputText.Text);
            string nl = "\r\n";

            if (string.IsNullOrEmpty(titleText.Text))
            {
                //Title is blank
                MessageBox.Show("The song title cannot be blank.");

            }
            else if (string.IsNullOrEmpty(artistText.Text))
            {
                // Artist is blank
                MessageBox.Show("The artist cannot be blank");
            }
            else if (string.IsNullOrEmpty(genreText.Text))
            {
                // Genre is blank
                MessageBox.Show("The genre cannot be blank");
            }
            else if (string.IsNullOrEmpty(yearText.Text))
            {
                // Year is blank
                MessageBox.Show("The year cannot be blank");
            }
            else if (string.IsNullOrEmpty(urlText.Text))
            {
                // URL is blank
                MessageBox.Show("The URL cannot be blank");
            }
            else
            {
                // Add title to ListBox
                songList.Items.Add(titleText.Text);
```

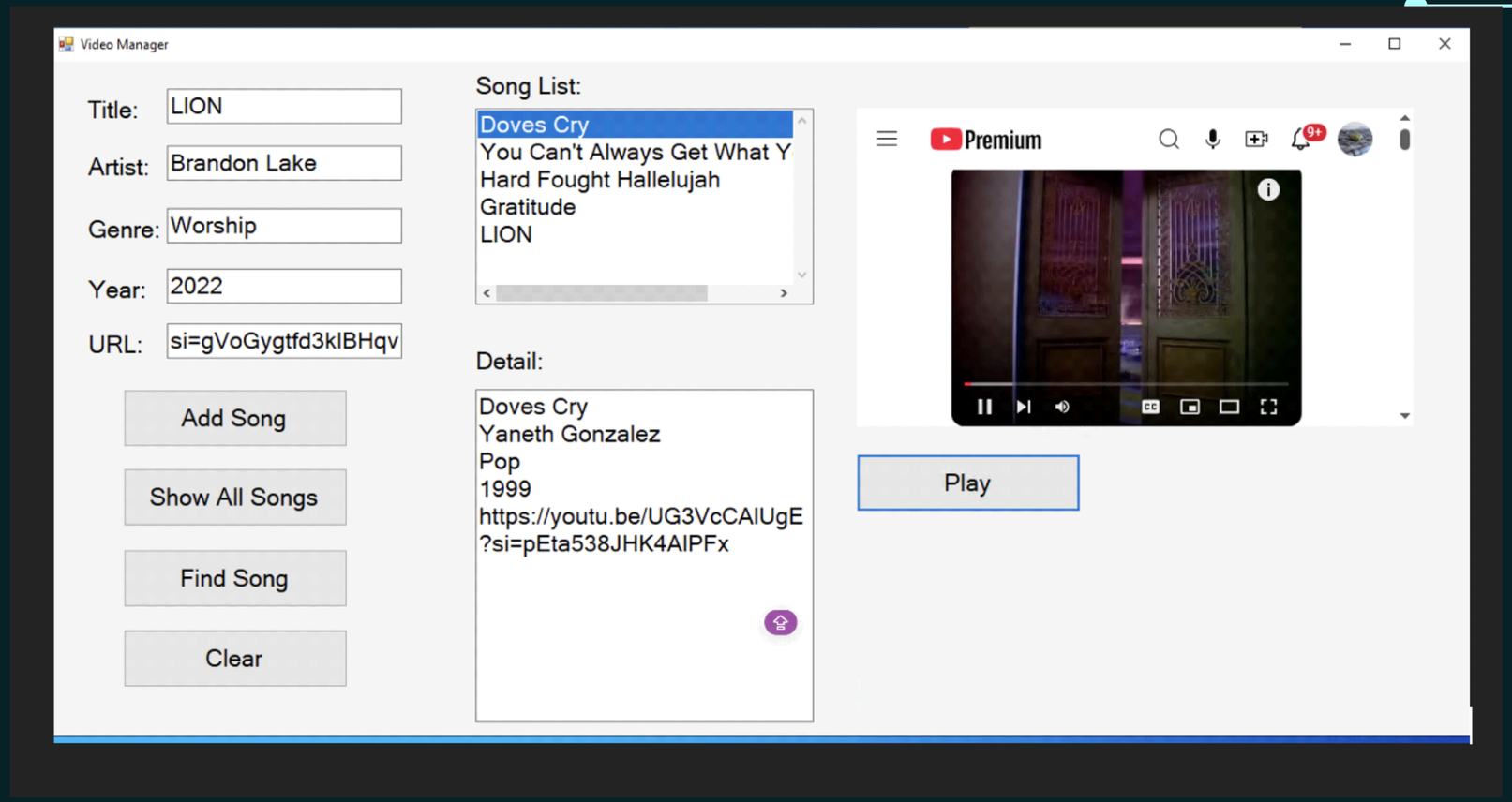
```
                //Build the output text
                sb.Append(titleText.Text);
                sb.Append(nl);
                sb.Append(artistText.Text);
                sb.Append(nl);
                sb.Append(genreText.Text);
                sb.Append(nl);
                sb.Append(yearText.Text);
                sb.Append(nl);
                sb.Append(urlText.Text);
                sb.Append(nl);
                sb.Append(nl);

                outputText.Text = sb.ToString();
            }









        }

        private void AllSongsButton_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder(string.Empty);
            string nl = "\r\n";

            {
                //Build the output text
                foreach (var item in songList.Items)
                    sb.Append(item.ToString());
                sb.Append(nl);
            }
            // Put the output text into the output textbox
            outputText.Text = sb.ToString();


        }
    }
}
```

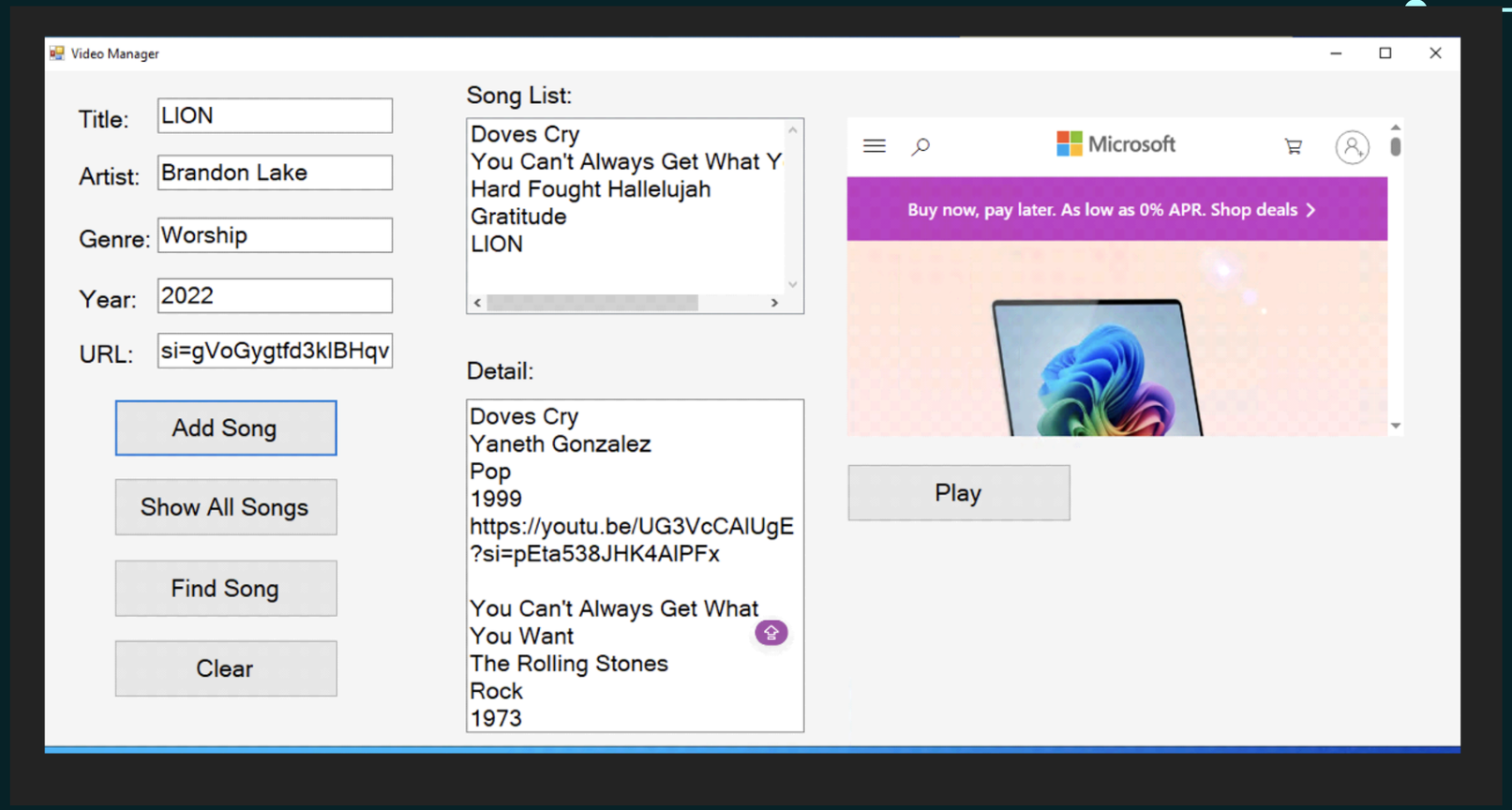
Module 3: Course Project - Writing utility methods and Storing Song Data in Arrays

In Module 3, we focused on improving our project with methods, arrays, and new features.

- First, we created a ValidInput method to check if all fields (Title, Artist, Genre, Year, URL) were filled. If any were empty, an error message showed, and the process stopped.
- Next, we added arrays to store song details like titles, artists, and URLs, and used a songCount variable to track how many songs were added.
- We updated the Add button to validate inputs, store song details in the arrays, and increase the song count. Then, we created methods to find a song's position in the list and check if a song existed, making it easier to search and display details.
- New features included a Find Song button to show song details and a Clear button to empty input fields. We also added a WebView2 control to display and play song URLs, along with a Play button to open the selected URL in the WebView2 window.

Finally, we tested everything by adding songs, searching for them, and playing videos, ensuring all features worked smoothly.


Video Manager
Title:
LION
Artist:
Brandon Lake
Genre:
Worship
Year:
2022
URL:
si=gVoGygtfd3klBHqv
Add Song
Show All Songs
Find Song
Clear
Song List:
Doves Cry
You Can't Always Get What Y
Hard Fought Hallelujah
Gratitude
LION
Detail:
Doves Cry
Yaneth Gonzalez
Pop
1999
https://youtu.be/UG3VcCAIUgE
?si=pEta538JHK4AIPFx
Premium
Play


Video Manager
Title:
LION
Artist:
Brandon Lake
Genre:
Worship
Year:
2022
URL:
si=gVoGygtfd3klBHqv
Add Song
Show All Songs
Find Song
Clear
Song List:
Doves Cry
You Can't Always Get What Y
Hard Fought Hallelujah
Gratitude
LION
Detail:
Doves Cry
Yaneth Gonzalez
Pop
1999
https://youtu.be/UG3VcCAlUgE
?si=pEta538JHK4AIPFx
You Can't Always Get What
You Want
The Rolling Stones
Rock
1973
Microsoft
Buy now, pay later. As low as 0% APR. Shop deals
Play

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using Microsoft.Web.WebView2.Core;

namespace Gonzalez_CourseProject_Part1
{
    public partial class MainForm : Form
    {
        // class level references
        string[] titleArray = new string[5];
        string[] artistArray = new string[5];
        string[] genreArray = new string[5];
        int[] yearArray = new int [5];
        string[] urlArray = new string[5];
        int songCount = 0;

        public MainForm()
        {
            InitializeComponent();
        }
        public bool ValidInput()
        {
            // Return true if all field are non-empty
            bool isValid = true;

            if (string.IsNullOrEmpty(titleText.Text))
            {
                //Title is blank
                MessageBox.Show("The song title cannot be blank.");
                isValid = false;
            }
            else if (string.IsNullOrEmpty(artistText.Text))
            {
                // Artist is blank
                MessageBox.Show("The artist cannot be blank");
                isValid = false;
            }
            else if (string.IsNullOrEmpty(genreText.Text))
            {
                // Genre is blank
                MessageBox.Show("The genre cannot be blank");
                isValid = false;
            }
            else if (string.IsNullOrEmpty(yearText.Text))

            {
                // Year is blank
                MessageBox.Show("The year cannot be blank");
                isValid = false;
            }
```

```
            else if (string.IsNullOrEmpty(urlText.Text))
            {
                // URL is blank
                MessageBox.Show("The URL cannot be blank");
                isValid = false;
            }
            return isValid;

        }
        private void AddButton_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder(outputText.Text);
            string nl = "\r\n";

            if(ValidInput() )

            {
                // Add title to ListBox and song list
                songList.Items.Add(titleText.Text);
                titleArray[songCount] = titleText.Text;
                artistArray[songCount] = artistText.Text;
                genreArray[songCount] = genreText.Text;
                yearArray[songCount] = int.Parse(yearText.Text);
                urlArray[songCount] = urlText.Text;



                    // Increment the song counter
                    songCount++;

                //Build the output text
                sb.Append(titleText.Text);
                sb.Append(nl);
                sb.Append(artistText.Text);
                sb.Append(nl);
                sb.Append(genreText.Text);
                sb.Append(nl);
                sb.Append(yearText.Text);
                sb.Append(nl);
                sb.Append(urlText.Text);
                sb.Append(nl);
                sb.Append(nl);

                outputText.Text = sb.ToString();
            }



        }

        private bool SongInList(string songTitle)
        {
```

```
            bool found = false;

            foreach (var item in songList.Items)
            {
                string currentSong = item as string;
                // lowercase comparison so not case sensitive
                if (songTitle.ToLower() == currentSong. ToLower() )
                {
                    found = true;
                }
            }

            return found;
        }

        private int GetSongIndex(string songTitle)
        {
            int songIndex = songList.Items.IndexOf(songTitle);
            return songIndex;
        }


        private void AllSongsButton_Click(object sender, EventArgs e)
        {
            StringBuilder sb = new StringBuilder(string.Empty);
            string nl = "\r\n";

            {
                //Build the output text
                foreach (var item in songList.Items)
                    sb.Append(item.ToString());
                sb.Append(nl);
            }
            // Put the output text into the output textbox
            outputText.Text = sb.ToString();


        }

        private void FindButton_Click(object sender, EventArgs e)
        {
            if(SongInList(titleText.Text) )
            {
                StringBuilder sb = new StringBuilder(string.Empty);
                string nl = "\r\n";

                int songIndex = GetSongIndex(titleText.Text);

                // Build the output text
                sb.Append(titleArray[songIndex]);
                sb.Append(nl);
                sb.Append(artistArray[songIndex]);
                sb.Append(nl);
                sb.Append(genreArray[songIndex]);
                sb.Append(nl);
                sb.Append(yearArray[songIndex]);
```

```
                sb.Append(nl);
                sb.Append(urlArray[songIndex]);
                sb.Append(nl);


                outputText.Text = sb.ToString();




            }
        }

        private void ClearButton_Click(object sender, EventArgs e)
        {
            titleText.Text = ""; // one to clear textbox
            artistText.Text = String.Empty;  // second way to clear
            genreText.Clear();  // third way to clear
            yearText.Clear();
            urlText.Clear();
        }

        private void PlayButton_Click(object sender, EventArgs e)
        {
            int songIndex = songList.SelectedIndex;
            string url = urlArray[songIndex];
            webViewDisplay.CoreWebView2.Navigate(url);


        }

        private void SongList_SelectedIndexChanged(object sender, EventArgs e)
        {
            int songIndex = songList.SelectedIndex;

            // If song is selected (index greater than -1), show the details
            if(songIndex > -1)
            {
                StringBuilder sb = new StringBuilder(string.Empty);
                string nl = "\r\n";

                // Build the output text
                sb.Append(titleArray[songIndex]);
                sb.Append(nl);
                sb.Append(artistArray[songIndex]);
                sb.Append(nl);
                sb.Append(genreArray[songIndex]);
                sb.Append(nl);
                sb.Append(yearArray[songIndex]);
                sb.Append(nl);
                sb.Append(urlArray[songIndex]);
                sb.Append(nl);


                outputText.Text = sb.ToString();
```

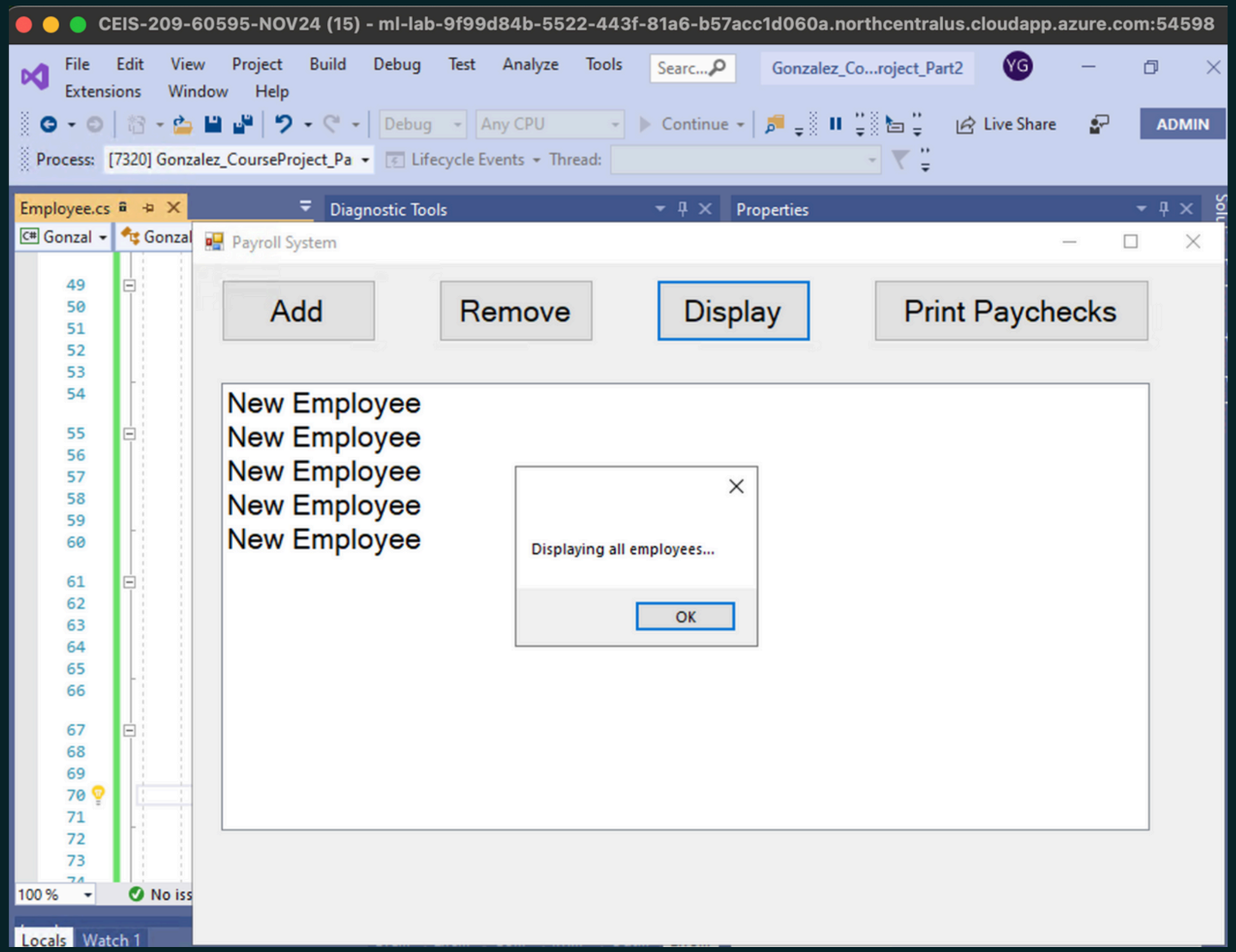
Module 4: Course Project - Create a Windows Forms Application With Basic Controls and Add an Employee Class

In the Module 4 Project, we made a simple payroll system with two forms that could work together. The goal was to manage employee data and add basic features like adding, removing, and displaying employee information.

First, we created the main form, renamed it "Payroll System," and added buttons for actions like Add, Remove, Display, and Print Paychecks. We also added a ListBox to show the list of employees.
We then wrote code for the buttons:

- The Add Button lets us add a new employee to the ListBox.
- The Remove Button deletes the employee that is selected in the ListBox.
- The Display Button shows a message with employee details (just a placeholder for now).
- The Print Paychecks Button shows a message saying it’s not finished yet.

Next, we created an Employee class to organize employee details like name, social security number, and hire date. This made it easier to handle employee information. The class included a method to summarize employee details (ToString) and a placeholder for calculating pay.


CEIS-209-60595-NOV24 (15) - ml-lab-9f99d84b-5522-443f-81a6-b57acc1d060a.northcentralus.cloudapp.azure.com:54598
File Edit View Project Build Debug Test Analyze Tools
Extensions Window Help
Gonzalez_Co...roject_Part2
Debug
Any CPU
Continue
Live Share
ADMIN
Process: [7320] Gonzalez_CourseProject_Pa
Lifecycle Events
Thread:
Employee.cs
Diagnostic Tools
Properties
Payroll System
Add
Remove
Display
Print Paychecks
New Employee
New Employee
New Employee
New Employee
New Employee
Displaying all employees...
OK
100 %
Locals
Watch 1

Employee.cs

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    public class Employee
    {
        // attributes
        private string firstName;
        private string lastName;
        private string ssn;
        private DateTime hireDate;

        // constructors
        public Employee()
        {
            firstName = "N/A";
            lastName = "N/A";
            ssn = "N/A";
            hireDate = new DateTime();
        }
        public Employee(string firstName, string lastName, string ssn, DateTime hireDate)
        {
            this.firstName = firstName;
            this.lastName = lastName;
            this.ssn = ssn;
            this.hireDate = hireDate;
        }

        // behaviors
        public override string ToString()
        {
            return "firstName=" + firstName
                + ", lastName=" + lastName
                + ", ssn=" + ssn
                + ", hireDate=" + hireDate.ToShortDateString();

        }

        public double CalculatePay()
        {
            return 0;
        }

        // properties
        public string FirstName
        {
            get { return firstName; }
            set { firstName = value; } // emp.FirstName ="Bob";
        }

        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }

        public string SSN
        {
            get { return ssn; }
            set { ssn = value; }
        }

        public DateTime HireDate
        {
            get { return hireDate; }
            set { hireDate = value; }
        }


    }
}
```

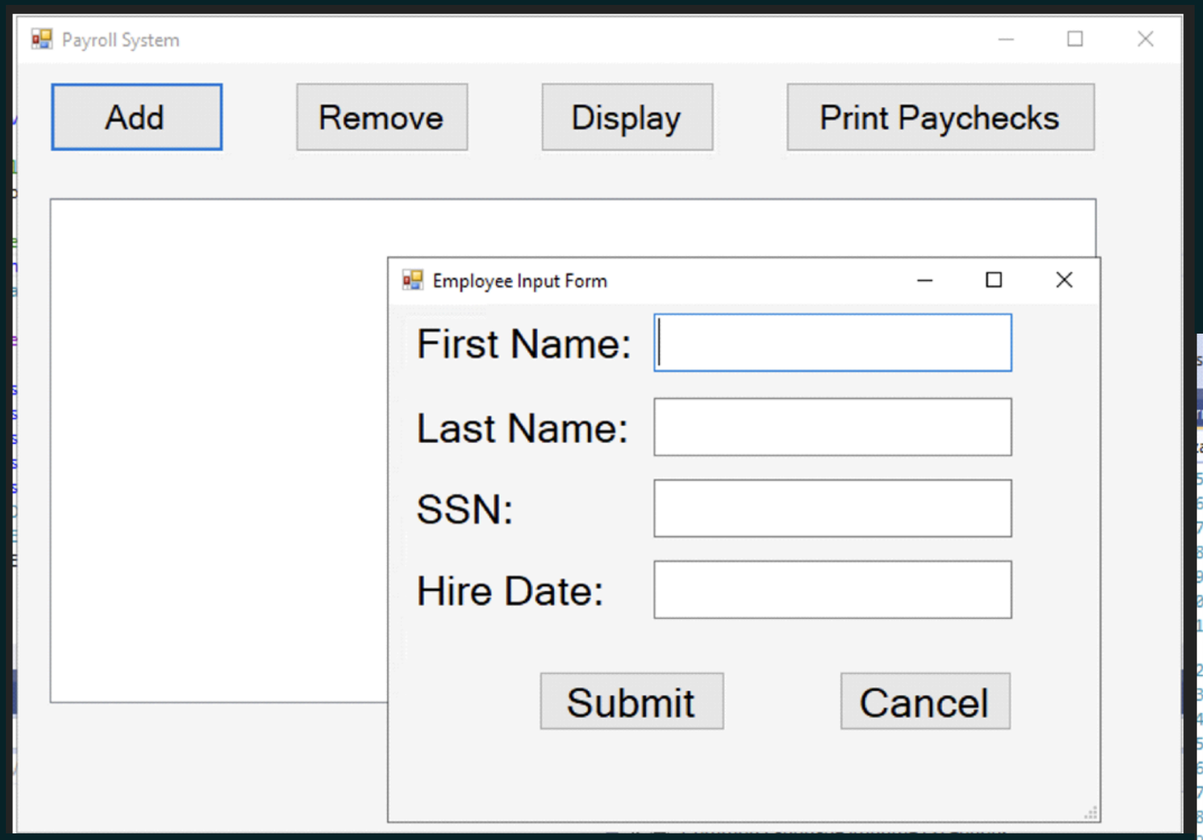
Module 5: Course Project - Encapsulating Data and Writing to a File

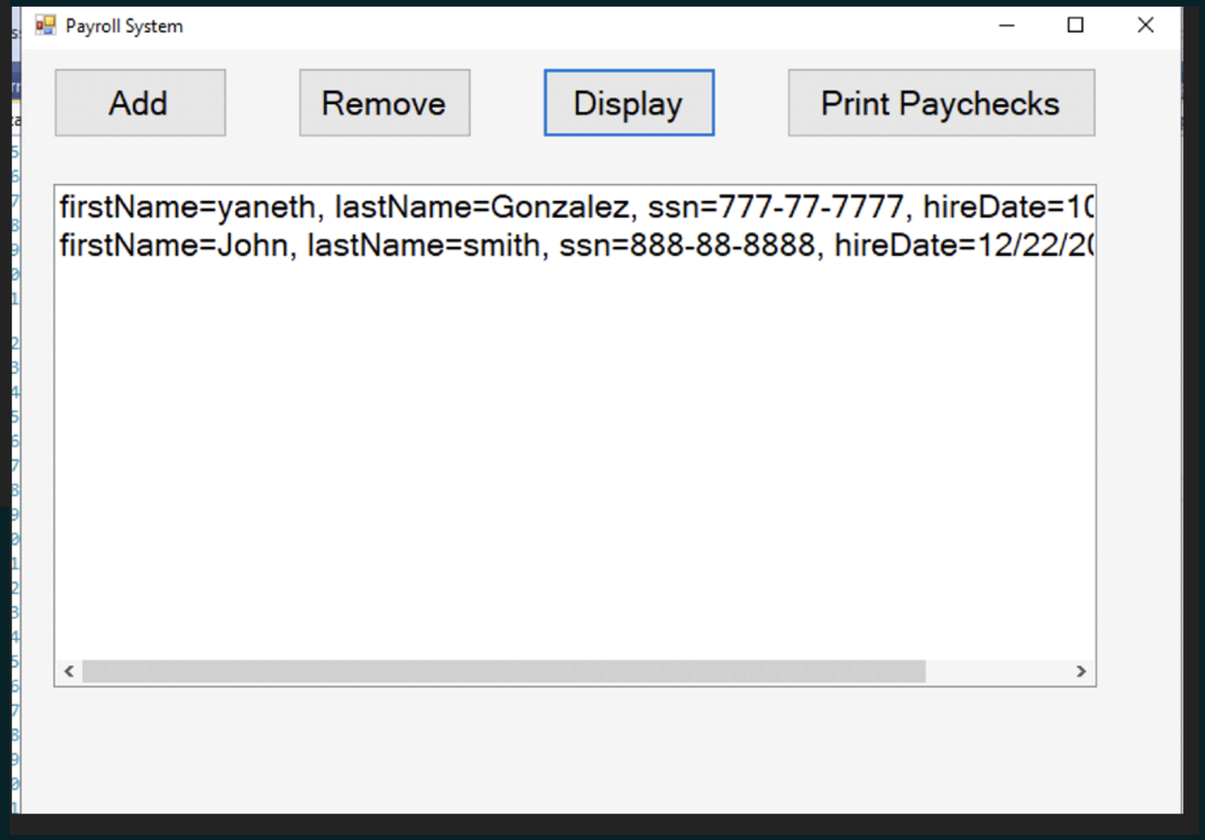
In Module 5, I learned how to integrate the Employee class into my payroll system application, encapsulating employee data to make the program more organized and maintainable.

- I created a second form, InputForm, to collect employee information and added functionality to create Employee objects.

- I also learned how to write and read employee data to and from a file using Comma-Separated Values (CSV) format, ensuring that employee data persists even after the application is closed.

This module helped me better understand how to manage data securely and efficiently, using object-oriented principles and file handling techniques.


Payroll System
Add
Remove
Display
Print Paychecks
Employee Input Form
First Name:
Last Name:
SSN:
Hire Date:
Submit
Cancel


Payroll System
Add
Remove
Display
Print Paychecks
firstName=yaneth, lastName=Gonzalez, ssn=777-77-7777, hireDate=1(
firstName=John, lastName=smith, ssn=888-88-8888, hireDate=12/22/2(

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part2
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            // add item to the Employee listbox
            InputForm frmInput = new InputForm();

            using (frmInput)
            {
                DialogResult result = frmInput.ShowDialog();

                // see if input form was cancelled
                if (result == DialogResult.Cancel)
                    return;   // end the method since cancelled

                // get user's input and create Employee object
                string fName = frmInput.FirstNameTextBox.Text;
                string lName = frmInput.LastNameTextBox.Text;
                string ssn = frmInput.SSNTextBox.Text;
                string date = frmInput.HireDateTextBox.Text;
                DateTime hireDate = DateTime.Parse(date);

                Employee emp = new Employee(fName, lName, ssn, hireDate);

                // add the Employee object to the Employee listbox
                EmployeesListBox.Items.Add(emp);

                // write all date to the file
                WriteEmpsToFile();


            }
```

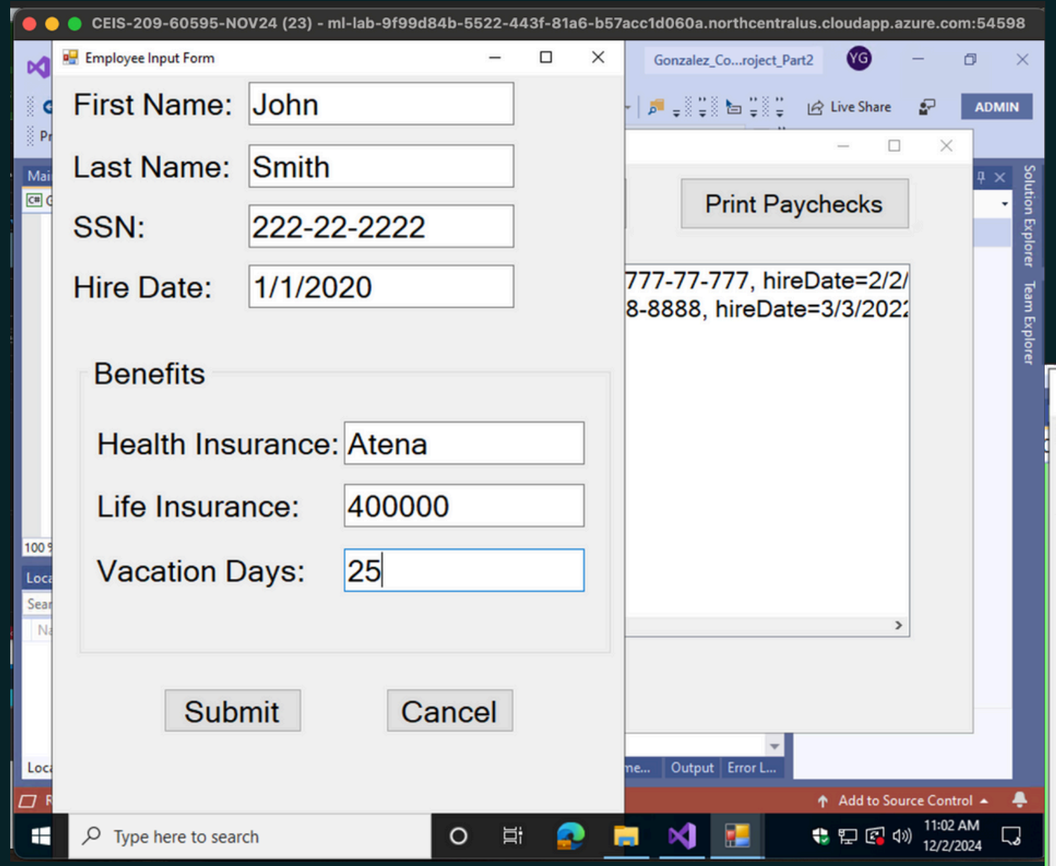
```
private void WriteEmpsToFile()
{
    string filename = "Employees.csv";

    StreamWriter sw = new StreamWrite(filename);

    foreach (Employee emp in EmployeesListBox.Items)
    {
        sw.WriteLine(emp.FirstName + ','
            + emp.LastName + ','
            + emp.SSN + ','
            + emp.HireDate.ToShortDateString());
    }

    sw.Close();

}

private void RemoveButton_Click(object sender, EventArgs e)
{
    //remove the selected items from the Employee listbox
    int itemNumber = EmployeesListBox.SelectedIndex;

    if(itemNumber > -1)
    {
        EmployeesListBox.Items.RemoveAt(itemNumber);
        WriteEmpsToFile();

    }
    else
    {
        MessageBox.Show("Please select employee to remove.");

    }

}

private void DisplayButton_Click(object sender, EventArgs e)
{
    // clear the Employee listbox
    EmployeesListBox.Items.Clear();

    // read employees from the file
    string filename = "Employees.csv";
    StreamReader sr = new StreamReader(filename);

    while(sr.Peek() > -1 )
    {
        string line = sr.ReadLine();
        string[] parts = line.Split(',');
        string firstName = parts[0];
        string lastName = parts[1];
        string ssn = parts[2];
        DateTime hireDate = DateTime.Parse(parts[3]);
        Employee emp = new Employee(firstName, lastName, ssn, hireDate);
```

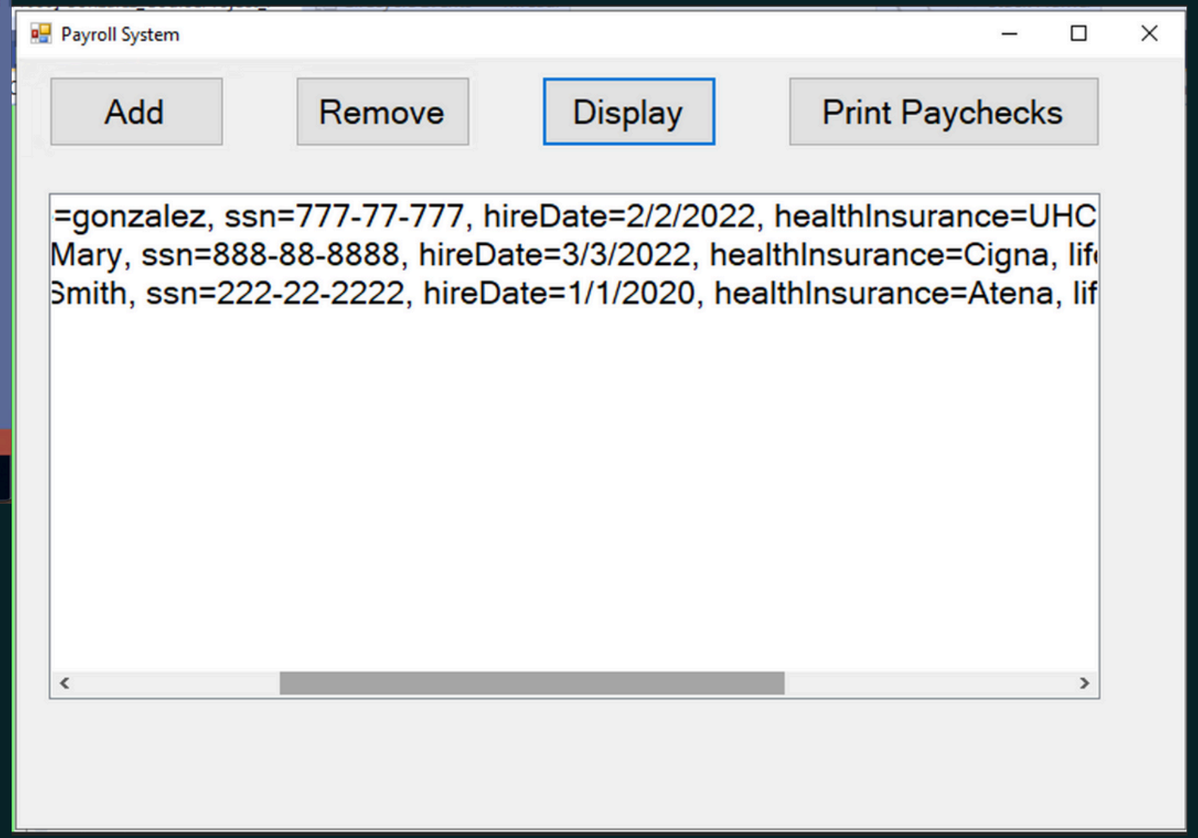
```
            EmployeesListBox.Items.Add(emp);

        }

        sr.Close();

    }

    private void PrintPaychecksButton_Click(object sender, EventArgs e)
    {
        MessageBox.Show("Printing paychecks for all employees...");
    }
}
}
```

Module 6: Course Project - Composition

In Module 6, we added benefits tracking to the employee program. We created a Benefits class to handle health insurance, life insurance, and vacation days.

- Then, we connected this class to the Employee class so each employee includes their benefits.
- The input form was updated to collect benefits details.
- We added these fields and adjusted the "Add" button to save the benefits information along with the employee details. The program now saves and reads this data correctly from a file.
- To update employee details, we added a double-click feature. When you double-click an employee’s name, their details, including benefits, are loaded into the form for editing. After submitting changes, the old record is replaced with the updated one.


CEIS-209-60595-NOV24 (23) - ml-lab-9f99d84b-5522-443f-81a6-b57acc1d060a.northcentralus.cloudapp.azure.com:54598
Employee Input Form
First Name: John
Last Name: Smith
SSN: 222-22-2222
Hire Date: 1/1/2020
Benefits
Health Insurance: Atena
Life Insurance: 400000
Vacation Days: 25
Submit
Cancel
Print Paychecks
777-77-777, hireDate=2/2/
8-8888, hireDate=3/3/202
Live Share
ADMIN
Solution Explorer
Team Explorer
Output
Error L...
Add to Source Control
Type here to search
11:02 AM
12/2/2024


Payroll System
Add
Remove
Display
Print Paychecks
=gonzalez, ssn=777-77-777, hireDate=2/2/2022, healthInsurance=UHC
Mary, ssn=888-88-8888, hireDate=3/3/2022, healthInsurance=Cigna, lif
Smith, ssn=222-22-2222, hireDate=1/1/2020, healthInsurance=Atena, lif

MainForm.cs

MAIN FORM:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part2




{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            // add item to the Employee listbox
            InputForm frmInput = new InputForm();

            using (frmInput)
            {
                DialogResult result = frmInput.ShowDialog();

                // see if input form was cancelled
                if (result == DialogResult.Cancel)
                    return;  // end the method since cancelled

                // get user's input and create Employee object
                string fName = frmInput.FirstNameTextBox.Text;
                string lName = frmInput.LastNameTextBox.Text;
                string ssn = frmInput.SSNTextBox.Text;
                string date = frmInput.HireDateTextBox.Text;
                DateTime hireDate = DateTime.Parse(date);
                string healthIns = frmInput.HealthInsuranceTextBox.Text;
                int lifeIns = int.Parse(frmInput.LifeInsuranceTextBox.Text);
                int vacation = int.Parse(frmInput.VactionDaysTextBox.Text);

                Benefits ben = new Benefits(healthIns, lifeIns, vacation);

                Employee emp = new Employee(fName, lName, ssn, hireDate, ben);

                // add the Employee object to the Employee listbox
                EmployeesListBox.Items.Add(emp);

                // write all date to the file
                WriteEmpsToFile();


            }


        }
        private void WriteEmpsToFile()
        {
            string filename = "Employees.csv";

            StreamWriter sw = new StreamWrite(filename);
```

```
            foreach (Employee emp in EmployeesListBox.Items)
            {
                sw.WriteLine(emp.FirstName + ','
                    + emp.LastName + ','
                    + emp.SSN + ','
                    + emp.HireDate.ToShortDateString() + ','
                    + emp.BenefitsEmp.HealthInsurance + ','
                    + emp.BenefitsEmp.LifeInsurance + ','
                    + emp.BenefitsEmp.Vacation);
            }

            sw.Close();

        }

        private void RemoveButton_Click(object sender, EventArgs e)
        {
            //remove the selected items from the Employee listbox
            int itemNumber = EmployeesListBox.SelectedIndex;

            if(itemNumber > -1)
            {
                EmployeesListBox.Items.RemoveAt(itemNumber);
                WriteEmpsToFile();

            }
            else
            {
                MessageBox.Show("Please select employee to remove.");

            }

        }

        private void DisplayButton_Click(object sender, EventArgs e)
        {
            // clear the Employee listbox
            EmployeesListBox.Items.Clear();

            // read employees from the file
            string filename = "Employees.csv";
            StreamReader sr = new StreamReader(filename);

            while(sr.Peek() > -1 )
            {
                string line = sr.ReadLine();
                string[] parts = line.Split(',');
                string firstName = parts[0];
                string lastName = parts[1];
                string ssn = parts[2];
                DateTime hireDate = DateTime.Parse(parts[3]);
```

```
                string healthIns = parts[4];
                int lifeIns = int.Parse(parts[5]);
                int vacation = int.Parse(parts[6]);

                Benefits ben = new Benefits(healthIns, lifeIns, vacation);
                Employee emp = new Employee(firstName, lastName, ssn, hireDate, ben );

                EmployeesListBox.Items.Add(emp);


            }

            sr.Close();

        }

        private void PrintPaychecksButton_Click(object sender, EventArgs e)
        {
            MessageBox.Show("Printing paychecks for all employees...");
        }

        private void EmployeesListBox_DoubleClick(object sender, EventArgs e)
        {
            // get the selected Employee object
            Employee emp = EmployeesListBox.SelectedItem as Employee;

            // show the Input/Update form with the Employee info
            InputForm frmUpdate = new InputForm();
            frmUpdate.FirstNameTextBox.Text = emp.FirstName;
            frmUpdate.LastNameTextBox.Text = emp.LastName;
            frmUpdate.SSNTextBox.Text = emp.SSN;
            frmUpdate.HireDateTextBox.Text = emp.HireDate.ToShortDateString();
            frmUpdate.HealthInsuranceTextBox.Text = emp.BenefitsEmp.HealthInsurance;
            frmUpdate.LifeInsuranceTextBox.Text = emp.BenefitsEmp.LifeInsurance.ToString();
            frmUpdate.VactionDaysTextBox.Text = emp.BenefitsEmp.Vacation.ToString();
            DialogResult result = frmUpdate.ShowDialog();

            // if cancelled, stop the method
            if (result == DialogResult.Cancel)
                return;  // end the method

            // delete the selected object
            int position = EmployeesListBox.SelectedIndex;
            EmployeesListBox.Items.RemoveAt(position);

            // create new employee using the updating information
            Employee newEmp = new Employee();
            newEmp.FirstName = frmUpdate.FirstNameTextBox.Text;
            newEmp.LastName = frmUpdate.LastNameTextBox.Text;
            newEmp.SSN = frmUpdate.SSNTextBox.Text;
            newEmp.HireDate = DateTime.Parse(frmUpdate.HireDateTextBox.Text);
            newEmp.BenefitsEmp.HealthInsurance = frmUpdate.HealthInsuranceTextBox.Text;
```

```
            newEmp.BenefitsEmp.LifeInsurance =int.Parse(frmUpdate.LifeInsuranceTextBox.Text);
            newEmp.BenefitsEmp.Vacation = int.Parse(frmUpdate.VactionDaysTextBox.Text);

            // add the new employee to the listbox
            EmployeesListBox.Items.Add(newEmp);
```

InputForm.cs

INPUT FORM:

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part2
{
    public partial class InputForm : Form
    {
        public InputForm()
        {
            InitializeComponent();
        }

        private void SubmitButton_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.OK;
            this.Hide();

        }

        private void ExitButton_Click(object sender, EventArgs e)
        {
            this.DialogResult = DialogResult.Cancel;
            this.Hide();
        }
    }
}
```

Employee.cs

Employee Class:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    public class Employee
    {
        // attributes
        private string firstName;
        private string lastName;
        private string ssn;
        private DateTime hireDate;
        private Benefits benefits;

        // constructors
        public Employee()
        {
            firstName = "N/A";
            lastName = "N/A";
            ssn = "N/A";
            hireDate = new DateTime();
            benefits = new Benefits();
        }
        public Employee(string firstName, string lastName, string ssn, DateTime hireDate, Benefits
benefits)
        {
            this.firstName = firstName;



            this.lastName = lastName;
            this.ssn = ssn;
            this.hireDate = hireDate;
            this.benefits = benefits;
        }

        // behaviors
        public override string ToString()
        {
            return "firstName=" + firstName
                + ", lastName=" + lastName
                + ", ssn=" + ssn
                + ", hireDate=" + hireDate.ToShortDateString()
                + ", healthInsurance=" + benefits.HealthInsurance
                + ", lifeInsurance=" + benefits.LifeInsurance
                + ", vacation=" + benefits.Vacation;

        }

        public double CalculatePay()
        {
            return 0;
        }

        // properties
        public string FirstName
        {
            get { return firstName; }
            set { firstName = value; } // emp.FirstName ="Bob";
        }

        public string LastName
        {
            get { return lastName; }
            set { lastName = value; }
        }

        public string SSN
        {
            get { return ssn; }
            set { ssn = value; }
        }

        public DateTime HireDate
        {
            get { return hireDate; }
            set { hireDate = value; }
        }

        public Benefits BenefitsEmp
        {


            get { return benefits; }
            set { benefits = value; }
        }


    }
}
```

Benefits.cs

Benefits class:

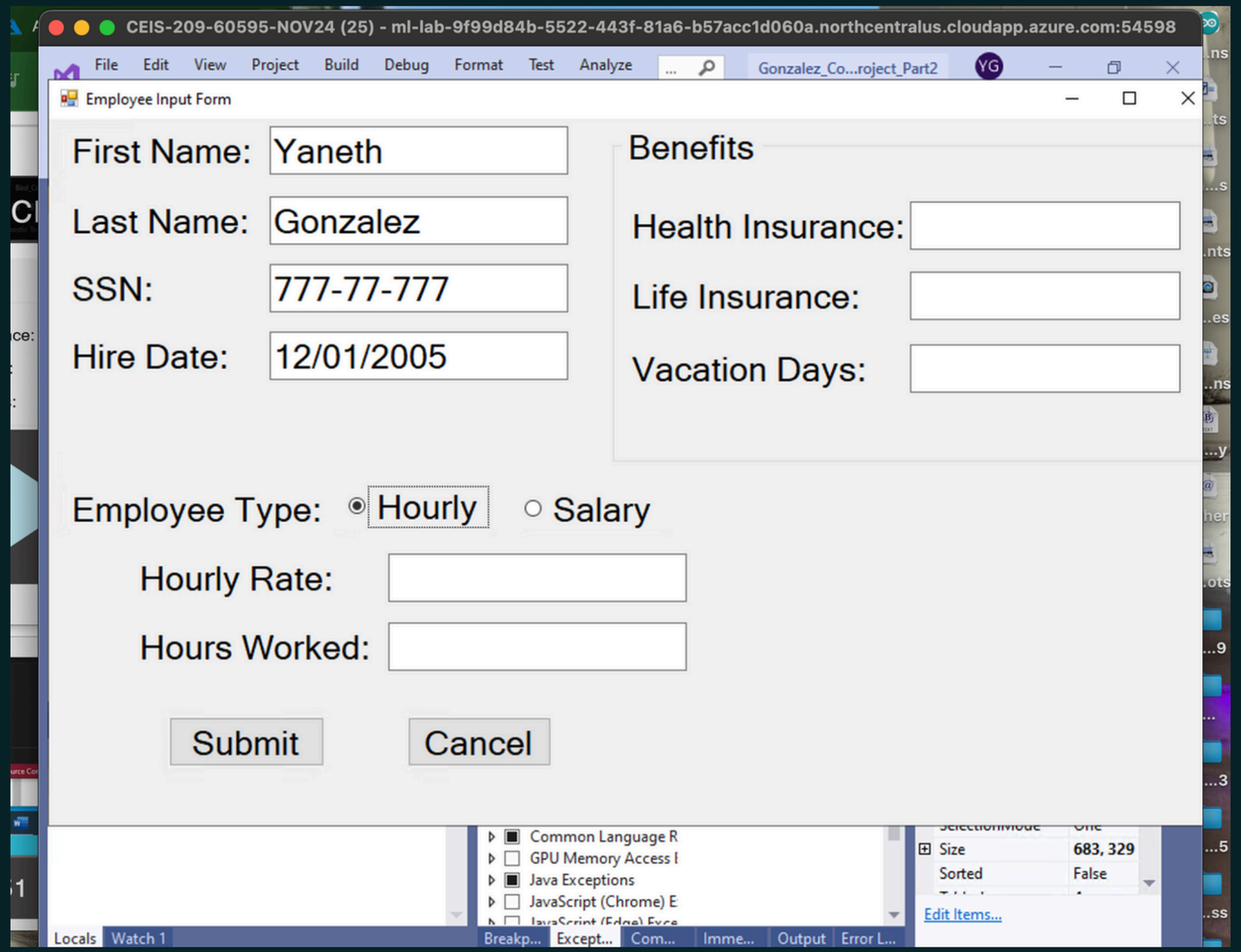
```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    public class Benefits
    {
        //attributes
        private string healthInsurance;
        private int lifeInsurance;
        private int vacation;

        // constructors
        public Benefits()
        {
            healthInsurance = "N/A";
            lifeInsurance = 0;
            vacation =0;
        }

        public Benefits(string healthInsurance, int lifeInsurance, int vacation)
        {
            this.healthInsurance = healthInsurance;
            this.lifeInsurance = lifeInsurance;
            this.vacation = vacation;
        }

        // behavior
        public override string ToString()
        {
            return "healthInsurance=" + healthInsurance
                + ", lifeInsurance=" + lifeInsurance
                + ", vacation=" + vacation;
        }

        // properties
        public string HealthInsurance
        {
            get { return healthInsurance; }
            set { healthInsurance = value; }

        }

        public int LifeInsurance
        {
            get { return lifeInsurance; }
            set { lifeInsurance = value; }
        }

        public int Vacation
        {
            get { return vacation; }
            set { vacation = value; }
        }
    }
```

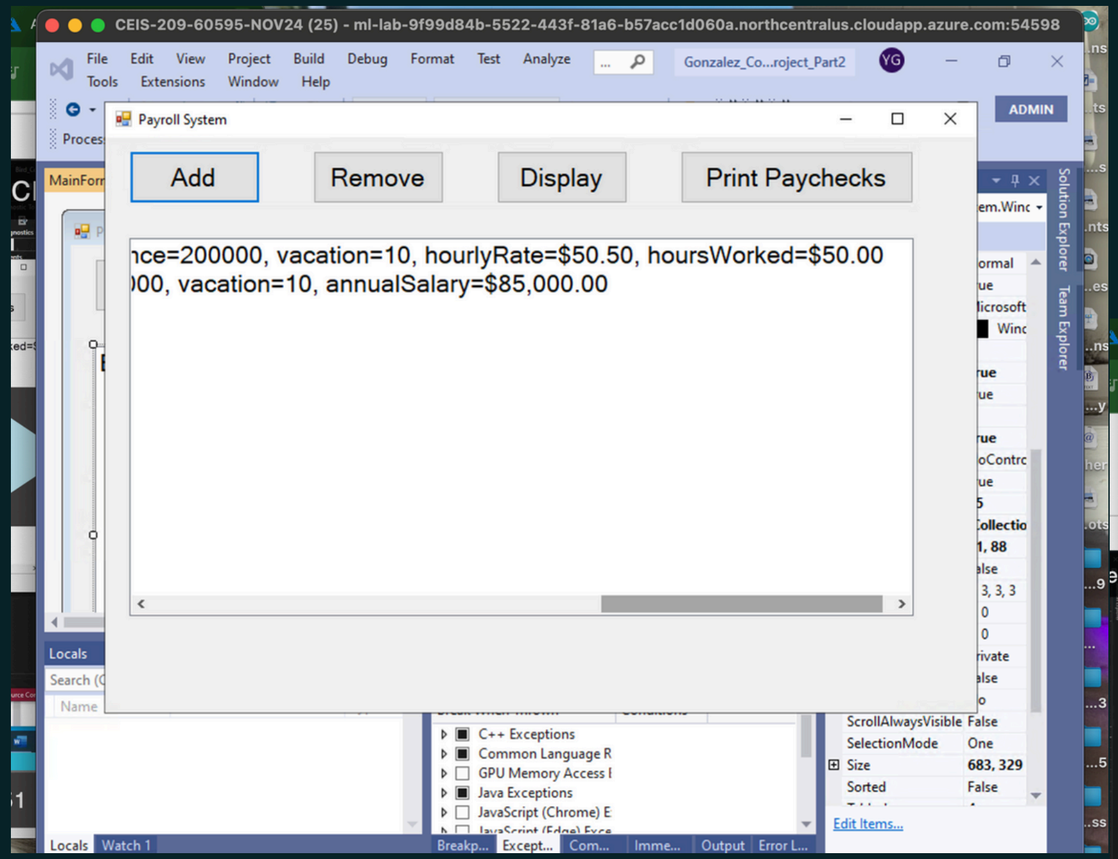
Module 7:

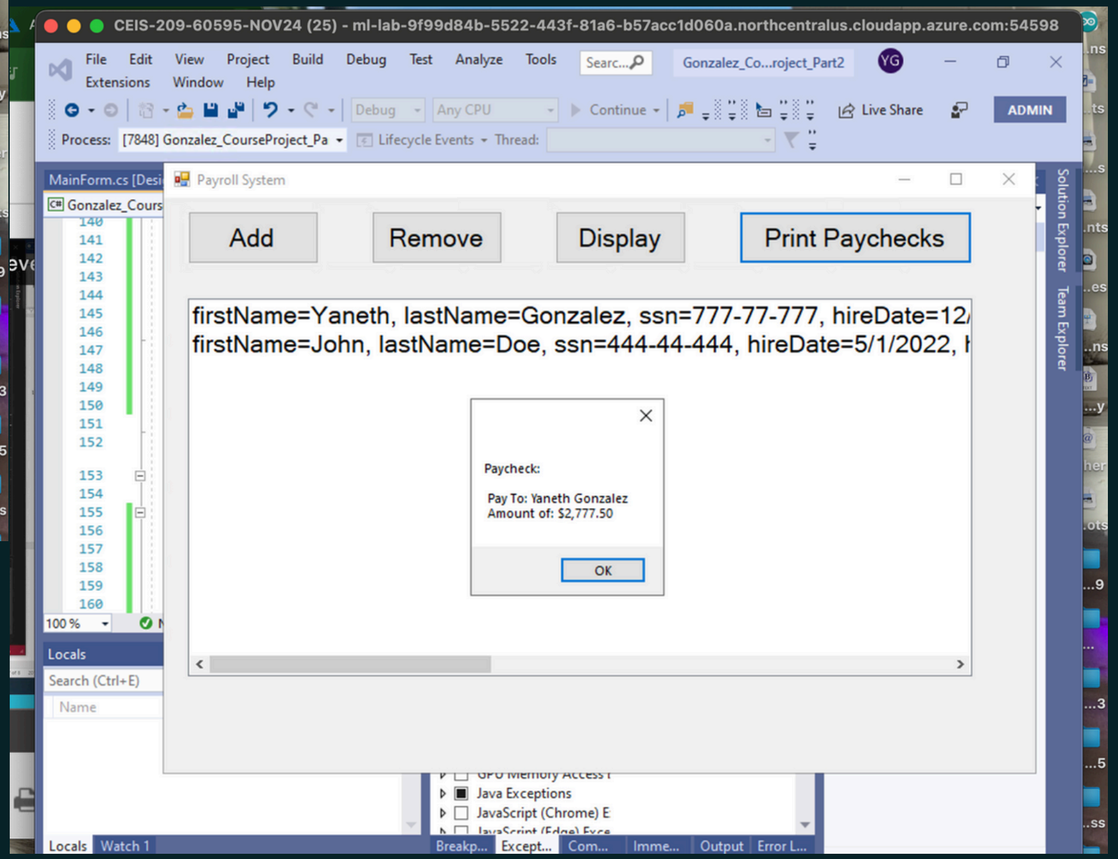
In Module 7, we learned how to handle paying different types of employees in one program using inheritance and polymorphism. Instead of creating separate programs for hourly, salaried, or other types of employees, we used a shared Employee class as a base.

- We made the Employee class abstract, meaning it can’t be used directly but serves as a template for other classes like Salary and Hourly. Each child class added its own details, like pay rates or hours worked, and defined how to calculate pay.
- Polymorphism allowed us to use one list for all employees, treating them the same while still handling their unique pay calculations.
- We updated the input form so users could choose an employee type, and the program showed the right fields for hourly or salaried employees. The data could be saved to a file and loaded back later.

Finally, we added a paycheck feature to calculate and display each employee's pay. This approach kept the program simple and flexible, making it easy to handle different employee types in one place.


CEIS-209-60595-NOV24 (25) - ml-lab-9f99d84b-5522-443f-81a6-b57acc1d060a.northcentralus.cloudapp.azure.com:54598
File
Edit
View
Project
Build
Debug
Format
Test
Analyze
Gonzalez_Co...roject_Part2
Employee Input Form
First Name:
Yaneth
Last Name:
Gonzalez
SSN:
777-77-777
Hire Date:
12/01/2005
Benefits
Health Insurance:
Life Insurance:
Vacation Days:
Employee Type:
Hourly
Salary
Hourly Rate:
Hours Worked:
Submit
Cancel
Common Language R
GPU Memory Access I
Java Exceptions
JavaScript (Chrome) E
Size
683, 329
Sorted
False
Edit Items...
Locals
Watch 1
Breakp...
Except...
Com...
Imme...
Output
Error L...


CEIS-209-60595-NOV24 (25) - ml-lab-9f99d84b-5522-443f-81a6-b57acc1d060a.northcentralus.cloudapp.azure.com:54598
Payroll System
Add
Remove
Display
Print Paychecks
nce=200000, vacation=10, hourlyRate=$50.50, hoursWorked=$50.00
)00, vacation=10, annualSalary=$85,000.00


CEIS-209-60595-NOV24 (25) - ml-lab-9f99d84b-5522-443f-81a6-b57acc1d060a.northcentralus.cloudapp.azure.com:54598
Payroll System
Add
Remove
Display
Print Paychecks
firstName=Yaneth, lastName=Gonzalez, ssn=777-77-777, hireDate=12/
firstName=John, lastName=Doe, ssn=444-44-444, hireDate=5/1/2022, h
Paycheck:
Pay To: Yaneth Gonzalez
Amount of: $2,777.50
OK

MainForm.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Runtime.Serialization.Formatters.Binary;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace Gonzalez_CourseProject_Part2
{
    public partial class MainForm : Form
    {
        // class level reference
        private const string FILENAME = " Employee.dat ";
        public MainForm()
        {
            InitializeComponent();
        }

        private void AddButton_Click(object sender, EventArgs e)
        {
            // add item to the Employee listbox
            InputForm frmInput = new InputForm();

            using (frmInput)
            {
                DialogResult result = frmInput.ShowDialog();

                // see if input form was cancelled
                if (result == DialogResult.Cancel)
                    return;   // end the method since cancelled

                // get user's input and create Employee object
                string fName = frmInput.FirstNameTextBox.Text;
                string lName = frmInput.LastNameTextBox.Text;
                string ssn = frmInput.SSNTextBox.Text;
                string date = frmInput.HireDateTextBox.Text;
                DateTime hireDate = DateTime.Parse(date);
                string healthIns = frmInput.HealthInsuranceTextBox.Text;
                int lifeIns = int.Parse(frmInput.LifeInsuranceTextBox.Text);
                int vacation = int.Parse(frmInput.VactionDaysTextBox.Text);

                Benefits ben = new Benefits(healthIns, lifeIns, vacation);
```

```
                Employee emp = null; // empty reference

                if(frmInput.SalaryRadioButton.Checked )
                {
                    double salary = double.Parse(frmInput.SalaryTextBox.Text);
                    emp = new Salary(fName, lName, ssn, hireDate, ben, salary);
                }
                else if ( frmInput.HourlyRadioButton.Checked)
                {
                    double hourlyRate = double.Parse(frmInput.HourlyRateTextBox.Text);
                    double hoursWorked = double.Parse(frmInput.HoursWorkedTextBox.Text);
                    emp = new Hourly(fName, lName, ssn, hireDate, ben, hourlyRate, hoursWorked);
                }
                else
                {
                    MessageBox.Show("Error. Please select an employee type");
                    return; // end the method
                }


                // add the Employee object to the Employee listbox
                EmployeesListBox.Items.Add(emp);

                // write all date to the file
                WriteEmpsToFile();


            }


        }
        private void WriteEmpsToFile()
        {
            // convert the ListBox items to a generic list
            List<Employee> empList = new List<Employee>();

            foreach(Employee emp in EmployeesListBox.Items)
            {
                empList.Add(emp);
            }

            // open a pipe to the file and create a translator
            FileStream fs = new FileStream(FILENAME, FileMode.Create);
            BinaryFormatter formatter = new BinaryFormatter();

            // write the generic list to the file
            formatter.Serialize(fs, empList);

            // close the pipe
```

```
            fs.Close();

        }

        private void RemoveButton_Click(object sender, EventArgs e)
        {
            //remove the selected items from the Employee listbox
            int itemNumber = EmployeesListBox.SelectedIndex;

            if(itemNumber > -1)
            {
                EmployeesListBox.Items.RemoveAt(itemNumber);
                WriteEmpsToFile();

            }
            else
            {
                MessageBox.Show("Please select employee to remove.");

            }

        }

        private void DisplayButton_Click(object sender, EventArgs e)
        {


            // check to see if the file exists
            if(File.Exists(FILENAME) && new FileInfo(FILENAME).Length> 0)
            {
                // create a pipe from the file and create a translator
                FileStream fs = new FileStream(FILENAME, FileMode.Open);
                BinaryFormatter formatter = new BinaryFormatter();

                // read the generic list from file
                List<Employee> list = (List<Employee>)formatter.Deserialize(fs);

                // close the pipe
                fs.Close();

                // clear the Employee listbox
                EmployeesListBox.Items.Clear();

                foreach (Employee emp in list)
                    EmployeesListBox.Items.Add(emp);

            }
```

(Continue) **MainForm.cs**

```
}

private void PrintPaychecksButton_Click(object sender, EventArgs e)
{
    foreach (Employee emp in EmployeesListBox.Items)
    {
        string line1 = " Pay To: " + emp.FirstName + " " + emp.LastName;
        string line2 = " Amount of: " + emp.CalculatePay().ToString("C2");

        string output = "Paycheck:\n\n" + line1 + "\n" + line2;

        MessageBox.Show(output);

    }
}

private void EmployeesListBox_DoubleClick(object sender, EventArgs e)
{
    // get the selected Employee object
    Employee emp = EmployeesListBox.SelectedItem as Employee;

    // show the Input/Update form with the Employee info
    InputForm frmUpdate = new InputForm();
    frmUpdate.FirstNameTextBox.Text = emp.FirstName;
    frmUpdate.LastNameTextBox.Text = emp.LastName;
    frmUpdate.SSNTextBox.Text = emp.SSN;
    frmUpdate.HireDateTextBox.Text = emp.HireDate.ToShortDateString();
    frmUpdate.HealthInsuranceTextBox.Text = emp.BenefitsEmp.HealthInsurance;
    frmUpdate.LifeInsuranceTextBox.Text = emp.BenefitsEmp.LifeInsurance.ToString();
    frmUpdate.VactionDaysTextBox.Text = emp.BenefitsEmp.Vacation.ToString();

    // check to see if emp is a Salary or Hourly object
    if (emp is Salary )
    {
        frmUpdate.HourlyRateLabel.Visible = false;
        frmUpdate.HourlyRateTextBox.Visible = false;
        frmUpdate.HoursWorkedLabel.Visible = false;
        frmUpdate.HoursWorkedTextBox.Visible = false;
        frmUpdate.SalaryLabel.Visible = true;
        frmUpdate.SalaryTextBox.Visible = true;

        // mark the Salary radiobutton as checked
        frmUpdate.SalaryRadioButton.Checked = true;

        // convert the Employee reference to a Salary object
        Salary sal = (Salary)emp;

        // show the Salary information
        frmUpdate.SalaryTextBox.Text = sal.AnnualSalary.ToString("F2");

    }
```

```
else if (emp is Hourly )
{
    frmUpdate.HourlyRateLabel.Visible = true;
    frmUpdate.HourlyRateTextBox.Visible = true;
    frmUpdate.HoursWorkedLabel.Visible = true;
    frmUpdate.HoursWorkedTextBox.Visible = true;
    frmUpdate.SalaryLabel.Visible = false;
    frmUpdate.SalaryTextBox.Visible = false;

    // mark the Hourly radiobutton as checked
    frmUpdate.HourlyRadioButton.Checked = true;

    // convert the Employee reference to a Hourly object
    Hourly hrly = (Hourly)emp;

    // show the Hourly information
    frmUpdate.HourlyRateTextBox.Text = hrly.HourlyRate.ToString("F2");
    frmUpdate.HoursWorkedTextBox.Text = hrly.HoursWorked.ToString("F2");

}
else
{
    MessageBox.Show("Error. Invalid employee type found.");
    return; // end the method
}

DialogResult result = frmUpdate.ShowDialog();

// if cancelled, stop the method
if (result == DialogResult.Cancel)
    return;  // end the method

// delete the selected object
int position = EmployeesListBox.SelectedIndex;
EmployeesListBox.Items.RemoveAt(position);

// create new employee using the updating information
Employee newEmp = null;

string fName = frmUpdate.FirstNameTextBox.Text;
string lName = frmUpdate.LastNameTextBox.Text;
string ssn = frmUpdate.SSNTextBox.Text;
DateTime hireDate = DateTime.Parse(frmUpdate.HireDateTextBox.Text);
string healthInsurance = frmUpdate.HealthInsuranceTextBox.Text;
int lifeInsurance =int.Parse(frmUpdate.LifeInsuranceTextBox.Text);
int vacation = int.Parse(frmUpdate.VactionDaysTextBox.Text);


Benefits ben = new Benefits(healthInsurance, lifeInsurance, vacation);

if(frmUpdate.SalaryRadioButton.Checked )
```

```
    {
        double salary = double.Parse(frmUpdate.SalaryTextBox.Text);
        newEmp = new Salary(fName, lName, ssn, hireDate, ben, salary);
    }
    else if( frmUpdate.HourlyRadioButton.Checked)
    {
        double hourlyRate = double.Parse(frmUpdate.HourlyRateTextBox.Text);
        double hoursWorked = double.Parse(frmUpdate.HoursWorkedTextBox.Text);
        newEmp = new Hourly(fName, lName, ssn, hireDate, ben, hourlyRate, hoursWorked);
    }
    else
    {
        MessageBox.Show("Error. Invalid employee type.");
        return; // end the method
    }

    // add the new employee to the listbox
    EmployeesListBox.Items.Add(newEmp);


    }

    private void EmployeesListBox_SelectedIndexChanged(object sender, EventArgs e)
    {

    }
}
}
```

Hourly.cs

HOURLY FORM:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    [Serializable]
    public class Hourly : Employee
    {
        // attributes
        private double hourlyRate;
        private double hoursWorked;

        // constructors
        public Hourly() : base()
        {
            hourlyRate = 0.0;
            hoursWorked = 0.0;
        }
        public Hourly(string firstName, string lastName,
            string ssn, DateTime hireDate, Benefits benefits,
            double hourlyRate, double hoursWorked)
            : base(firstName, lastName, ssn, hireDate, benefits)
        {
            this.hourlyRate = hourlyRate;
            this.hoursWorked = hoursWorked;
        }

        // behaviors
        public override double CalculatePay()
        {
            double pay = 0.0;

            if( hoursWorked> 40.0)
            {
                double basePay = 40.0 * hourlyRate;
                double overtime = (hoursWorked - 40.0) * hourlyRate * 1.5;
                pay = basePay + overtime;
            }
            else
            {
                pay = hoursWorked * hourlyRate;
            }

            return pay;

        }

        public override string ToString()
        {
            return base.ToString() + ", hourlyRate="
                + hourlyRate.ToString("C2")
                + ", hoursWorked=" + hoursWorked.ToString("C2");
        }

        // properties
        public double HourlyRate
        {
            get { return hourlyRate; }
            set { hourlyRate = value; }

        }

        public double HoursWorked
        {
            get { return hoursWorked; }
            set { hoursWorked = value; }
        }
```

Salary.cs

```
    }
}
```

SALARY FORM:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    [Serializable]
    public class Salary : Employee
    {
        // attributes
        private double annualSalary;

        // constructors
        public Salary() : base()
        {
            annualSalary = 0.0;

        }

        public Salary(string firstName, string lastName,
            string ssn, DateTime hireDate, Benefits benefits,
            double annualSalary)
            : base(firstName, lastName, ssn, hireDate, benefits)
        {
            this.annualSalary = annualSalary;
        }

        // behaviors
        public override double CalculatePay()
        {
            return annualSalary / 26.0;
        }

        public override string ToString()
        {
            return base.ToString() + ", annualSalary="
                + annualSalary.ToString ("C2");
        }

        // properties
        public double AnnualSalary
        {
            get { return annualSalary; }
            set { annualSalary = value; }
```

Employee .cs

```
        }
    }
}
```

EMPLOYEE FORM:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    [Serializable]
    public abstract class Employee
    {
        // attributes
        private string firstName;
        private string lastName;
        private string ssn;
        private DateTime hireDate;
        private Benefits benefits;

        // constructors
        public Employee()
        {
            firstName = "N/A";
            lastName = "N/A";
            ssn = "N/A";
            hireDate = new DateTime();
            benefits = new Benefits();
        }
        public Employee(string firstName, string lastName, string ssn, DateTime hireDate,
Benefits benefits)
        {
            this.firstName = firstName;
            this.lastName = lastName;
            this.ssn = ssn;
            this.hireDate = hireDate;
            this.benefits = benefits;
        }

        // behaviors
        public override string ToString()
        {
            return "firstName=" + firstName
                + ", lastName=" + lastName
                + ", ssn=" + ssn
                + ", hireDate=" + hireDate.ToShortDateString()
                + ", healthInsurance=" + benefits.HealthInsurance
                + ", lifeInsurance=" + benefits.LifeInsurance
                + ", vacation=" + benefits.Vacation;
```

Benefit.cs

```
        }
```

BENEFIT FORM:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace Gonzalez_CourseProject_Part2
{
    [Serializable]
    public class Benefits
    {
        //attributes
        private string healthInsurance;
        private int lifeInsurance;
        private int vacation;

        // constructors
        public Benefits()
        {
            healthInsurance = "N/A";
            lifeInsurance = 0;
            vacation =0;
        }

        public Benefits(string healthInsurance, int lifeInsurance, int vacation)
        {
            this.healthInsurance = healthInsurance;
            this.lifeInsurance = lifeInsurance;
            this.vacation = vacation;
        }

        // behavior
        public override string ToString()
        {
            return "healthInsurance=" + healthInsurance
                + ", lifeInsurance=" + lifeInsurance
                + ", vacation=" + vacation;
        }
```

CHALLENGES

Throughout this course, I faced several challenges that pushed me to grow as a programmer.
In the beginning, understanding how to structure a GUI layout and connect it to the logic behind the scenes was tricky.

- Debugging errors during input validation and event handling required patience and attention to detail. Managing multiple classes, like Employee and Benefits, while ensuring they worked together seamlessly was challenging, especially when incorporating concepts like encapsulation and composition.
- Learning inheritance and updating methods to handle different employee types tested my ability to think through complex relationships between classes.
- Utilizing Azure through my school was another challenge because my MacBook Pro wasn't compatible with downloading Visual Studio. At times, Azure ran slowly, was glitchy, and even caused my files to be lost, leaving me feeling frustrated. File handling also posed difficulties, particularly in ensuring data was saved and loaded correctly without errors.

Despite these obstacles, each challenge taught me persistence and problem-solving skills, making me a stronger and more confident programmer by the end of the course

CAREER SKILLS

This course helped me build career-ready programming skills that I can use in real-world projects.

- I learned how to design apps that are organized, easy to update, and built for long-term use by using object-oriented programming concepts like encapsulation, composition, and inheritance.
- Working on a payroll system project taught me how to handle data and connect different parts of an application smoothly.
- I also gained experience designing user-friendly interfaces with Windows Forms, which is important for creating apps people will enjoy using. File handling was another key skill, helping me learn how to save and load data reliably.
- Adapting to cloud-based tools like Azure taught me how to work with platforms outside my comfort zone, which is important in the tech field. Debugging and solving problems strengthened my persistence and attention to detail.

This course gave me the tools and confidence to grow in my career as a programmer.

CONCLUSION

This course was a challenging yet rewarding experience that helped me grow as a programmer.

- I learned how to design functional and user-friendly applications, organize data efficiently, and use advanced programming concepts like object-oriented techniques. Working on the payroll system project gave me hands-on experience, teaching me how to solve real-world problems and adapt to challenges, such as using Azure to overcome technical limitations with my MacBook Pro.
- These skills have not only strengthened my technical abilities but also boosted my confidence in tackling complex tasks.

This course has laid a strong foundation for my career in programming and has motivated me to continue learning and improving. I'm excited to apply what I've learned to future projects and take the next steps toward my goals in technology.

REFERENCE

- Module 1 Course Project Create a Windows Form Application With Basic Controls [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676628?module_item_id=15082454
- Module 2 Course Project Validating Input from a Form and Looping Through Songs [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676636?module_item_id=15082554
- Module 3 Course Project Writing utility methods and Storing Song Data in Arrays [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676644?module_item_id=15082668
- Module 4 Course Project Create a Windows Forms Application With Basic Controls and Add an Employee Class [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676649?module_item_id=15082738
- Module 5 Course Project Encapsulating Data and Writing to a File [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676653?module_item_id=15082771
- Module 6 Coure Project Composition [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/assignments/3676658?module_item_id=15082803
- Module 7 Final Course Project Inheritance [Video]. Devryu.Instructure.com. https://devryu.instructure.com/courses/116154/pages/module-7-final-course-project-due-module-8?module_item_id=15082844

THANK YOU

Presentation by

YANETH GONZALEZ