# Introduction

## Problem

In this lab, you will create a Windows Forms movie collection program. You will only be collecting information about a single movie in this lab. The program will behave similar to the previous lab except it will be using forms instead of a Console window.

## Solution

Your program will consist of a main form, a couple of child forms for collecting and displaying information and a menu to allow selecting options.

Note: All screenshots are for demonstration purposes. You can make adjustments to meet your needs.

# Setting Up the Solution

To prepare for later labs where you will have different layers of your application, you will be breaking up your code into multiple projects. For each future lab, you will be creating a new solution but you will be able to copy over the existing projects that you have already created as needed.

1. Create a new solution.
   1. Create a new solution (Other Project Type\Visual Studio Solutions).
   2. Call it Lab2.
   3. Ensure the Location is under your Labs folder in your local Git repository so it can be committed to Github.
   4. It is strongly recommended that you keep each lab in its own folder. Visual Studio will attempt to enforce this but will allow you to override it.
2. Add the main application project. This will be the main application.
   1. Add a new Windows Forms project (Visual C#\Windows Forms App (.NET Framework)) to the solution.
   2. Call it MovieLib.Windows.
   3. Set it as the startup project.
   4. Rename the Form1.cs file to MainForm. Ensure that you rename the type as well when prompted.
3. Add the business project. This is where business types will go.
   1. Add a new class library project (Visual C#\Class Library (.NET Framework)) to the solution.
   2. Call it MovieLib.
   3. Remove the Class1.cs file.
4. Add a reference to the business project in the application project so it can use the functionality.
   1. Right click the application project and select Add\Reference.
   2. Alternatively, right click the References folder in Solution Explorer and select Add Reference.
   3. Select the Projects\Solution node in the tree on the left.
   4. Check the box for the business project on the right.
   5. Click OK to add a reference.
5. Build the code to ensure everything is working correctly.

# Main Form

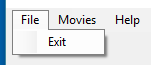
The main form should be set up as follows.

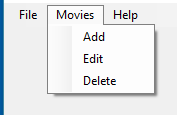
1. Window title should say Movie Library.
2. The Minimize, Maximize and Close buttons should be visible in the title bar.
3. The form should be at least 700 x 400 in size.
4. The content area will be empty other than a menu.

# Main Menu

The main menu will have the following structure where top level items are at the root of the menu and the child items are children under the menu header.

* File
  + Exit
* Movies
  + Add
  + Edit
  + Delete
* Help
  + About







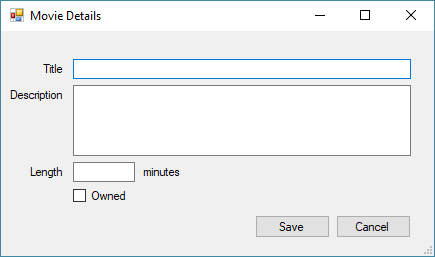
The behavior of each menu item is listed under Commands later. Use descriptive names for the menu item fields so they are easier to identify later.

# Movie Detail Form

Create a new form (Windows Form) that will display the details of a movie. It will allow the user to edit the details of a movie. It will be used for adding new movies and editing existing movies. Ensure it has a reasonable title and size.

The form will consist of the following.

1. A Label and TextBox for providing the title of the movie. Provide enough space for a title (assume an average of 50 characters).
2. A Label and TextBox for providing the description of the movie. Use a multiline field that allows at least 3 lines of text.
3. A Label and TextBox for providing the length of the movie. Provide enough space for the length (assume 90 to 240 minutes). Make it clear that the length is in minutes.
4. A Label and CheckBox for indicating if the movie is owned or not.
5. A Button to save the changes.
6. A Button to cancel the changes.

For all fields ensure the labels line up as do the controls in a reasonable fashion. Ensure the tab ordering is set such that the user can tab between each of the fields. The buttons should be grouped at the bottom together. Use reasonable names for the interactive fields.

When the form loads, if a movie is associated with it then the details of the movie will be filled into the fields on the form. Otherwise the fields will be empty.

When the Save button is selected then validate the fields (as defined later). If the fields are valid then save the values to either the movie that was provided or a new movie. Then close the form. If the fields are invalid then display an error and return to the form.

When the Cancel button is selected then close the form without saving any changes.

Ensure when returning from the form that the appropriate dialog result is returned so the parent form can react accordingly.

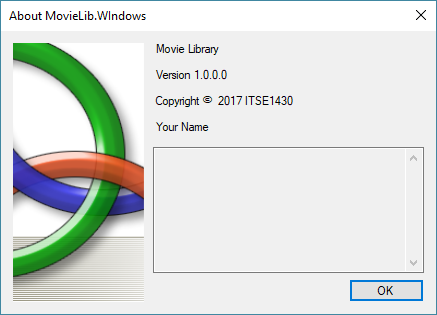
## Movie Validation

The following rules should be enforced when validating the movie.

1. Title cannot be empty.
2. Length must either be empty or a value >= 0.

# About Form

Create a new form (About Box) to display information about your program. The form should show, at a minimum, the name of the program and your name.



The data is provided by the project settings. To set the project settings go to the project’s properties, then Application and then Assembly Information. Set the following values.

1. Product – Movie Library
2. Company – Your Name
3. Copyright – Add ITSE 1430 to the end

# Movie Class

The information about a movie will be stored in a Movie class. The class represents a business object and should be in the business project. Ensure the class is public so it accessible outside the project. Ensure the class is properly doc commented.

The class will provide properties for the information that is needed by the application. Fields will be used to store the underlying data but not accessible outside the class.

For review, a movie will consist of the following information

|  |  |  |  |
| --- | --- | --- | --- |
| **Name** | **Type** | **Required** | **Comments** |
| Title | Text | Y | Unique title of the movie. |
| Description | Text | N | Optional description of the movie. |
| Length | Integer | N | Length of the movie in minutes (default to 0). If specified then it must be >= 0. |
| Owned | Boolean | Y | True if the movie is owned or false if it is on the wishlist. |

## Storing the Movie

To keep things simple, store an instance of the Movie class as a field of the main form. Pass the instance to any code that needs it. It is strongly recommended that you do not initialize this field. A null movie indicates a movie has not been added yet and/or it has been deleted. This is a valid situation that some commands will check for.

# Commands

## File \ Exit

Close the application.

## Movie \ Add

Display the Movie Detail form. The form should start out empty. If the user selects the OK option then set the form’s movie to the new movie otherwise do nothing.

## Movie \ Edit

If there is no movie defined yet then display an error message. Otherwise display the Movie Detail form. The form should display the values of the form’s movie. If the user selects the OK option then update the form’s movie to the new, edited movie otherwise do nothing.

It is strongly recommended that you copy the existing movie details into a temporary movie object while editing. If the user decides to cancel the editing or if they attempt to save the changes and validation fails (will be added later) then you do not want to have the form corrupt the original movie data.

## Movie \ Remove

If there is no movie defined yet then display an error message. Otherwise display a confirmation message that includes the movie title. If the user selects Yes then clear the form’s movie. If the user selects No then do nothing.

## Help \ About

Display the About form.

# Requirements

Your program must meet all the following requirements.

* Compile cleanly with no warnings or errors.
* Meet all the points mentioned in the solution.
* Ensure each file has a file header indicating the course, your name and date.
* Solution, project and code should be uploaded to Github
* Submit lab in MyTCC by providing link to Github URL