PROJECT: WPF MASTER/DETAIL APP S1 (THE DESIGN PATTERN SETUP AND DETAIL VIEW)

OVERVIEW

The goal of Sprint 1 is to develop the framework for a single window N-tier WPF application using the MVP Design Pattern. The **View** and **Presenter** will display a detail view using binding and styles.

NOTES

- Students are encouraged to use all resources available including their instructor and peers.

USER STORIES

Epic	User Story	Acceptance Criteria
As a user, I need to see detail information for one product.	As a user, I will open the application, so that the detail of a product and company information is displayed in the window.	The window has an area that displays all product details.
		The window has an area that displays the company information.
	As a user, I will click the **Quit** button, so the application will close.	The window closes and the application exits when the **Quit** button is pressed.

INSTRUCTIONS

1. Download the ***WPF_MasterDetailApp*** solution from GitHub.
2. Extract and open the solution in Visual Studio.
3. Run the ***WPF_MasterDetailApp.S1.Sol*** project to confirm that it works and see a completed example of Sprint 1.
4. Right click the ***WPF_MasterDetailApp.S1.Sol*** project and choose **Unload Project**. The code in the S1 solution project will be unavailable and safe. Right clicking and choosing **Reload Project** will make it available again if access to the working code is needed.
5. Setup the **N-tier Pattern**.
    a. Add a method, **Application_Startup**, to the **App.xaml.cs** file to handle the Startup event by instantiating a ***ProductBL*** object.

```
namespace WPF_MasterDetailApp
{
    /// <summary>
    /// Interaction logic for App.xaml
    /// </summary>
    public partial class App : Application
    {
        private void Application_Startup(object sender, StartupEventArgs e)
        {
            ProductBL productViewerBL = new ProductBL();
        }
    }
}
```

b. Modify the **App.xaml StartUpUri** attribute to **StartUp** and call the **Application_Startup** method.

```
<Application x:Class="WPF_MasterDetailApp.App"
             xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
             xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
             xmlns:local="clr-namespace:WPF_MasterDetailApp"
             Startup="App_Startup">
    <Application.Resources>

    </Application.Resources>
</Application>
```

c. Open the ***ProductBL*** class file in the **BusinessLayer** and add a field for both the **View** and the **Presenter**.

```
#region FIELDS

ProductWindowView _productWindowView;
ProductWindowPresenter __productWindowPresenter;

#endregion
```

d. Update the constructor with the following code. Note that some code will have errors until subsequent code is added.

```
#region CONSTRUCTORS

public ProductBL()
{
    //
    // instantiate the view model and initialize the data set
    //

    __productWindowPresenter = new ProductWindowPresenter(GetCompanyData(), GetProductData());

    //
    // instantiate, set the data context, and show the Main Window
    //
    _productWindowView = new ProductWindowView(__productWindowPresenter);
    _productWindowView.DataContext = __productWindowPresenter;
    _productWindowView.Show();
}

#endregion
```

6. Add the following properties to the ***Company*** class in the **Models** folder. Note that the C# Auto Implemented Property has been used ("prop" – Tab –Tab) and that no fields were created. This is a simple class and will only hold information about the company and so fields are unnecessary.

```
public class Company
{
    public string Name { get; set; }
    public string Address { get; set; }
    public string City { get; set; }
}
```

7. Setup the ***Product*** class in the Models folder.
    a. Choose a real or fictional product for the application.
    b. Add a minimum of six field/property pairs to the ***Product*** class using the following types; string, double, int, enums and DateTime properties.
    c. Add a field/property pair for the **ImageName**.
8. Choose one product to build out the detail view in the window.
    a. Add an image of the chosen object to the **Images** folder.

b. Add the ***GetProductData*** and ***GetCompanyData*** methods in the ***ProductBL*** class located in the ***BusinessLayer*** folder and include the specific information for the company and product. These methods will generate the seed data for the **Presenter**.

Example:

```
#region METHODS

private Company GetCompanyData()
{
    return new Company()
    {
        Name = "Troglodyte Talent Agency",
        Address = "465 Jurassic Lane",
        City = "Bedrock"
    };
}

private Product GetProductData()
{
    return new Product()
    {
        Id = 1,
        FirstName = "Fred",
        LastName = "Flintstone",
        Age = 28,
        Gender = Product.GenderType.male,
        ImageFileName = "fred_flintstone.jpg",
        Description = "Fred is the main character of the series. He's an accident-prone bronto-crane operator at the Slate Rock and Gravel
          Company and the head of the Flintstone household. He is quick to anger (usually over trivial matters), but is nonetheless a very
          loving husband and father. He's also good at bowling and is a member of the fictional Loyal Order of Water Buffaloes (Lodge No.
          26), a men-only club paralleling real-life fraternities such as the Loyal Order of Moose.",
        HireDate = DateTime.Parse("03-23-1963"),
        AverageAnnualGross = 23445.85
    };
}
```

9. Update the ***ProductWindowPresenter*** class in the **PresentationLayer** folder. (Note: This file will be referred to as the **Presenter**.)

a. The **Presenter** will have two objects to provide to the **View**, a ***Company*** and ***Product*** object. Add the fields and properties.

```
#region FIELDS

private Company _companyInfo;
private Product _selectedProduct;

#endregion

#region PROPERTIES

public Company CompanyInfo
{
    get { return _companyInfo; }
    set { _companyInfo = value; }
}

public Product SelectedProduct
{
    get { return _selectedProduct; }
    set { _selectedProduct = value; }
}

#endregion
```

b. Both objects will be provided to the **Presenter** via the constructor as arguments when it is instantiated and the constructor is called. Add the constructor with the **Company** and **Product** as parameters in the constructor.

```
#region CONSTRUCTORS

public ProductWindowPresenter(Company company, Product product)
{
    _selectedProduct = product;
    _companyInfo = company;
}

#endregion
```

c. Add a **QuitApplication** method to handle the **Click** event from the **View**.

```
#region METHODS

public void QuitApplication()
{
    Environment.Exit(0);
}

#endregion
```

10. Update the code behind for the ***ProductWindowView*** class in the **PresentationLayer** folder. (Note: This file will be referred to as the **View**.)
    a. Add a field for the Presenter.

```
#region FIELDS

ProductWindowPresenter _productWindowPresenter;

#endregion
```

    b. Modify the constructor to set the **Presenter** filed and set the **DataContext**.

```
#region CONSTRUCTORS

public ProductWindowView(ProductWindowPresenter productPresenter)
{
    _productWindowPresenter = productPresenter;

    DataContext = _productWindowPresenter;

    InitializeComponent();
}

#endregion
```

    c. Add a method to handle the **Click** event for the **Quit** button.

```
#region METHODS (pass events to view model)

private void Button_Quit_Click(object sender, RoutedEventArgs e)
{
    _productWindowPresenter.QuitApplication();
}

#endregion
```

11. Build the solution, confirm that it compiles and opens a blank window.

- Do not move on until the solution compiles runs and opens a blank window. -

12. Develop the ***ProductWindowView*** class in the **PresentationLayer** folder per the user stories in the sprint. (Note: This file will be referred to as the **View**.)
    a. Add code to the **ProductWindowView.xaml** file.
        i. Resize the window per the wireframe diagram.
        ii. Change the **Title** of the application.
        iii. Code the xaml in the **View** to display the company info and product detail as the wireframe diagram indicates.
        iv. Bind all text boxes to the appropriate property in the ***Company*** and ***Product*** classes.
        v. Add a Quit button with a Click event.
13. Build the solution, confirm that it compiles and opens a window that displays all company and product information correctly.

14. Style the **View**.
    a. Add the following styles in the **Window.Resources** tag.
        i. **HeaderStyle** (Company Title Label)
        ii. **SubHeaderStyle** (Company Address Label)
        iii. **DetailLabelStyle** (Product Property Labels)
        iv. **DetailTextBoxStyle** (Product Property Display/Edit TextBoxes)
        v. **ButtonStyle** (Window Buttons)
    b. Apply the styles to the appropriate elements on the window.
15. Build the solution, confirm that it compiles, opens a window that displays all company and product information correctly styled, and that the **Quit** button closes the application.

SUBMIT FOR GRADE

1. Prepare for submission.
    a. Run the application to confirm that all implemented user stories are fully functional and tested. Any user stories that are not fully functional **MUST** be noted on the **Sprint 1 Deliverables Marking Guide and Checklist** and commented out in the code. Remember, a sprint deliverable should be a robust solution and able to be run by the stakeholders without any issues per the task list.
    b. Download and complete the **Sprint 1 Deliverables Marking Guide and Checklist.**
        i. Complete the **User Stories Checklist** and give full credit for each item completed.
        ii. Highlight all incomplete items in yellow.
        iii. Give credit for each item submitted.
        iv. Self-score the **Current Progress**.
        v. Total all scores.
    c. Create a video presentation (less than five minutes).
        i. State the course name, project name, author, and date.
        ii. Briefly describe the goals of the sprint.
        iii. Demonstrate all functioning user stories.
        iv. Discuss any issues. State if and how they were resolved.
        v. Display the completed **Sprint 1 Deliverables Marking Guide and Checklist**. Justify the **Current Progress** grade.
    d. Push the most current version of the solution to GitHub.
2. Login to Moodle and open the ***Project: The Master/Detail App S1 (Benchmark)*** assignment. Note: Submissions will not be graded without all deliverables below completed.
    a. Submit the link to the streaming video presentation.
    b. Submit the link to the remote repository.
    c. Confirm that the .exe file in the **Bin/Debug** folder functions and submit it.
3. Return to the Moodle assignment later to view your grade.

SPRINT 1 DELIVERABLES MARKING GUIDE AND CHECKLIST

Developer _____ **Reviewer(s)** _____

Deliverables Marking Guide

		Points	Score
Video	Video created and streamed	**5**	
GitHub	Most current version of the app is pushed to GitHub	**5**	
Executable File	Executable file is fully function.	**5**	
User Story Checklist	User story checklist completed and discussed in video	**5**	
Current Progress (Base on Acceptance Criteria)	• Significant (30 points) • Adequate (25 points) • Minimal (20 points)	**30**	
	Total Benchmark Points	**50**	

User Story Check List

Epic	User Story	Acceptance Criteria	Done
As a user, I need to see detail information for one product.	As a user, I will open the application, so that the detail of a product and company information is displayed in the window.	The window has an area that displays all product details.	
		The window has an area that displays the company information.	
	As a user, I will click the **Quit** button, so the application will close.	The window closes and the application exits when the **Quit** button is pressed.	

Notes and Comments (Include additions, deletions, modifications, and issues with the application)