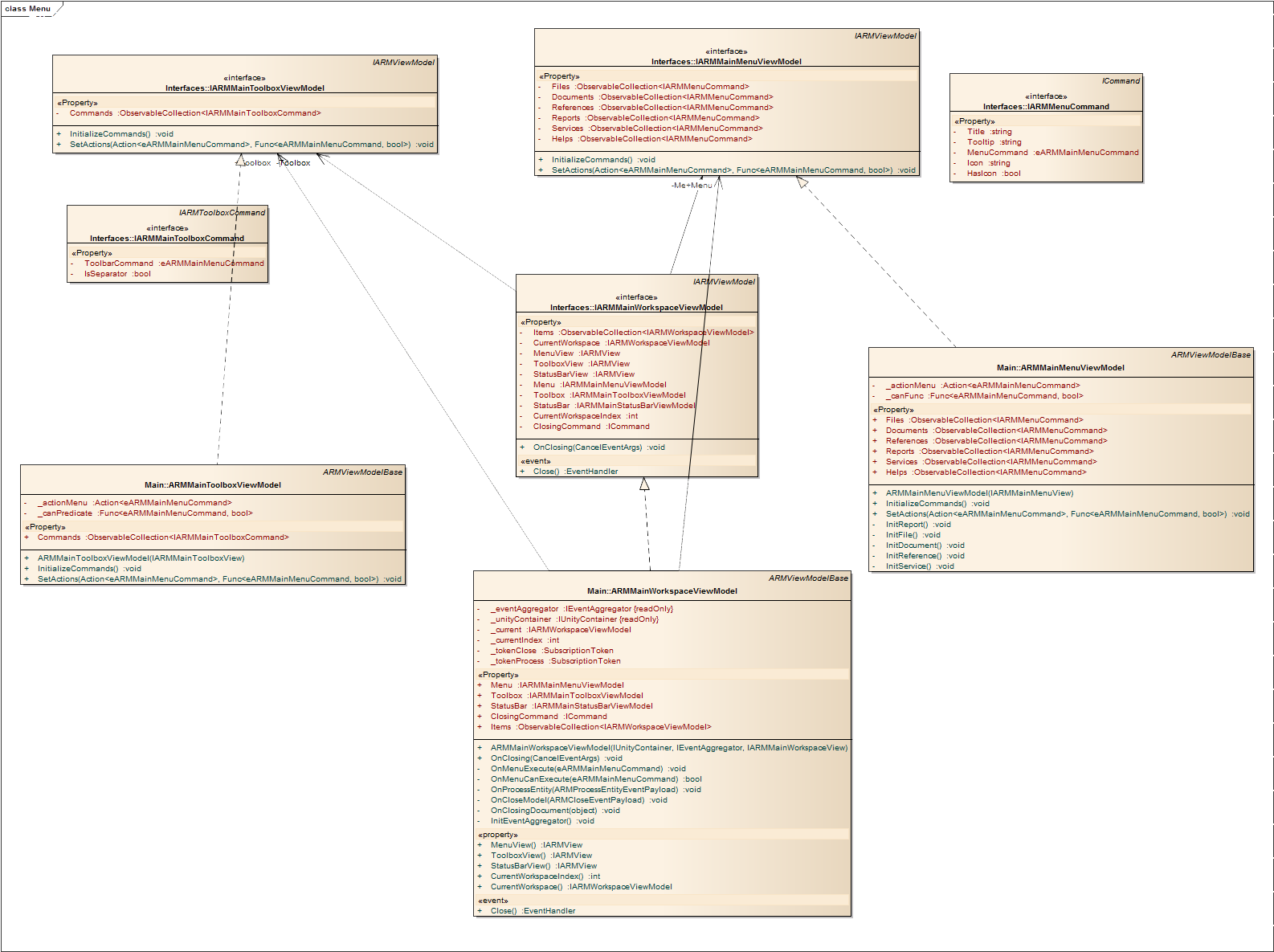
В нашому випадку ми обрали підхід, який пов’язаний не з обробкою повідомлень які надходять від головного меню та панелі управляння. А обрали підхід прив’язки команд до пунктів меню та кнопок панелі управління. У випадку команд обробка вводу користувача проходить на більш високому семантичному рівні, ніж нам це дозволяє роботи обробка подій. За допомогою команд ми здатні абстрагуватись від пристрою вводу. В даному випадку нам не важливо на якому елементів управління натиснув користувач, нам тільки важливо знати яку команду хоче виконати користувач. Тому команди ми можемо просто і легко прив’язати до таких елементів управління як пункт меню (MenuItem), кнопка (Button – властивість ICommand).

На малюнку нижче приведена діаграма класів яка відображає структуру меню та панелі управління:



Розкриємо об’єкти, які входять в дану діаграму.

Клас ARMMainMenuViewModel реалізовує інтерфейс IARMMainMenuViewModel. Даний клас є моделюю представлення головного меню програми. Даний клас містить декілька колекцій, які містять об’єкти типу IARMMenuCommand. Інтерфейс IARMMenuCommand представляє собою команду для меню. Нижче перераховані колекції команд:

1. ObservableCollection<IARMMenuCommand> Files { get; } – дана колекція містить команди, які відображаються в меню «Файл».
2. ObservableCollection<IARMMenuCommand> Documents { get; } – дана колекція містить команди, які відображаються в меню «Документи».
3. ObservableCollection<IARMMenuCommand> References { get; } – дана колекція містить команди, які відображаються в меню «Довідники».
4. ObservableCollection<IARMMenuCommand> Reports { get; } – дана колекція містить команди, які відображаються в меню «Звіти».
5. ObservableCollection<IARMMenuCommand> Services { get; } – дана колекція містить команди, які відображаються в меню «Сервіси»
6. ObservableCollection<IARMMenuCommand> Helps { get; } – дана колекція містить команди, які відображаються в меню «Допомога».

Кожна команда реалізовує базовий інтерфейс команд ICommand. Також команди наділені спеціальними властивостями, які дозволяються задавати картинки (іконки) – Icon, Image. Дані властивості повертають шлях до картинки. Title & Tooltip – дозволяють надати команді унікальне ім’я та підказку.

Клас ARMMainToolboxViewModel реалізовує IARMMainToolboxViewModel. Даний клас включає в себе функціональність головної панелі управління. Даний клас містить тільки одну колекцію команд:

1. ObservableCollection<IARMMainToolboxCommand> Commands { get; } – дана колекція містить визначений набір команд, що прив’язується до елемента управління «Toolbar».

Два вище згаданих інтерфейси входять в склад більш високоорганізованого класу ARMMainWorkspaceViewModel. Даний клас містить дві властивості, які відповідають головному меню програми “Menu” та головній панелі управління “Toolbox” програми. При створені головної моделі програми створюються моделі головного меню та головної панелі управління.

public ARMMainWorkspaceViewModel(IUnityContainer unityContainer, IEventAggregator eventAggregator,

IARMMainWorkspaceView workspaceView)

: base(workspaceView)

{

Items = new ObservableCollection<IARMWorkspaceViewModel>();

\_unityContainer = unityContainer;

\_eventAggregator = eventAggregator;

**Menu = \_unityContainer.Resolve<IARMMainMenuViewModel>();**

**Toolbox = \_unityContainer.Resolve<IARMMainToolboxViewModel>();**

StatusBar = \_unityContainer.Resolve<IARMMainStatusBarViewModel>();

**Toolbox.SetActions(OnMenuExecute,OnMenuCanExecute);**

**Menu.SetActions(OnMenuExecute, OnMenuCanExecute);**

**Menu.InitializeCommands();**

**Toolbox.InitializeCommands();**

InitEventAggregator();

ClosingCommand = new DelegateCommand<object>(OnClosingDocument, o => true);

ARMSystemFacade.Instance.Logger.LogInfo("Application Started....");

AppDomain.CurrentDomain.FirstChanceException += CurrentDomainOnFirstChanceException;

}

На даному етапі створюються екземпляр меню рядком:

**Menu = \_unityContainer.Resolve<IARMMainMenuViewModel>();**

та екземпляр моделі панелі управління:

**Toolbox = \_unityContainer.Resolve<IARMMainToolboxViewModel>();**

Після чого в кожен екземпляр встановлюються обробники натискання на пункти меню та на кнопки панелі управління.

**Toolbox.SetActions(OnMenuExecute,OnMenuCanExecute);**

**Menu.SetActions(OnMenuExecute, OnMenuCanExecute);**

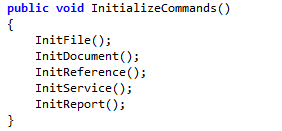
Дані обробники розкриються нижче. Після чого проходить ініціалізація команд:

**Menu.InitializeCommands();**

**Toolbox.InitializeCommands();**

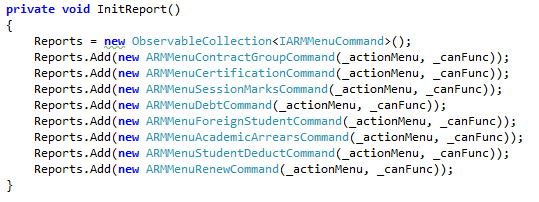
В даних методах проходить ініціалізація та формування команд меню та панелі управління.

Метод **InitializeCommands**  класу ARMMainMenuViewModel:



В даному методі проходить почергова ініціалізація та формування колекцій команди для пунктів «Файл», «Документи», «Довідники», «Сервіси», «Звіти».

Для прикладу наведемо код ініціалізації команд пункту «Звіти»:



Як бачимо з вище приведеного коду в першу чергу створюється колекція команд. Потім почергово додаються команди для запуску відповідних звітів. Кожна команди містить свій заголовок, підказку та, при можливості, іконку. Також кожній команді передається обробник. Даний обробник був встановлений раніше за допомогою методу SetActions(OnMenuExecute,OnMenuCanExecute). Таким чином формуються списки команд для головного меню програми та панелі управління.

Після створення екземплярів та формування списків команд форми меню та панелі управляння вставляються у чітко визначені місця на головній формі програми.

Під час відображення головного інтерфейсу списки команди прив’язуються до відповідних пунктів меню та кнопок на панелі управління.

Пункти меню створюються за допомогою слідуючих шаблонів:

<DataTemplate x:Key="MenuIconItemTemplate">

            <Grid x:Name="Root">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition Width="16"></ColumnDefinition>

                    <ColumnDefinition Width="Auto"></ColumnDefinition>

                </Grid.ColumnDefinitions>

                <Grid.ToolTip>

                    <ToolTip>

                        <TextBlock Text="{Binding Tooltip}"></TextBlock>

                    </ToolTip>

                </Grid.ToolTip>

                <Image Grid.Column="0" Source="{Binding Icon}" Width="16" Height="16" Margin="-45 0 0 0"></Image>

                <TextBlock Grid.Column="1" Text="{Binding Title}"></TextBlock>

            </Grid>

        </DataTemplate>

        <DataTemplate x:Key="MenuItemTemplate">

            <Grid x:Name="Root">

                <Grid.ColumnDefinitions>

                    <ColumnDefinition Width="16"></ColumnDefinition>

                    <ColumnDefinition Width="Auto"></ColumnDefinition>

                </Grid.ColumnDefinitions>

                <Grid.ToolTip>

                    <ToolTip>

                        <TextBlock Text="{Binding Tooltip}"></TextBlock>

                    </ToolTip>

                </Grid.ToolTip>

                <Image Grid.Column="0"></Image>

                <TextBlock Grid.Column="1" Text="{Binding Title}"></TextBlock>

            </Grid>

        </DataTemplate>

Команди з колекції через механізм «Binding» приєднується до елемента під назвою «Root» з шаблона в його властивість DataContext.

Кнопки панелі управління створюються за допомогою слідуючого шаблону:

<DataTemplate x:Key="ButtonToolboxTemplate">

            <Grid Width="32" Height="32" Margin="2">

                <Button Command="{Binding}">

                    <Button.ToolTip>

                        <ToolTip>

                            <TextBlock Text="{Binding Tooltip}"></TextBlock>

                        </ToolTip>

                    </Button.ToolTip>

                    <Image Source="{Binding Image}"></Image>

                </Button>

            </Grid>

 </DataTemplate>

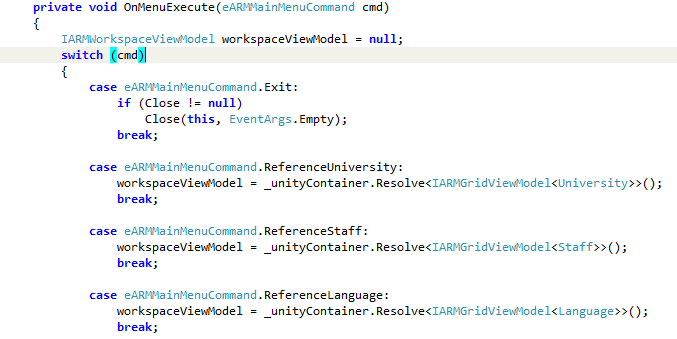
З даного шаблону ми бачимо як за допомогою механізму «Binding» команда приєднується до кнопки (до властивості «Command»). Тому при натисканні користувачем на кнопку, буде виконуватись команда.

Пункти меню та кнопки працюють таким чином, що вони підтримують прив’язку до об’єктів, що реалізовують інтерфейс ICommand. Тобто у випадку натискання на пункт меню або на кнопку виконується метод ICommand.Execute. Для перевірки доступності відповідного пункту меню або кнопки виконується метод ICommand.CanExecute.

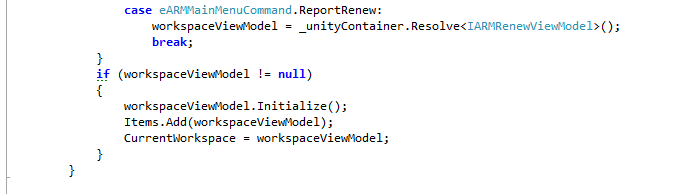
Опишемо послідовність роботи на прикладі меню.

При кліку на певному пункті меню виконується метод ICommand.Execute, який в свою чергу викликає обробник, який ми передали через конструктор команди. Який в свою чергу був переданий в модель головного меню з екземпляру класу ARMMainWorkspaceViewModel. Тому в нашому випадку програма виконує метод OnMenuExecute.

Код метода OnMenuExecute:



………….



Як бачимо в даний метод приходить значення перечислення eARMMainMenuCommand. Дане унікальне значення присвоєно кожній команді. Тому в даному випадку програма знає, який об’єкт їй потрібно створювати. Тому за допомогою оператора вибору створюється відповідний екземпляр відповідного класу і виконується код ініціалізації для створеної моделі.

Таким чином ми бачимо що обробка всіх кліків на пунктах меню та кнопках на панелі управління відбувається в одному місці. Це досить зручно, тому що правки потрібно проводити тільки в одному місці програми і вони автоматично діють на роботу меню та панелі управління. Так само з методом OnMenuCanExecute, який призначений для визначення чи доступний в певний момент часу пункт меню або кнопка панелі управління. Ми можемо одним методом вплавити на доступність як пунктів меню так і кнопок на панелі управляння.