Упражнение 3

Създаване на **Windows Phone** приложение, основано на модела **Model-View-ViewModel (MVVM).**

Целта на приложението е да илюстрира:

- създаването на приложение, имплементиращо модела MVVM;
- използването на механизма, известен като *XAML binding*;
- възможността за съхраняване на данни в паметта за **settings** на мобилния телефон (емулатора).

Моделът **MVVM** позволява да се отдели създаването на модел на данните от проектирането на потребителския интерфейс (изгледа на данните) на приложението. Моделът данни (Model) представлява **клас на C#,** докато потребителският интерфейс (изгледа на данните, View) е **XAML user control**. ViewModel е връзката между модела данни (Model) и изгледа на данните (View) и също представлява C# клас, който предоставя колекция от обекти от класа на модела данни и два метода: за зареждане на колекцията и за запазване на колекцията в settings на телефона или емулатора.

Settings представлява възможност за съхранение в паметта на мобилния телефон (**IsolatedStorageSettings**) на двойки **key-value**, като не се допуска дублиране по ключ (key). Използва се класът **IsolatedStorageSettings,** неговите свойства **ApplicationSettings, Count, Value** и неговите методи **Add(), Save(), Contains().**

Задача. Да се създаде **Windows Phone** приложение, използващо модела MVVM, като бъдат създадени: отделен модел на данните (Model), отделен изглед към данните(View), както и клас (ModelView), отговарящ за връзката между данните и изгледа на данните. За привързването на изгледа на данните към самите данни да се използва **XAML binding**. Допълнително, данните на приложението да бъдат съхранени в isolated storage.

За създаване на приложението трябва да се изпълнят следните стъпки.

1. Създайте нов проект на **Visual C# (Silverlight)**, като използвате шаблона **Windows Phone Application** (изберете Windows Phone OS 7.1). Задайте име за проекта **MyMVVMproject**.

2. В режим на проектиране (**MainPage.xaml**) създайте 3 нови папки с имена **DataModel, DataView, ModelView** (за да създадете всяка от трите папки, в прозореца **Solution Explorer** от контекстното меню за проекта изберете команда **Add** и след това команда **New Folder**, като въведете предложените по-горе имена на папките).

3. Към вече създадената папка с име **DataModel** добавете клас (за да добавите нов клас към папка, от контекстното меню за папката изберете команда **Add** и след това изберете **Class**). За име на новия клас посочете **MyData**. Новият клас **MyData** се добавя към проекта в папката с име **DataModel** и се отваря в редактора на кода.

4. В редактора на кода (**code view - MyData.cs**) дефинирайте новия клас **MyData**, който представлява модела на данните за конкретното приложение. Започнете с добавяне на пространство на имената **System.ComponentModel**. Класът **MyData** трябва да наследява интерфейса **INotifyPropertyChanged**. Когато даден клас наследява интерфейс **INotifyPropertyChanged,** класът има възможност да уведомява изгледите на данните за всяка промяна в стойността на свойство на класа; на свой ред изгледът на данните актуализира потребителския интерфейс, за да отрази промяната в стойността на свойството. Дефиницията на класа **MyData** включва две **private** полета данни (име на студент **name** и възраст **age**) и съответните **public** свойства за тях: **Name** и **Age**. Класът **MyData** предлага също собствено събитие с име **PropertyChanged**, както и метод за „запалване” на това събитие с име **Fire_PropertyChanged**. В дефиницията на класа, собственото събитие **PropertyChanged** трябва да се дефинира като се използва делегат **PropertyChangedEventHandler (**вграден делегат в средата за програмиране) и

ключова дума **event**. Съгласно правилата класът, дефиниращ собствено събитие трябва също да дефинира метод за «запалване» на това събитие. Ето защо в дефиницията на класа се включва метода, наречен **Fire_PropertyChanged**, който проверява дали има делегат, чакащ на опашка (свързан списък от делегати) за обслужване на събитието и ако да – да подава управлението на първия делегат от списъка делегати. Методът **Fire_PropertyChanged** се извиква явно в кода на двете свойства **Name** и **Age** на класа **MyData**.

Класът **MyData** има следния общ вид:

```
using System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.ComponentModel;

namespace MyMVVMproject.DataModel
{
    public class MyData : INotifyPropertyChanged
    {
        private int age;
        public int Age
        {
            get
            {
                return age;
            }
            set
            {
                age = value;
                Fire_PropertyChanged("Age");
            }
        }
        private string name;
        public string Name
        {
            get
            {
                return name;
            }
            set
            {
                name = value;
                Fire_PropertyChanged("Name");
            }
        }
        public event PropertyChangedEventHandler PropertyChanged;
        private void Fire_PropertyChanged(string property)
        {
            if (this.PropertyChanged != null)
            {
                this.PropertyChanged(this, new PropertyChangedEventArgs(property));
```

```
            }
        }
        public MyData GetCopy()
        {
            MyData copy = (MyData)this.MemberwiseClone();
            return copy;
        }
    }
}
```

5. Към вече създадената папка с име **ModelView** добавете клас**.** За име на новия клас посочете **ConnectToMyData**. Новият клас **ConnectToMyData** се добавя към проекта в папката с име **ModelView** и се отваря в редактора на кода.

6. В редактора на кода (**ConnectToMyData.cs**) дефинирайте новия клас **ConnectToMyData**, който реализира връзката между модела на данните и изгледа на данните. Преди всичко добавете следните пространства на имената:

```
using System.IO.IsolatedStorage;
using System.Collections.ObjectModel;
using MyMVVMproject.DataModel;
```

Класът **ConnectToMyData** включва колекция от елементи от класа на модела на данните **MyData**. Колекцията е наречена **List** и е обект от класа **ObservableCollection.** Това позволява на класа **ConnectToMyData** да уведомява своите изгледи при всяка промяна в елементите на колекцията; на базата на такава промяна изгледът на данните обновява потребителския интерфейс.
В класа **ConnectToMyData** дефинирайте метод **GetList()**, в който включете оператори на езика C# за попълване на колекцията **List** с обекти от класа **MyData**. В метода **GetList()** първо проверете дали в isolated storage има съхранени данни. Ако да – колекцията **List** трябва да се попълни именно от isolated storage. Ако няма запазени обекти от класа **MyData** в isolated storage, колекцията **List** се попълва от обекти от класа **MyData**, създадени и инициализирани в кода на метода.
В класа **ConnectToMyData** дефинирайте и метод **SaveList()**, който съдържа оператори на C# за съхранение на елементите от колекцията **List** (обекти от клас **MyData**) в isolated storage.
Класът **ConnectToMyData** има следния общ вид:

```
sing System;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Ink;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using System.IO.IsolatedStorage;
using System.Collections.ObjectModel;
using MyMVVMproject.DataModel;

namespace MyMVVMproject.ModelView
{
```

```
    public class ConnectToMyData
    {
        public ObservableCollection<MyData> List
        {
            get;
            set;
        }
        public void GetList()
        {
            if (IsolatedStorageSettings.ApplicationSettings.Count > 0)
            {
                ObservableCollection<MyData> a = new ObservableCollection<MyData>();
                foreach (Object o in
IsolatedStorageSettings.ApplicationSettings.Values)
                {
                    a.Add((MyData)o);
                }
                List = a;
                MessageBox.Show("Info - objects are loaded from the isolated storage
settings");
            }
            else
            {
                ObservableCollection<MyData> a = new ObservableCollection<MyData>();
                a.Add(new MyData() { Name = "Ivan", Age = 25 });
                a.Add(new MyData() { Name = "Aneta", Age = 45 });
                a.Add(new MyData() { Name = "Suzana", Age = 20 });
                List = a;
                MessageBox.Show("Info is loaded from default values");
            }
        }
        public void SaveList()
        {
            IsolatedStorageSettings sett =
IsolatedStorageSettings.ApplicationSettings;
            foreach (MyData x in List)
            {
                if (sett.Contains(x.Name))
                {
                    sett[x.Name] = x;
                }
                else
                {
                    sett.Add(x.Name, x.GetCopy());
                }
            }
            sett.Save();
            MessageBox.Show("All data is saved into the dictionary");
        }
    }
}
```

7. Към вече създадената папка **DataView** добавете изглед за данните с име **MyView**, който представлява потребителски контрол състоящ се от един контрол **ListBox**, привързан към данните. За да създадете потребителски контрол изберете команда **Add** от контекстното меню за папката **DataView** и след това изберете **New Item**; използвайте

шаблона **Windows Phone User Control;** задайте име на потребитлския контрол **MyView**.
Потребителският контрол **MyView** се появявя в режим на проектиране (**design view**). От **ToolBox** привлечете контрол **ListBox**. За контрола **ListBox** установете направо в XAML кода свойството **ItemsSource="{Binding}”.** Създайте две колони за контрола **ListBox**. Свържете първата колона с контрол **TextBox**, за който установете свойство **Text="{Binding Path=Name, Mode=TwoWay}";** по този начин контролът **TextBox** ще бъде свързан с данните от свойство **Name** за обекти от класа **MyData**; връзката с данните е двупосочна. Свържете втората колона с друг контрол **TextBox**, за който установете свойство **Text="{Binding Path=Age, Mode=TwoWay}".** По този начин контролът **TextBox** ще бъде свързан с данните от свойство **Age** за обекти от класа **MyData**; връзката с данните отново е двупосочна.
XAML кодът за потребителския контрол **MyView** изглежда по следния начин (допълненията в кода са подчертани):

```
<UserControl x:Class="MyMVVMproject.DataView.MyView"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    d:DesignHeight="480" d:DesignWidth="480">

    <Grid x:Name="LayoutRoot">
        <ListBox Height="275" HorizontalAlignment="Left" Margin="51,47,0,0"
Name="listBox1" VerticalAlignment="Top" Width="296" ItemsSource="{Binding}">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <Grid>
                        <Grid.ColumnDefinitions>
                            <ColumnDefinition Width="200"/>
                            <ColumnDefinition Width="100"/>
                        </Grid.ColumnDefinitions>
                        <TextBox x:Name="Name" Text="{Binding Path=Name, Mode=TwoWay}"
Grid.Column="0" HorizontalAlignment="Left" VerticalAlignment="Bottom"/>
                        <TextBox x:Name="Age" Text="{Binding Path=Age, Mode=TwoWay}"
Grid.Column="1" HorizontalAlignment="Center" VerticalAlignment="Bottom"/>
                    </Grid>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>
    </Grid>
</UserControl>
```

8. Допълнете главната страница на приложението - **MainPage.xaml**. Главната страница на приложението трябва да съдържа създадения в предишната стъпка изглед на данните **MyView**. Трябва да се добави пространството на имената, съдържащо изгледа на данните:
`xmlns:views="clr-namespace:MyMVVMproject.DataView".`

След това трябва да се добави код за посочване на изгледа, съответстващ на създадения по-горе потребителски контрол за изгледа на данните:
`<views:MyView x:Name="MyViewOnPage" Height="400"/>.`

XAML кодът на главната страница на приложението - **MainPage.xaml** - с внесените допълнения трябва да изглежда по следния начин (корекциите и допълненията в кода са подчертани):

```
<phone:PhoneApplicationPage
    x:Class="MyMVVMproject.MainPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    mc:Ignorable="d" d:DesignWidth="480" d:DesignHeight="768"
    xmlns:views="clr-namespace:MyMVVMproject.DataView"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    shell:SystemTray.IsVisible="True">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Exercise 4"
Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Persons data" Margin="9,-7,0,0"
Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <views:MyView x:Name="MyViewOnPage" Height="400"/>
        </Grid>
    </Grid>
</phone:PhoneApplicationPage>
```

9. Към **MainPage.xaml.cs** в редактора на кода (**code view**) добавете оператори на езика C#. С тези оператори на C# създайте обект от класа **ConnectToMyData** (класът **ConnectToMyData** беше дефиниран в папката **ModelView** и представя връзката между модела данни и изгледа данни; класът **ConnectToMyData** съдържа колекцията **List** от обекти от класа **MyData**). За целта първо добавете пространство на имената:
`using MyMVVMproject.ModelView;`

После дефинирайте **private** поле данни от типа **ConnectToMyData.** След това, в конструктора на главната страница на приложението (**MainPage),** след метода **InitializeComponent(),** създайте обект от класа **ConnectToMyData**.

Override метода **OnNavigateTo()** и в този метод извикайте метода **GetList()** за обекта от класа **ConnectToMyData** и установете свойството **DataContext** на потребителския контрол.
Кодът на C# на **MainPage.xaml.cs** трябва да изглежда по следния начин:

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Net;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Documents;
using System.Windows.Input;
using System.Windows.Media;
using System.Windows.Media.Animation;
using System.Windows.Shapes;
using Microsoft.Phone.Controls;
using MyMVVMproject.ModelView;

namespace MyMVVMproject
{
    public partial class MainPage : PhoneApplicationPage
    {
        //деклариране на поле данни от тип ConnectToMyData
        private ConnectToMyData mv;

        public MainPage()
        {
            InitializeComponent();
            //създаване на обект от класа ConnectToMyData
            mv = new ConnectToMyData();
        }

        protected override void
OnNavigatedTo(System.Windows.Navigation.NavigationEventArgs e)
        {
            base.OnNavigatedTo(e);
            mv.GetList();
            MyViewOnPage.DataContext = mv.List;
        }
    }
}
```

10. Добавете към **MainPage.xaml** един контрол **Button** като го привлачите от прозореца **ToolBox**. За бутона установете атрибут **Content = “SAVE”** и атрибут **Name = “save”.** Създайте метод за обработка на събитието **Click** за бутона **SAVE**. В тялото на метода за обработка на събитието **Click** за бутона **SAVE** включете само един оператор на езика C# за извикване на метода **SaveList()** на класа **ConnectToMyData** (запазване съдържанието на колекцията **List** от обекти от класа **MyData** в **settings** на телефона или емулатора). Методът за обработка на събитието **Click** за бутона **SAVE** изглежда по следния начин:

```
private void save_Click(object sender, RoutedEventArgs e)
{
      mv.SaveList();
}
```

11. Построете и стартирайте приложението.