Упражнение 2

Създаване на **Windows Phone** приложение, което съхранява въведена от виртуалната клавиатура на телефона информация в текстов файл в паметта на приложението (*Isolated storage*) и използва бутони-икони в **ApplicationBar** за поддържане на директории.

Целта на приложението е да илюстрира:

- получаването на референция към паметта на приложението, предназначена за работа с текстови файлове;
- създаването на директория (папка);
- създаването на нов текстов файл;
- записването на ново съдържание или прочитане на съдържанието на текстов файл;
- използването на бутони-икони в **ApplicationBar** на **Windows Phone** приложение.

Isolated storage на **Windows Phone** приложение се поддържа от набор класове, които са дефинирани в две пространства на имената:

- **System.IO**
- **System.IO.IsolatedStorage**

Класът **IsolatedStorageFile** представя област от *Isolated storage*, която може да съдържа директории и текстови файлове.
Класът **IsolatedStorageFileStream** представлява файл в *Isolated storage*.
Класовете **StreamWriter** и **StreamReader** имплементират съответно класовете **TextWriter,** предназначен за извеждане на символи в поток и **TextReader** за четене на поток от символи.

ApplicationBar е лента в долния край на страницата на **Windows Phone** приложение и по подразбиране съдържа т. нар. бутони-икони с ограждащи ги кръгчета (icon buttons) и текстови меню-елементи (menu items). Бутоните-икони включват изображение (.png файл) и текст (кратък надпис). Меню-елементите се представят само с кратък текст и могат да бъдат разрешени/забранени чрез установяване на свойство **IsMenuEnabled**.

ApplicationBar има свойства (**Mode**, **Opacity, IsVisible** и др.) които могат да се установят еднократно при създаването му или да се променят динамично по време на изпълнение.

Има два начина да добавим **ApplicationBar** към **Windows Phone** приложение. По-лесният начин е като използваме **XAML**, още повече, че предлаганият от **Windows Phone SDK** шаблон на приложение включва по подразбиране коментиран код на **XAML** за **ApplicationBar** и единственото, което е нужно е да се изтрие коментара и да се установят свойствата на бутоните-икони и/или на меню-елементите. Другата възможност е да се създаде обект от клас **ApplicationBar** в програмен код на C#.

Задача 1. Да се създаде **Windows Phone** приложение, което има само една страница. В главната (и единствена) страница на приложението да се добави контрол **TextBox**, контрол **TextBlock** и два контрола **Button**. Контролът **TextBox** да се използва за въвеждане на текст от виртуалната клавиатура, който да се записва в новосъздаден файл, разположен в новосъздадена директория. Контролът **TextBlock** да се използва за извеждане на

съдържанието на предварително създадения текстов файл. Операцията създаване на нов файл и запис в него да се инициира с единия от двата контрола **Button**, а операцията отваряне на съществуващ файл и прочитане на съдържанието му – с другия контрол **Button**.
Да се добави към приложението **ApplicationBar,** включващ само два бутона-икони с подходящи изображения, като единият от тях да позволява създаване на директория, а другият - изтриване на директория.

За създаване на приложението трябва да се изпълнят следните стъпки.

1. Създайте нов проект на **Visual C# (Silverlight)**, като използвате шаблона **Windows Phone Application.**

2. В режим на проектиране (**MainPage.xaml**) променете името на приложението на **Exercise 2** и името на страницата на **FILE I/O**.

3. В режим на изпълнение (**MainPage.xaml.cs**) добаведе следните две пространства на имената, които имат връзка с файловия вход/изход:

```
using System.IO;
using System.IO.IsolatedStorage;
```

4. В режим на проектиране добавете (чрез влачене и пускане от прозореца **ToolBox**) един контрол **TextBox,** един контрол **TextBlock** и два контрола **Button**. За първия контрол **Button**

сменете свойство **Content = “CreateFile”** и свойство **Name = “createFile”.** За втория контрол **Button** сменете свойство **Content = “ReadFile”** и свойство **Name = “readFile”.**

5. В кода на C#, в класа на главната форма **MainPage** добавете следните три полета данни:

```
private IsolatedStorageFileStream myFile;
private StreamWriter s;
private  StreamReader r;
```

6. Създайте метод за обработка на събитието **Click** за бутон **CreateFile**. В метода за обработка на събитието **Click** за бутона **CreateFile** включете оператор на C# за получаване на референция към паметта за приложението, предназначена за работа с текстови файлове – клас **IsolatedStorageFile**. Добавете оператор на C# за създаване на нова директория, ако такава вече не съществува, като използвате съответните методи на обекта от клас **IsolatedStorageFile.** Добавете оператор на C# за отваряне на нов текстов файл (файлов поток), като създадете обект от клас **IsolatedStorageFileStream** и подадете на конструктора: име на файла (файлова спецификация), режим на създаване на файла (създаване на нов файл) и референцията от клас **IsolatedStorageFile**. Добавете оператор на C# за създаване на обект от клас **StreamWriter**. За да запишете във файла текста, въведен от виртуалната клавиатура, извикайте метода **WriteLine** на класа **StreamWriter**. За създаване на файла и на обекта от клас **StreamWriter** използвайте оператор **using**, за да не се налага да затваряте изрично файла и „райтера“. Методът за обслужване на събитието **Click** за бутона **CreateFile** има следния общ вид:

```
private void createFile_Click(object sender, RoutedEventArgs e){

// получаване на референция към паметта за текстови файлове
IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();

//създаване на нова директрия myFolder, ако такава вече не съществува
if(!(myStore.DirectoryExists("myFolder")))
myStore.CreateDirectory("myFolder");

//създаване на нов файл и отварянето му за запис
using (myFile = new IsolatedStorageFileStream("myFolder\\textA.txt",
FileMode.OpenOrCreate, myStore))
{
    //създаване на обект от клас StreamWriter, необходим за запис във файла
     using (s = new StreamWriter(myFile))
     {
         s.WriteLine(textBox1.Text);
     }
}
}
```

7. Създайте метод за обработка на събитието **Click** за бутон **ReadFile**. В метода за обработка на събитието **Click** за бутона **ReadFile** включете оператор на C# за получаване на референция към паметта на приложението, предназначена за работа с текстови файлове – клас **IsolatedStorageFile**. Добавете оператор на C# за отваряне на съществуващ текстов файл (файлов поток), като създадете обект от клас **IsolatedStorageFileStream** и подадете на конструктора: име на файла (файлова спецификация), режим на създаване на файла (отваряне на съществуващ файл) и референцията от тип **IsolatedStorageFile**. Включете оператор на C# за създаване на обект от клас **StreamReader**. За да прочетете съдържанието

на файла, извикайте метод **ReadLine** на класа **StreamReader** в цикъл **while** до достигане на края на файла. Тъй като операциите, свързани с файлове могат да дадат грешка и да изгубите възможността да затворите файла, то трябва да използвате **try-catch-finally** блок и да затворите файла в блока **finally (**алтернатива на използването на **using)**. Методът за обслужване на събитието **Click** за бутона **ReadFile** има следния общ вид:

```
private void readFile_Click(object sender, RoutedEventArgs e)  {

string str = "";
//получаване на виртуална памет за приложението
IsolatedStorageFile myStore = IsolatedStorageFile.GetUserStoreForApplication();
try
{//отваряне на съществуващ файл за четене
   myFile = new IsolatedStorageFileStream("myFolder\\textA.txt", FileMode.Open,
myStore);
       try
       {
          //създаване на обект от клас StreamReader за четене от файла
           r = new StreamReader(myFile);
          //четене на съдържанието на файла и извеждане в контрола textBlock1
          while (!r.EndOfStream)
          {
              str += r.ReadLine();
          }
          textBlock1.Text = str;
          r.Close();
          myFile.Close();//затваряне на файла при успешна операция

      }
      catch (Exception ex)
      {
          MessageBox.Show(ex.Message);
      }
      finally
      {
          r.Close();
      }
 }
 catch (Exception ex)
 {
     MessageBox.Show(ex.Message);
 }
 finally
 {
     if(myFile != null) myFile.Close();//затваряне на файла при грешка
 }
}
```

8. Построете и изпълнете приложението.
Въведете текст в контрола **TextBox**, натиснете бутона **CreateFile** за да запишете въведения текст в новосъздаден файл. Натиснете бутона **ReadFile** за да прочетете съдържанието на файла и да го изведете в контрола **TextBlock**.
Забележка. Доколкото в резултат на изпълнението на приложението в паметта на приложението се създава текстов файл, при напускане и повторно връщане към страницата

на приложението може направо да се отвори файла и да се прочете или покаже неговото съдържание или може да се промени съдържанието на файла.

9. Премахнете коментара от групата оператори на XAML свързани с елемента **PhoneApplicationPage.ApplicationBar**, за да добавите към приложението **ApplicationBar**. Изтрийте редовете, свързани с елемента **ApplicationBar.MenuItems**, тъй като приложението няма да използва меню-елементи, а само два бутона-икони.

10. Редактирайте XAML кода на **ApplicationBar** за двата бутона-икони. За всеки от двата бутона-икони установете свойство **IconUri** така че да сочи избрани от Вас .png файлове с изображения.
Забележка. Преди да се използват изображения (.png файлове) като икони на бутоните от **ApplicationBar**, тези изображения трябва да бъдат добавени към приложението за да станат част от неговия XAP файл; за целта се използва вече описаната в Упражнение 1 процедура за добавяне към приложението на .jpg файловете.

11. Установете свойство **Text** на разположения вляво бутон от **ApplicationBar** като: **Text = “Create Folder”,** а на разположения вдясно бутон като: **Text = “Delete Folder”.**

12. За всеки от двата бутона-икони добавете метод за обработка на събитието **Click**. Редактираният код на XAML за елемента **ApplicationBar** включващ два бутона-икони трябва да изглежда по следния начин:

```
<!--Sample code showing usage of ApplicationBar-->
    <phone:PhoneApplicationPage.ApplicationBar>
        <shell:ApplicationBar IsVisible="True" IsMenuEnabled="True">
            <shell:ApplicationBarIconButton IconUri="34437-Files.png"
Text="Create Folder" Click="ApplicationBarIconButton_Click"/>
            <shell:ApplicationBarIconButton IconUri="pic.png" Text="Delete
Folder" Click="ApplicationBarIconButton_Click_1"/>
        </shell:ApplicationBar>
    </phone:PhoneApplicationPage.ApplicationBar>
```

13. В метода за обработка на събитието **Click** за бутона-икона **“Create Folder”** включете оператор на C# за получаване на референция към паметта на приложението, предназначена за работа с текстови файлове – клас **IsolatedStorageFile**. Добавете оператор на C# за създаване на директория с посочено име, ако такава директория не съществува. Методът за обслужване на събитието **Click** за бутона-икона **“Create Folder”** има следния общ вид:

```
private void ApplicationBarIconButton_Click(object sender, EventArgs e)
{
   IsolatedStorageFile myStore=IsolatedStorageFile.GetUserStoreForApplication();
   if (!myStore.DirectoryExists("myFolder"))
           myStore.CreateDirectory("myFolder");
}
```

14. В метода за обработка на събитието **Click** за бутона-икона **“Delete Folder”** включете оператор на C# за получаване на референция към паметта на приложението, предназначена за работа с текстови файлове – клас **IsolatedStorageFile**. Добавете оператор на C# за изтриване на директория с посочено име, ако тя съществува. Обработете случая, когато

директорията съществува, но съдържа файлове и не може да бъде изтрита. Методът за обслужване на събитието **Click** за бутона-икона **“Delete Folder”** има следния общ вид:

```
private void ApplicationBarIconButton_Click_1(object sender, EventArgs e)
{
    IsolatedStorageFile myStore=IsolatedStorageFile.GetUserStoreForApplication();
    try
    {
        if (myStore.DirectoryExists("myFolder"))
                myStore.DeleteDirectory("myFolder");
    }
    catch(Exception ex)
    {
       MessageBox.Show("Folder is not empty. Delete files from the folder and try
again.");
    }
}
```

15. Построете и изпълнете приложението.