

Упражнение №4 по ПС - Част 2

XAML

System.Xaml
Страници
Панели
Списъчни контроли
Ресурси
Стилове

Целта на това упражнение:

Запознаване с основните принципи при организация на GUI със XAML, които осигуряват неговата гъвкавост.

Необходими познания:

От предметите дотук и предното упражнение трябва да можете:

- Да може да сглобите основно визуално приложение
- Да може да опишете основен интерфейс със XAML

Легенда:

Най-важното

- Схематично и накратко

Обяснение. Уточнение.

Пример: или **Задача:** оградено трябва да изпълните за да е синхронизиран проекта ви

- Подходящо място да стартирате изпълнение за да проверите дали работи вярно.

Задачите в упражнението изграждат:

Приложение за следене на разходи

В това упражнение: Създаваме ново упражнително приложение за разходи

- създаваме един прозорец със списък на всички хора с разходи и бутон за преминаване към друга страница
- създаваме втора страница с интерфейс за извеждане на разходи
(данните за разходите не са описани)
- добавяме стилове и ги прилагаме за използваните контроли

В края на упражнението:

Потребителят вижда прозорец със списък на всички хора с разходи и бутон за преминаване към друга страница, в която има интерфейс за извеждане на разходите на избрания човек.

1. WPF приложение с навигация и страници

Със следващите стъпки ще изградим едно по-сложно приложение на WPF, което ползва навигация и страници. Ще разположим елементите в него с помощта на Grid. Ще направим малко по-съвременен дизайн и ще предаваме данни от една страница в друга.

1. Към вече направения Solution, добавете нов проект (натиснете с десен бутон върху името на Solution-а и изберете Add ▢ New Project...) и го кръстете **ExpenseIt**.
2. Отворете App.xaml. Този XAML файл дефинира WPF приложението и бъдещи негови ресурси. Ще използваме този файл по-късно, за да уточним кой UI ще бъде автоматично зареждан при стартиране на програмата; в нашия случай това е MainWindow.xaml.
3. Отворете MainWindow.xaml. Този XAML файл е главния прозорец на вашето приложение и ще изобразява съдържанието в страници. Класът Window дефинира свойствата на прозорец, а именно – заглавие, големина, икона, обработва събития като затваряне или скриване.
4. Изтрийте Grid елемента в MainWindow.xaml.
5. Променете типа от Window на NavigationWindow.

Правим тази смяна защото приложението ще навигира между различно съдържание в зависимост от действията на потребителя. NavigationWindow наследява всички свойства на Window, но в същото време предоставя възможност за навигация подобно на тази в уеб браузър.

6. Променете следните свойства на NavigationWindow елемента:
 - Променете свойството Title на "ExpenseIt".
 - Променете свойството Width на 500 pixels.
 - Променете свойството Height на 350 pixels.

Кодът ви трябва да изглежда така:

```
<NavigationWindow x:Class="ExpenseIt.MainWindow"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    Title="ExpenseIt" Height="350" Width="500">

</NavigationWindow>
```

7. Отворете MainWindow.xaml.cs. Това е файла съдържащ code-behind, обработващ събитията декларирани в MainWindow.xaml. Този файл съдържа частичен клас за прозореца дефиниран в XAML.
8. Променете MainWindow класа да наследява **NavigationWindow**, а не **Window**.
9. Направете приложението основно за Solution-а. С десен бутон върху него, от контекстното меню изберете **Set as StartUp project**.
10. Компилирайте и стартирайте приложението като натиснете F5 или изберете Start Debugging от менюто Debug.

1.1. Добавяне на ресурсни файлове към приложението

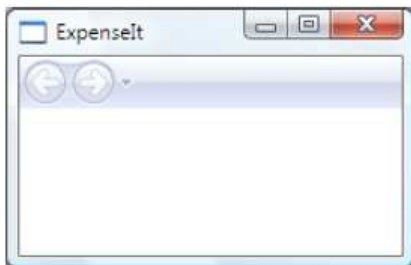
В тази част от упражнението ще добавим две страници и изображение към приложението.

1. Добавете нова страница Page към проекта (с десен бутон натиснете върху името на проекта и от контекстното меню изберете Add → New Item → Page). Дайте ѝ име **ExpenseltHome.xaml**

Това ще е началната страница, която ще се показва при стартиране на приложението. Тя ще показва списък от хора, от които потребителя ще може да избере един, за да се покаже справката за разходите му.

2. В отворения се **ExpenseltHome.xaml** настройте заглавието **Title** на "**Expenselt - Home**".
3. В **MainWindow.xaml** сменете стойността на **Source** свойството на **NavigationWindow** на "**ExpenseltHome.xaml**". Това ще зададе ExpenseltHome.xaml да бъде първата страница отворена, когато приложението стартира.
(Сменяйте го или от редактора на свойства, като маркирате цялата форма в дизайнера, или директно в XAML описанието в Markup)
4. Добавете още една страница както в точка 1, но я кръстете **ExpenseReportPage.xaml** този път. В ExpenseReportPage.xaml сменете името **Title** на "**Expenselt - View Expense**". Тази страница ще показва справката за разходите на избрания от първата страница човек.
5. Свалете изображението от адрес <https://i-msdn.sec.s-msft.com/dynimg/IC816740.jpeg> и го запаметете на компютъра под името watermark.png. Добавете изображението към проекта, като натиснете десен бутон върху проекта → Add → Existing Item ... и изберате файла на изображението.
6. Компилирайте и стартирайте приложението като натиснете F5 или изберете Start Debugging от менюто Debug.

Приложението ви трябва да изглежда по подобен начин. Забележете характерните за **NavigationWindow** бутона горе в ляво.



7. Затворете приложението и нека продължим с разработката му във Visual Studio.

2. Подредба на елементите (Layout)

Подредбата (Layout) позволява различни начини на позициониране на UI елементите, като се грижи за тяхната големина и положение при преоразмеряване на потребителския интерфейс.

Подредбата на интерфейса се осъществява чрез **панели**.

Панелите в WPF са контроли, които служат като контейнери и определят местоположението на вложените в тях контроли.

Панели са контролите:

- Canvas
- WrapPanel
- StackPanel
- DockPanel
- Grid
- UniformGrid

Вложените контроли в панелите може да са други панели.

Всяка форма може да съдържа като цяло само една контрола, т.е. съдържа един панел, в който са поместени всички останали контроли.

В дотук разглежданите примери основният панел е таблица Grid.

2.1. Работа с таблици (Grid)

В тази част ще създадем **таблица** с **една колона** и **3 реда**, като добавим дефиниции за редове и колони в Grid-а на ExpenseltHome.xaml.

1. В ExpenseltHome.xaml, настройте **Margin** свойството на Grid елемента на "**10,0,10,10**". Тези стойности определят разстоянията от ляво, горе, дясно и долу.
2. Добавете следния XAML код между отварящия и затварящ таг Grid, за да създадете дефинициите на редове и колони.

```
<Grid.ColumnDefinitions>
    <ColumnDefinition />
</Grid.ColumnDefinitions>
<Grid.RowDefinitions>
    <RowDefinition Height="Auto"/>
    <RowDefinition />
    <RowDefinition Height="Auto"/>
</Grid.RowDefinitions>
```

Таговете `<Grid.ColumnDefinitions>` и `<Grid.RowDefinitions>` :

- Служат за настройка на тагът-родител `<Grid>`.
- Те не са част от съдържанието.
- Заради това се постават или веднага след отварящия `<Grid>`
- Останалото съдържание в грида се съобразява с посоченото в тях

Височината на два от редовете се задава като Auto, което означава, че реда ще се оразмери в зависимост от неговото съдържание. Височината по подразбиране е със стойност звездичка (*),

което означава, че реда ще заеме претеглена пропорция от свободното място. Например ако имаме два реда, всеки от които е с височина "*", и двата ще имат височина равна на половината от заделеното пространство.

2.1.1. Добавяне на контроли на позиция

В тази подточка ще променим UI на началната страница, така че да показва списък от хора, от които потребителя да може да избира. Контролите, които добавяме, са UI обекти, които позволяват на потребителя да взаимодейства с приложението.

За да създадем този UI, ще използваме следните елементи и ще ги добавим към ExpenseltHome.xaml:

- ListBox (за списъка от хора).
- Label (за хедър на списъка).
- Button (за да може да се натиска и да сочи към справката за разходите на избрания човек).

Всяка контрола се поставя в ред на Grid-а като се настройва свойството Grid.Row (attached property).

1. В ExpenseltHome.xaml добавете следния XAML код между </Grid.RowDefinitions> и </Grid> таровете.

```
<!-- People list -->
<Label VerticalAlignment="Center" Foreground="White">Names</Label>
<ListBox Name="peopleListBox">
  <ListBoxItem>Mike</ListBoxItem>
</ListBox>
```

2. Посочете къде в грида се намират добавените контроли:

2.1. Поставете Label-а на първия ред

```
Grid.Column="0" Grid.Row="0"
```

2.2. Поставете ListBox-а на втория ред

```
Grid.Column="0" Grid.Row="1"
```

Стартирайте програмата с **F5** или зеления триъгълник.

3. Добавете бутона

```
<!-- View report button -->
<Button Margin="0,10,0,0">View</Button>
```

3.1. Добавете го на третия ред в таблицата

3.2. Направете го широк 125 и висок 25 пиксела (`Width="" Height=""`)

3.3. Преместете го в десния край на реда (`HorizontalAlignment=""`)

Стартирайте програмата с **F5** или зеления триъгълник.

4. Нека поставим надписа (Label-a) на тъмен фон. Преместете го в (огредете го с) Border контрола:

```
<Border Height="35" Padding="5" Background="#4E87D4">  
  
</Border>
```

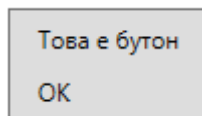
Тогава: Label-a вече се намира в Border, а не в грида. На самия Label вече няма нужда да посочваме къде се намира в грида, а на Border-a трябва да посочим. (Поставете го на същото място, където до сега беше Label-a.)

2.1.2. Влагане на таблица

- Таблицата също е контрола.
- Както вложихме по-горе Label в Border, така може и да влагаме таблици

Пример: Бутон разделен на два реда:

```
<Button Margin="350,36,0,0" Width="100">  
  <Grid>  
    <Grid.RowDefinitions>  
      <RowDefinition/>  
      <RowDefinition/>  
    </Grid.RowDefinitions>  
  
    <Label Grid.Row="0" Content="Това е бутон"/>  
    <Label Grid.Row="1" Content="OK"/>  
  </Grid>  
</Button>
```



В проекта **StudentInfoSystem**:

Задача: В предното упражнение използвахме GroupBox само като визуална рамка на итнтерфейса. Сега можем да сложим и контролите в GroupBox-a в самия XAML:

- Вложете в GrouBox контролите по една таблица Grid
- Преместете контролите, които трябва да са групирани заедно, в Grid-a на GroupBox-a.

2.2. Списъчни контроли

WPF раполага с множество контроли способни да визуализират списъци.

По-горе използвахме ListBox:

ListBox контрола е най-простата списъчна контрола позволяваща избор на отделен елемент.

- Списъкът, който се визуализира е от обекти. Няма задължителен тип.

- Т.е. вътре може да се помести списък от други контроли ,напр. Label
- Съществува специализирана контрола ListBoxItem, подобна на Label, която може да има съдържание

В предиен пример добавихме един елемент от тип ListBoxItem, т.е. до тук в ListBox-а има списък с един елемент.

В проекта **Expenselt**:

1. Добавете в списъка (ListBox) още: Lisa, John, Mary, Theo

Стартирайте програмата с **F5** или зеления триъгълник.

Приложението би трябвало да изглежда подобно.



2.2.1. Работа с елементи на ListBox

- Списъкът от обекти в `ListBox` се съхранява в свойството `Items`.
- Текущият маркиран обект от списъка се пази в `SelectedItem`
 - ...а индексът му се пази в `SelectedIndex`
- `SelectedItem` може да се използва и за задаване, т.е. да променим през кода текущия маркиран обект

1. Добавете в конструктора на прозореца (след метода `InitializeComponent()`) още двама души: James и David:

```
peopleListBox.Items.Add("James");
peopleListBox.Items.Add("David ");
```

Стартирайте програмата с **F5** или зеления триъгълник.

Така се получава известно разминаване, защото в XAML добавяме в списъка обекти от тип `ListBoxItem` (`<ListBoxItem></ListBoxItem>`), а конструктора добавихме низове (`String`)

2. Променете добавянето в конструктора с обекти от тип `ListBoxItem`:

```
ListBoxItem james = new ListBoxItem();
james.Content = "James";
peopleListBox.Items.Add(james);
```


... и аналогично за David.

3. Задайте по подразбиране да е избран James:

```
peopleListBox.SelectedItem = james;
```

Редът ще е маркиран, но може и да не се вижда без скролване. Има друга функция за скролване на ред, така че да стане видим.

4. Добавете бутон, който да изкарва съобщение с името на текущия избран човек от списъка.
 - а. Добавете бутон до бутона "View" (в същата клетка в таблицата)
(Може да го зададете в левия край на реда `HorizontalAlignment="Left"`)
 - б. В събитие на бутона изведете съобщението:

```
string greetingMsg;  
greetingMsg = peopleListBox.SelectedItem.ToString();  
MessageBox.Show("Hi " + greetingMsg);
```

Тъй като `SelectedItem` е от тип `object`, базовия метод изкарва и името на класа на обекта:



докато обектите, които сме поместили в списъка са от тип `ListBoxItem`, който има свойство `Content`, в което се съдържа видимата част (която се визуализира в списъка).

5. Поправете кода на бутона да се обръща към текущия маркиран обект в списъка като към `ListBoxItem`:

```
(peopleListBox.SelectedItem as ListBoxItem).Content.ToString()
```

2.3. Добавяне на изображение и заглавие

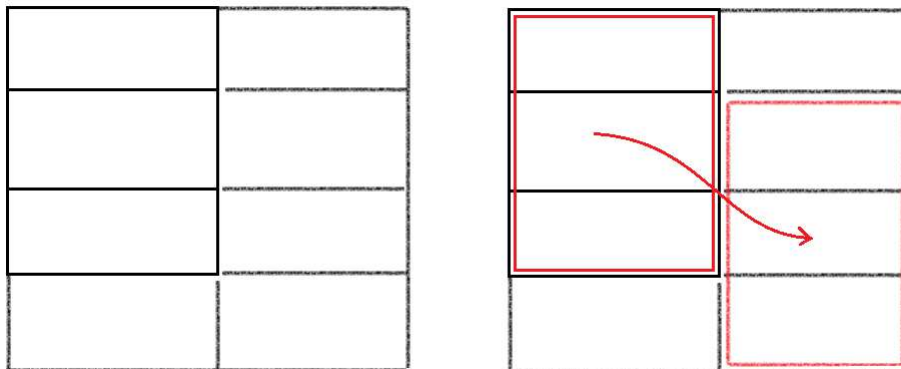
1. В `ExpenseltHome.xaml` добавете още една колона към `ColumnDefinitions` с фиксирана ширина `Width` на 230 пиксела.

```
<ColumnDefinition Width="230" />
```

2. Добавете и още един ред в `RowDefinitions`.

```
<RowDefinition />
```

3. „Преместете“ контролите във втората колона като настроите **Grid.Column** на **1**. Преместете и всяка от контролите с 1 ред надолу, като увеличите стойността на **Grid.Row** с **1**.



Така ще освободим място да се вижда фона.

4. Настройте фона на Grid елемента, така че да приема картинката **watermark.png**, която добавихме в предните стъпки.

- 4.1. Добавете следния таг, настройващ целия грид, при `<Grid.RowDefinitions>` и `<Grid.ColumnDefinitions>` таговете.

```
<Grid.Background>
```

```
</Grid.Background>
```

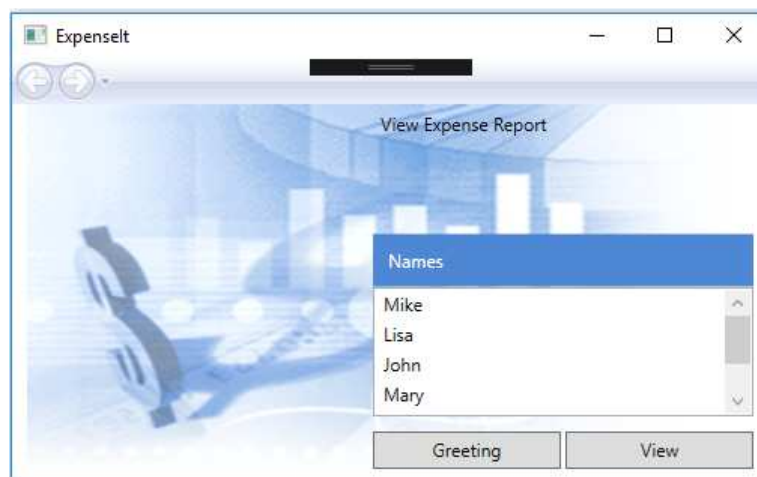
- 4.2. Поставете в него настройка за ImageBrush:

```
<ImageBrush ImageSource="watermark.png"/>
```

5. Преди Border, добавете Label елемент със съдържание "View Expense Report", представляващ заглавието на страницата.

```
<Label Grid.Column="1">View Expense Report</Label>
```

6. Компилирайте и стартирайте приложението.



3. Навигация между страници

1. В **ExpenseltHome.xaml** изберете **бутона** и създайте събитие за **Click**, в което ще пишем.
2. Добавете следния код в метода **Button_Click**

```
ExpenseReportPage expenseReportPage = new ExpenseReportPage();  
this.NavigationService.Navigate(expenseReportPage);
```

3.1. Създаване на UI за втората страница

ExpenseReportPage.xaml е предназначен да показва справката с разходите на конкретен човек, избран от страницата ExpenseltHome.xaml.

В тази подточка ще добавим контроли и ще създадем графичния интерфейс на ExpenseReportPage.xaml. Ще добавим фонове и цветове за някои UI елементи.

1. Отворете **ExpenseReportPage.xaml** и добавете в него XAML код между Grid таговете, така че да добие следния изглед:

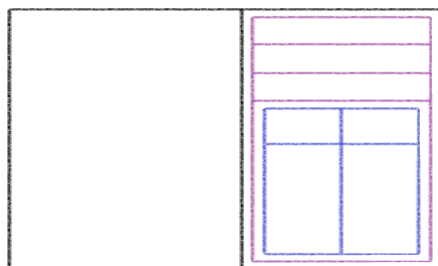


Този UI е подобен на графичния интерфейс в ExpenseltHome.xaml.

Подсказки:

- Внимавайте на кои редове ще зададете **Height="Auto"**
- Гридове могат да се вложат един в друг
- Под Expense Type и Amount може да поставите ListBox контроли с примерни данни (за сега)

Примерна структура на таблиците (може да използвате друга):



Изисква три вложени таблици

```
<Grid>
  <Grid Grid.Column="1">
    <Grid Grid.Row="3">

      <!-- Примерни данни за разходи -->

    </Grid>
  </Grid>
</Grid>
```

Вложената е във втората колона, а най-вложената – на 4тия ред на вложената.

```
<Grid>
  <Grid Grid.Column="1">
    <Grid Grid.Row="3">

      <!-- Примерни данни за разходи -->

    </Grid>
  </Grid>
</Grid>
```

Най-външната таблица има 2 колони

```
<Grid.ColumnDefinitions>
  <ColumnDefinition Width="*" />
  <ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
```

Вложената таблица има 4 реда

```
<Grid.RowDefinitions>
  <RowDefinition Height="*" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
</Grid.RowDefinitions>
```

Най-вложената таблица има 2 реда и 2 колони

```
<Grid.RowDefinitions>
  <RowDefinition Height="Auto" />
  <RowDefinition Height="*" />
</Grid.RowDefinitions>
<Grid.ColumnDefinitions>
  <ColumnDefinition Width="*" />
```

```
<ColumnDefinition Width="*" />
</Grid.ColumnDefinitions>
```

В таква структура или по-подходяща ваша трябва да положите подходящите контроли на подходящите места.

2. Компилирайте и стартирайте приложението.
3. Натиснете бутона **View**, за да се покаже страницата за справката с разходи.

Забележете, че бутона „назад“ от навигацията е станал автоматично активен.

4. Ресурси в XAML

За постигане на гъвкавост WPF дава възможност да запазим XAML код като ресурс, който след това можем да използваме много пъти. Това е много полезно.

Ресурси можем да записваме към:

- дадена контрола
- даден прозорец
- глобално за цялото приложение

Кода в ресурса може да бъде практически всичко:

- данни
- обекти
- контроли
- йерархия от контроли

Нека добавим заглавието на програмта като ресурс.

1. В ExpenseltHome.xaml добавете таг за ресурси към страницата.

```
<Page.Resources>
```

```
</Page.Resources>
```

2. Нека добавим ресурс от тип string със заглавието на страницата.

```
<sys:String>View Expense Report</sys:String>
```

- 2.1. Трябва да добавим библиотеката, в която се съдържа . В тагът **<Page>** добавяме атрибут-дефиниция:

```
xmlns:sys="clr-namespace:System;assembly=mscorlib"
```

(На практика `System.String`)

- 2.2. Трябва да зададем идентификатор на ресурса, за да можем да го намираме по него

```
x:Key="strTitle"
```

3. За да използваме новия ресурс в Label контролата, в която извеждаме заглавието на програмата, трябва да посочим съдържанието ѝ в атрибут, а не в явен вид.
 - 3.1. Изтриваме текста между отварящия и затварящ таг: `<Label Grid.Column="1"></Label>`
 - 3.2. Добавяме атрибут `Content`
 - 3.3. Задаваме му стойност `"{StaticResource strTitle}"`

Стартирайте програмата с **F5** или зеления триъгълник.

5. Стилизиране в XAML

Изгледът на контролите в XAML може да се описва със стилове. Стилите могат да се прилагат върху множество контроли.

5.1. Синтаксис на стилове

1. Можем променяме директно свойствата на дадена контрола чрез атрибутите и вложени тагове. Например за Label контролата със заглавието на програмата (в ExpenseltHome.xaml) можем да зададем атрибути (може би вече имате някои от тях):

```
VerticalAlignment="Center"  
FontFamily="Trebuchet MS"  
FontWeight="Bold"  
FontSize="24"  
Foreground="#0066cc"
```

Стартирайте програмата с **F5** или зеления триъгълник.

2. Същото можем да преработим и като стил:
 - 2.1. Добавете **таг за ресурси** на Label контролата.
 - 2.2. В него добавете таг

```
<Style TargetType="Label"> </Style>
```

- 2.3. В тагът за стил добавяме `<Setter></Setter>` тагове за всяко свойство, което искаме да задава стила:

- 2.3.1. Например за атрибута `VerticalAlignment` добавяме таг:

```
<Setter Property="Label.VerticalAlignment" Value="Center"></Setter>
```

- 2.3.2. За атрибута `FontFamily` добавяме `<Setter></Setter>` таг и му задаваме `Property="Label.FontFamily"` и `Value="Trebuchet MS"`.

- 2.4. Можем да премахнем атрибутите в `<Label ...>` и да оставим само стила.

Стартирайте програмата с **F5** или зеления триъгълник.

Изгледът не следва да се е променил.

5.2. Стилите за много контроли

Няма смисъл да задаваме стил за всяка контрола.

Има смисъл да зададем стил на по-горно ниво (грид, страница, прозорец, програма) за да го използват всички вложени контроли.

1. Нека добавим глобален стил за всички Label контроли в цялото приложение (освен онзи със заглавието, за който вече зададохме частен стил).

- 1.1. Отворете App.xaml и добавете `<Style>` таг във вече съществуващия таг за ресурси на `Application`:

```
<Style TargetType="Label"></Style>
```

- 1.2. Добавете по един `<Setter>` за всяко от следните свойства:

```
VerticalAlignment ="Top"  
HorizontalAlignment ="Left"  
FontWeight ="Bold"  
Margin ="0,0,0,5"
```

Стартирайте програмата с **F5** или зеления триъгълник.

Label контролите са променени. (Повечето са в ExpenseReportPage.xaml). Надписът "Names" не се събира в заделеното място.

5.3. Стиллове като ресурс

Ако зададем стойност на свойството `x:Key` на даден стил, WPF разбира, че искаме да го използваме само за конкретни контроли по идентификатор, и няма да го приложи автоматично.

Например:

2. Да добавим нов стил в ресурсите на App.xaml:

```
<Style TargetType="Label"></Style>
```

- 2.1. Да му зададем да определя свойствата:

```
VerticalAlignment ="Top"  
HorizontalAlignment ="Left"  
Foreground =" White"
```

- 2.2. Да му зададем идентификатор

```
x:Key="listHeaderTextStyle"
```

- 2.3. За Label контролата със съдържание "Names" **заменяме** използваните атрибути с единствено:

```
Style="{StaticResource listHeaderTextStyle}"
```

Стартирайте програмата с **F5** или зеления триъгълник.

Надписът "Names" използва отделен стил и отново се събира в заделеното място.

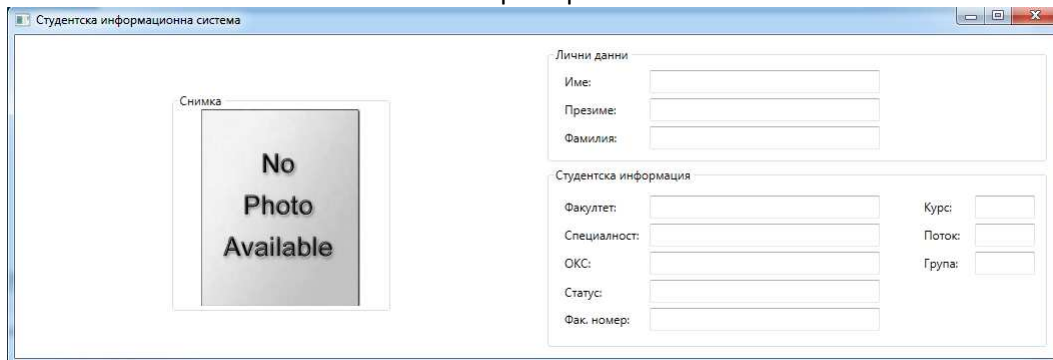
Общи задачи (пример за работа в час):

В проекта **StudentInfoSystem**:

Задача: Създайте следната функционалност:

- Контролите за данни на студент да са вдясно на страницата/прозореца (включително при промяна размерите на прозореца от потребителя).
- Отляво да се визуализира снимка на студента (Image контрола)

Пример:



Запишете си проекта! USB, DropBox, GoogleDrive, e-mail, SmartPhone, където и да е.

До края на упражненията ще работите върху този проект.

Вкъщи също ще работите върху този проект.

Накрая трябва да го представите завършен.