

Sofia University
Department of Mathematics and Informatics

Course : OO Programming C#.NET

Date: December 6, 2018

Student Name:

Lab No. 13

Submit the all C# .NET files developed to solve the problems listed below. Use comments and Modified-Hungarian notation.

Problem No.1

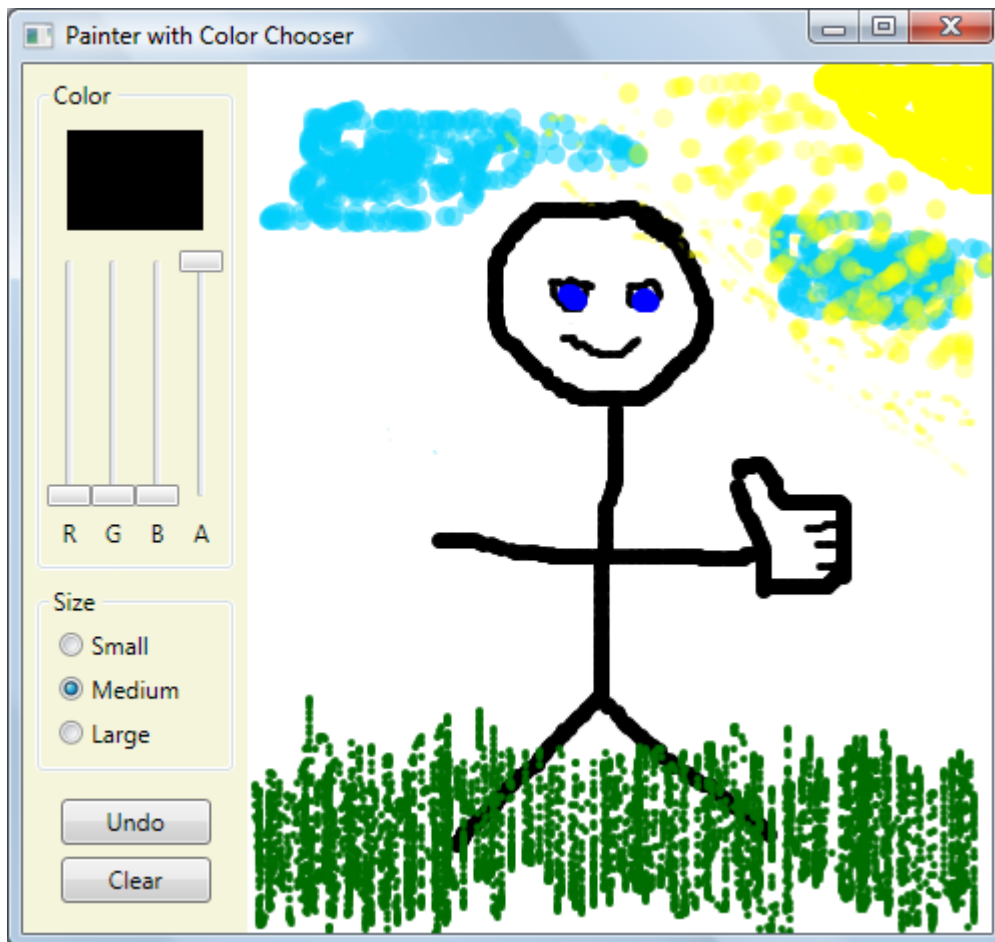
Create a WPF **UserControl** called **LoginPasswordUserControl**. The **LoginPasswordUserControl** contains a **Label (lblLogin)** that displays **String "Login:"**, a **Text-Box (txtLogin)** where the user inputs a **login name**, a **Label (lblPassword)** that displays the **String "Password:"** and finally, a **TextBox (txtPassword)** where a user inputs **a password** (do not forget to set property **PasswordChar** to **"*"** in the **TextBox's Properties** window). **LoginPasswordUserControl** must provide **Public** read-only properties **Login** and **Password** that allow an application to retrieve the user input from **txtLogin** and **txtPassword**. There should be **button OK** to export to an application the values input by the user in **LoginPasswordUserControl**. Accordingly, **button Cancel**, clears the strings in the textboxes (**txtLogin**, **txtPassword**) and exports empty strings to the host application (the one that embeds the user control)

Write a C#.NET WPF application to **test** the **LoginPasswordUserControl**- the main form of the application contains an instance of the **UserControl**. In **addition** to the user control the main form should have a **TextBox**. On clicking the **OK** of the **LoginPasswordUserControl** the strings for the username/password input by the user in the user control are added accordingly to the **TextBox**. On clicking the **Cancel** of the **LoginPasswordUserControl** a **MessageBox** displays a **warning message** that no username/password are entered.

Hint: Add a Reference to System.Xaml

Problem No.2

Incorporate an RGBA color chooser into the Painter example (attached to the Lab as **Painter.rar**) to look like the figure displayed below. Let the user select the brush color using the color chooser instead of the group of **RadioButtons**. You should use a style to make all the sliders look the same.



Additionally, **modify the drawing procedure to use line segments** connecting consecutive positions of the mouse cursor, while dragging it over the Canvas. The free **line thickness and color**, **Undo** and **Clear** must be **managed by means of the tools on the left side** of the Canvas as shown on the above figure

Problem No. 3

Create a C#.NET calculator as a WPF **UserControl** that allows the user to input numbers in a textbox and choose an operation to perform on them (addition, multiplication, division, subtraction) as it is done with a usual calculator (*see the design of the Calculator application in the Accessories Program group in the MS Windows environment*) . Design buttons to execute these operations, as well as, buttons:

- a) to remember the currently displayed number (**M** operation)
- b) to add the currently displayed number with the number stored in memory and display the result (**M+** operation)
- c) to subtract the currently displayed number with the number stored in memory and display the result (**M-** operation)
- d) to clear the memory (**MC-** operation)

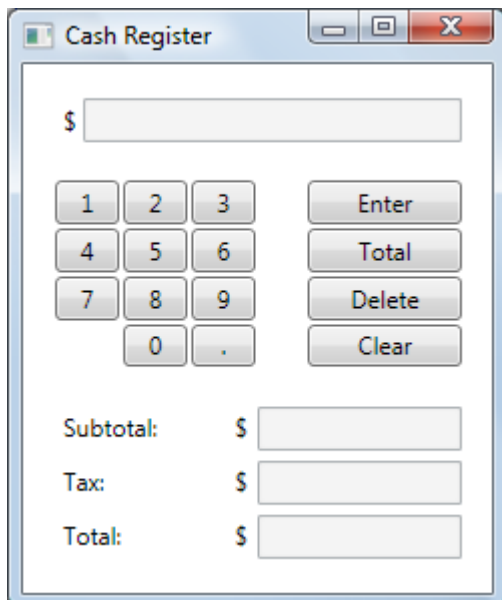
The methods performing the Calculator operations must be **public**. There should be also **two public set properties** for the user numeric input, necessary to complete the calculator operations. There should be a **public get property** for the Calculator result.

Catch division by zero exceptions, by canceling the division operation and displaying an error message in the textbox. **Allow only legal numeric user input** in the textbox.

Write a C#.Net WPF application to test this **user control**. (*submit the source code of the user control, its DLL file and the full set of files of the C#.Net Windows application*)

Problem No.4

Create a cash-register application modeled after the one presented below. It should allow users to enter a series of prices, then obtain the total. The **Delete** button should clear the current entry, and the **Clear** button should reset the application.



Problem No.5

Create a C#.NET calculator as a WPF **UserControl** that allows the user to input numbers in a textbox and choose an operation to perform on them (addition, multiplication, division, subtraction) as it is done with a usual calculator (*see the design of the Calculator application in the Accessories Program group in the MS Windows environment*) . Design buttons to execute these operations, as well as, buttons:

- e) to remember the currently displayed number (**M** operation)
- f) to add the currently displayed number with the number stored in memory and display the result (**M+** operation)
- g) to subtract the currently displayed number with the number stored in memory and display the result (**M-** operation)
- h) to clear the memory (**MC-** operation)

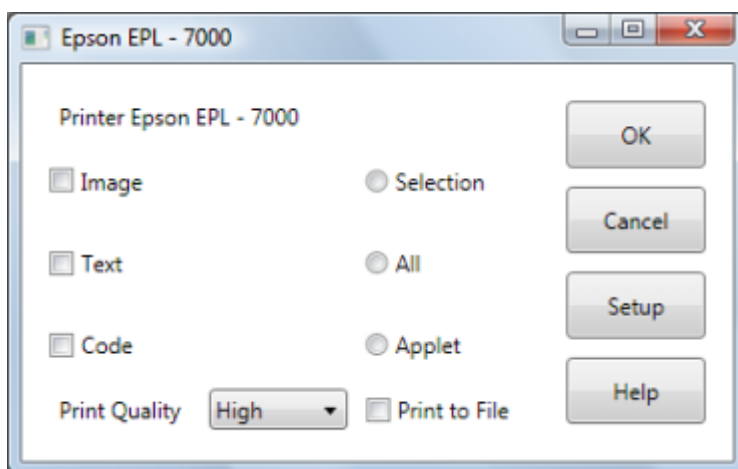
The methods performing the Calculator operations must be **public**. There should be also **two public set properties** for the user numeric input, necessary to complete the calculator operations. There should be a **public get property** for the Calculator result.

Catch division by zero exceptions, by canceling the division operation and displaying an error message in the textbox. **Allow only legal numeric user input** in the textbox.

Write a C#.Net WPF application to test this **user control**. (*submit the source code of the user control, its DLL file and the full set of files of the C#.Net Windows application*)

Problem No.6

Create the GUI displayed (you do not have to provide functionality) using WPF. Do not use a Canvas. Do not use explicit sizing or positioning



Problem No.7

WPF allows two-way data bindings. In a normal data binding, if the data source is updated, the binding's target will update, but not vice versa. In a two-way binding, if the value is changed in either the binding's source or its target, the other will be automatically updated. To create a two way binding, set the Mode property to TwoWay at the Binding's declaration. Create a phone-book application modeled after the one shown below. When the user selects a contact from the contacts list, its information should display in an editable GridView. As the information is modified, the contacts list should display each change.

Phone Book

Phone Book

Name: New Contact

Phone:

Email:

Name	Phone	Email
New Contac		

New

Delete