Assignment 3
GUI and Methods with parameters

CBS
Cinema Booking System
Version 1- GUI and Input
Mandatory

Farid Naisan
University Lecturer
Department of Computer Sciences
Malmö University, Sweden
## Contents

Assignment 3 ......................................................................................................................................................................................... 3  
1. Objectives ......................................................................................................................................................................................... 3  
2. Description .................................................................................................................................................................................. 3  
3. Requirements .............................................................................................................................................................................. 4  
4. The Project .................................................................................................................................................................................. 6  
5. Design of GUI .............................................................................................................................................................................. 6  
6. The InputUtility class ................................................................................................................................................................. 8  
7. The MainForm class ..................................................................................................................................................................... 9  
8. Help and Guidance ............................................................................................................................................................. 12  
    Input validation ........................................................................................................................................................................ 12  
    To set focus to and highlight the contents of a TextBox ........................................................................................................ 13
Assignment 3: CBS Version 1, GUI and Input

1. Objectives

The main objectives of this assignment are:

- To begin writing Windows Forms based desktop applications with graphical user interface (GUI) using most common Windows controls.
- Use parameterized methods to establish communication between objects.
- To handle input and perform input control, using the TryParse methods.

2. Description

A new movie theater has opened in your town and the owner needs a system that facilitates the reservation of seats in the cinema's auditorium. Your assignment is to write a GUI-based application that facilitates the reservation of tickets for this movie theater. The user of this application is a cinema staff, for example the Cashier. The Cashier registers the name of the customer and the price for the seat. The program assigns the first vacant seat number counted from the seat at the most rear part of the auditorium, i.e. the last chair. However, in this version, seats are not assigned as demonstrated in the sample run program below.

The main job in this assignment is:

- to design the GUI, with controls for input and output.
- to create a utility class for handling input, `InputUtility`, with methods that validates numerical values entered by the user.
- write code in the `MainForm` to read, validate and test the user input, using the `InputUtility` object.

The `InputUtility` class is meant to be reused in future projects as well and because it is a general utility class, the methods can be declared to be static.

Receiving input from the user through textboxes, Combiboxes, Buttons, etc. is something that you will always need to cope with. What is good about this
usually tedious task is that you proceed in the same way. Therefore, this exercise is designed to give you good training in acquiring input from the controls and validating the values given by the user. Make sure that you understand every step in what you are doing.

Table 1 summarizes the tasks and gives an overall idea of the tasks expected to be done in this version.

<table>
<thead>
<tr>
<th>To Do</th>
<th>Classes involved</th>
</tr>
</thead>
<tbody>
<tr>
<td>• Start a new Visual C# Windows Form project in Visual Studio</td>
<td>• MainForm</td>
</tr>
<tr>
<td>• Draw GUI on your Form (rename Form1 to MainForm)</td>
<td>• InputUtility</td>
</tr>
<tr>
<td>• Create a new class InputUtility with two static methods:</td>
<td></td>
</tr>
<tr>
<td>• GetDouble</td>
<td></td>
</tr>
<tr>
<td>• Get Integer</td>
<td></td>
</tr>
<tr>
<td>• Read input from Textboxes on the GUI and validate using the above methods from the InputUtility class</td>
<td></td>
</tr>
<tr>
<td>• Test the application and ensure that the application has full control of the values given by the user.</td>
<td></td>
</tr>
</tbody>
</table>

3. Requirements

3.1 The GUI must include textboxes for input, labels for readonly information such as headings and also output. In addition a listbox must also be used.

3.2 The InputUtility class must be saved on a separate file.

3.3 The values entered by the user in the textboxes must be validated when the user clicks the Reserve/Cancel button.

3.3.1 The value entered in the name textbox should at least contain one character that is not a blank, otherwise a message box is to be shown to the user with appropriate error.

3.3.2 The value entered in the price textbox must be a valid double value greater or equal to 0 (0 for free tickets)

3.4 Test the application with a total number of seats = 240. The program should keep track of the number of vacant seats.

3.4.1 Every time the user clicks the Reserve/Cancel button, and if the radio button Reserve is
checked, increase the number of vacant seats by one. Also accumulate the price of each reserved seat to show in the revenue output label, i.e. revenue = Sum of prices.

3.5 When the user selects the Cancel Reservation option, the TextBoxes are to be disabled. Even the Reserve/Cancel button is to be disabled as cancel function will not be implemented in this version. We will work with that in the next assignment.

3.6 Messages boxes used to give error messages to the user, should have a caption and an icon, as those given earlier in this document. As an exercise to find documentation and other information from the MSDN, it is required that you search at msdn.microsoft.com and read about the MessageBox class (.NET Framework 4) and learn how the buttons and icons are constructed.

3.7 All methods must be commented using the /// comment type. You must know the benefits of using this type of comments to document your classes and their members. Discuss this in the forum if you need to have more clarity on this question.
4. The Project

Create a new Visual C#, Windows Form application in Visual Studio (VS). VS will then create also a solution for your project. Change the default class file name, Form1 to **MainForm**. VS will be asking you whether you wish to even change class name which you should accept thankfully.

5. Design of GUI

5.1 Draw and Design the GUI, using the Toolbox in VS, to match the run example shown above. You may of course use your imaginations and bring improvements to this design. The controls to be used are outlined in the table that follows. Give controls suitable names. It is recommended that you begin the names with a tre-letter prefix as suggested in the table.

<table>
<thead>
<tr>
<th>Control</th>
<th>Name</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>ListBox</td>
<td>lstSeats</td>
<td>Showing all vacant/reserved seats in the cinema’s auditorium</td>
</tr>
<tr>
<td>GroupBox</td>
<td>grpInput, grpOutput</td>
<td>Grouping radio buttons and other input/output components.</td>
</tr>
<tr>
<td>RadioButton</td>
<td>rbtnReserve, rbtnCancel</td>
<td>Determines if you want to reserve or cancel a reservation i.e. make a seat booked or vacant.</td>
</tr>
<tr>
<td>TextBox</td>
<td>txtName, txtPrice</td>
<td>Takes input from the user.</td>
</tr>
<tr>
<td>Button</td>
<td>btnOK</td>
<td>Issues a reservation / cancellation.</td>
</tr>
<tr>
<td>Label</td>
<td>lblNumOfSeats, lblNumOfReservedSeats, etc..</td>
<td>Used for naming other components and for output. (Use Fixed3d border for output labels). Labels used for headings can have their default values like Label1, etc, but those that will be addressed from code must have suitable names, like lblNumOfSeats.</td>
</tr>
</tbody>
</table>

The figure on the next page illustrates these graphically.
Figur 1: GUI Design
6. The InputUtility class

This class is intended to be used in your later assignments (and your future projects, even after the course completion) as well.

6.1 Create a new class; name it as **InputUtility** and save it as **InputUtility.cs**.

6.2 Write two methods, **GetDouble** and **GetInteger** that:

6.2.1 convert a given string into a **double** and **int** respectively, and

6.2.2 if conversion is successful, validate that the converted value is within a given range.

6.2.3 The figure below shows the method signature for the **GetDouble** method. Write the **GetInteger** method in the same manner.

```csharp
public class InputUtility
{
    /// <summary>
    /// Converts a string represented Double value into a Double type, and validates
    /// the converted value to be within (and inclusive) a range, defined by minLimit and
    /// maxLimit.
    /// </summary>
    /// <param name="stringToConvert">String representing the Double value.</param>
    /// <param name="dblOutValue">Output parameter, the converted Double value.</param>
    /// <param name="minLimit">The output value should be greater or equal to minLimit.</param>
    /// <param name="maxLimit">The output value should be less than or equal to maxLimit</param>
    /// <returns>true if the conversion is successful and the converted value is within the
given range.</returns>
    public static bool GetDouble(string stringToConvert, out double dblOutValue, double minLimit, double maxLimit)
    {
        //Write code to complete
    }
}
```

6.2.4 Write code in the above method to make the function work properly. It is a requirement to use the **double.TryParse** and **int.TryParse** methods to perform the conversions.
Requirement for a higher grade (ECT A and B, Swe VG, skip if you aim at only a passing grade): In C#, you can write methods that have same name but different types of parameters, or different number of parameters. The methods are said to be overloaded. Overload the GetDouble and GetInteger methods as follows:

6.2.5 Write a GetDouble method and a GetInteger method as defined in above figure, but without the maxLimit parameter, so it can be used when a string is to be converted to a numerical value with only the minLimit condition. For example when you wish to get a value >= 0 for the price – no max limit required.

6.2.6 A GetDouble method and a GetInteger method without both the minLimit and maLimit. This method can be useful when you have no min and max conditions.

7. The MainForm class

Now that you have programmed the InputUtility class, you can use its GetDouble method in the MainForm class to read and validate the user input.

7.1 Input: The user (the cashier) must provide the following data (which will make our input):

7.1.1 The name of the customer (string) who wishes to reserve a seat in the cinema.

7.1.2 The price for the seat (double).

7.1.3 The choice of Reserve or Cancel Reservation (the Checked value of the RadioButtons is to be used in coding).

7.1.4 These variables are to be declared and created as local variables.

7.2 Output: To store number of reserved seats and the sum of the prices as a new clicking on the button takes place, you might need to declare a couple of instance variables, as exemplified in the code clip below.

7.3 To begin your coding, open the MainForm's code file; you may see an almost empty class with only the constructor; this is because VS hides its own generated code in another file. You can begin declare your fields as in the figure above. The table below lists the methods that are to be written in the MainForm class.

7.4 The method InitializeComponents is a method that is generated by VS. Begin your own code after this line (call to InitializeComponent is an example).
<table>
<thead>
<tr>
<th>Method and description</th>
<th>Purpose/Comments</th>
</tr>
</thead>
</table>
| 1                      | • Do initializations here which will be executed before the form is displayed.  
                          • This method must be called from the Constructor, after the call to `InitializeComponents`, as explained in the previous page.  |
|                        |                  |
| 2                      | • Double-click on the button you have drawn on the Form.  
                          • Copy the code given here.  
                          • Complete the methods as described below.  |

```csharp
1
/// <summary>
/// Clear the input and output controls (if needed).
/// Do other initializations, for example select one of the radio-
/// buttons as default.
/// </summary>
/// <remarks>This is to be called from the constructor, AFTER the
call to InitializeComponents.</remarks>
private void InitializeGUI()
{
    btnReserved.Checked = true;
    lstReservations.Items.Clear();
    txtName.Text = string.Empty;

    //continue with other initializations
}

2
/// <summary>
/// Event-handler method for the Click-event of the button. When the user
/// clicks the button, this method will be executed automatically.
/// Call the ReadAndValidateInput method, save its return value in a
/// Boolean variable, if the return value is true, then call
/// the UpdateGUI method to display the results.
/// </summary>
/// <param name="sender">Reference to the object that has fired the Click event (the button)</param>
/// <param name="e">Contains information about the event.</param>
private void btnOK_Click(object sender, EventArgs e)
{
    string customerName = string.Empty;
    double seatPrice = 0.0;

    bool inputOk = ReadAndValidateInput(out customerName, out seatPrice);

    if (inputOk)
    {
        numOFReservedSeats++;  
        revenue += seatPrice;  
        UpdateGUI(customerName, seatPrice);
    }
}
```
### ReadAndValidateName(out string name)

- Has the name entered in the name TextBox at least one char?
  - If yes, return true. The out parameter will have a valid value returning to the caller.
  - If no show a message box with a friendly error message
    - Set the focus to the textbox
    - return false.

**Tips:** The method `string.IsNullOrEmpty` comes to good help here.

### ReadAndValidatePrice(out double price)

- Convert the contents of the price TextBox to a `double` and validate the converted value so it is >= 0.0.
  - Call the GetDouble method from the InputUtility class, with min value 0 and max value some big number.
  - Use a `const` declaration for the max.value
  - If the validation is true, return true and the out parameter will have a valid value returning to the caller.
  - If the validation is not true, give a friendly message to the user.
  - Set the focus to this price textbox.
  - Return false.

### ReadAndValidateInput(out string name, out double price)

- Call the methods (3) and (4).
- Return true if both methods evaluate to a true value and return false otherwise.
- **Tips:** Use the `&&` operator on the results of the two method calls..
8. Help and Guidance

Input validation

C# provides a neat function for doing this called TryParse (read more at http://msdn.microsoft.com/en-us/library/f02979c7.aspx). TryParse relies on the use of `out` parameter, which is a way of having a function returning multiple values.

Each numeric type has a TryParse method for parsing string values into its type. The function for parsing a double looks as following:

```csharp
public static bool TryParse(string s, out int result)
```

What happens when you use TryParse is that when you pass a string representing a numeric value and an empty result variable, the string value will be parsed to a numeric type and stored in the result variable. If the parsing fails the function will return false, and if it succeeds it will return true.

This means that the function both returns a Boolean value and the result of the attempted parsing.

As an example assume that we have a numeric value entered in a TextBox on your form as in the figure. The input is saved by Windows as a string containing a sequence of characters, '8' '.' '9' '8' represented as a string:

```csharp
txtPrice.Text = "8.98";
```

This is because the Property Text of a TextBox is declared as string, but what we need is not the string "8.98", rather a double value 8.98 (no quotation marks). Therefore we must convert the contents of txtPrice.Text ("8.98") to a double value (8.98) and save it in a variable of double type.

In this case TryParse will return `true`, and the `price` variable will be assigned the value 8.98.

Why use `TryParse` and what do we do with the return value `goodNumber`? Well, next time, the user may write mistakenly or intentionally "8.9B" in the TextBox and this is not a number. If we try to convert this value to a number in other ways, we will get wrong results causing unpredictable consequences, or no results and in most cases abnormal termination of the program. The `TryParse` methods intelligently takes care all failures during the process of the conversion and let us know by its return value if the conversion has been successful or not. It is then important to always control the return value of the `TryParse` operation before proceeding with the next step.
To set focus to and highlight the contents of a TextBox

The call to **Focus** and **SelectAll** must be placed as the last lines in your code block so the focus to your control is not lost by other operations.

```csharp
MessageBox.Show(String.Format("Invalid input in price field!"), "Error!", MessageBoxButtons.OK, MessageBoxIcon.Error);

txtPrice.Focus();
txtPrice.SelectAll();
```

**Good Luck!**

*Programming is fun. Never give up. Ask for help!*

*Farid Naisan  
Course Responsible and Instructor*