CSE 215L: Programming Language II Lab
Faculty: Silvia Ahmed, Sec – 2, 3
Fall 2017
Lab Instructor: Marufa Ferdausi

1. Objectives

The main objectives of this assignment are:

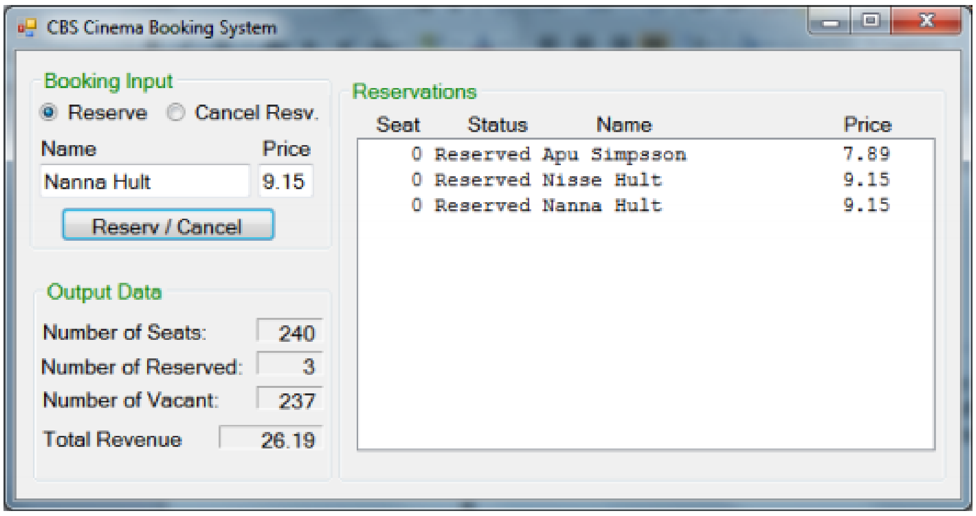
- To begin writing Windows Forms based desktop applications with graphical user interface (GUI) using most common Windows controls.
- Use parameterized methods to establish communication between objects.

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 ex 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.

Receiving input from the user through textboxes, Comboboxes, 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.



Name: Nanna Hult
Price: 9.15

Number of Seats: 240
Number of Reserved: 3
Number of Vacant: 237
Total Revenue: 26.19

```
Seat   Status     Name                  Price
  0 Reserved Apu Simpsson               7.89
  0 Reserved Nisse Hult                 9.15
  0 Reserved Nanna Hult                 9.15
```

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

To Do	Classes involved
• Draw GUI • Create a new class InputUtility with two static methods: ♣ GetDouble ♣ Get Integer • Read input from Textboxes on the GUI and validate using the above methods from the InputUtility class • Test the application and ensure that the application has full control of the values given by the user. :	• Mainform • InputUtility

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.

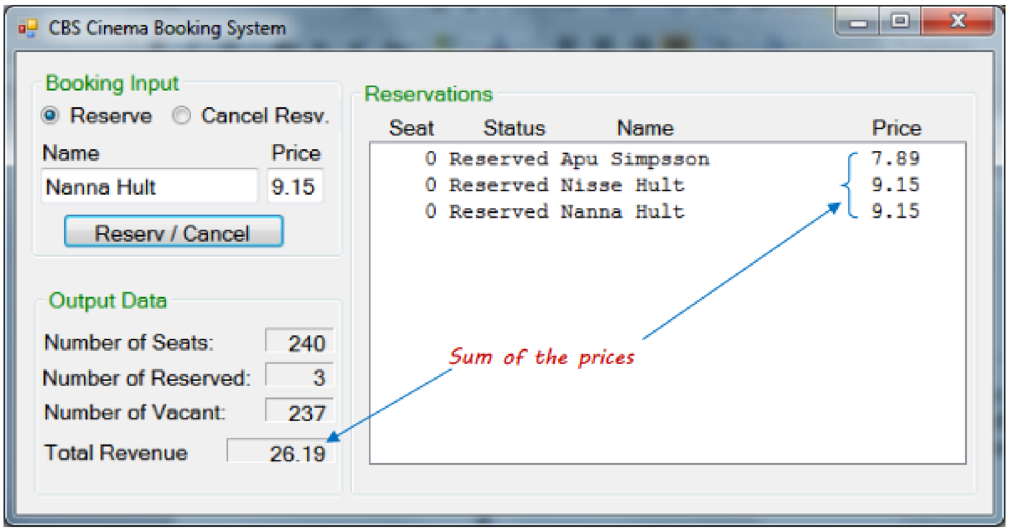
3.3 The values entered by the user in the textboxes must be validated when the user clicks the Reserve/Cancel button.

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 most 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.



Name: Nanna Hult
Price: 9.15

Number of Seats: 240
Number of Reserved: 3
Number of Vacant: 237
Total Revenue 26.19

```
0 Reserved Apu Simpsson      7.89
0 Reserved Nisse Hult        9.15
0 Reserved Nanna Hult        9.15
```

3.5 When the user selects the Cancel Reservation option, the TextBoxes are to be disabled

3.6 Messages boxes used to give error messages to the user, should have a caption and an icon. As an, it is required that you search 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

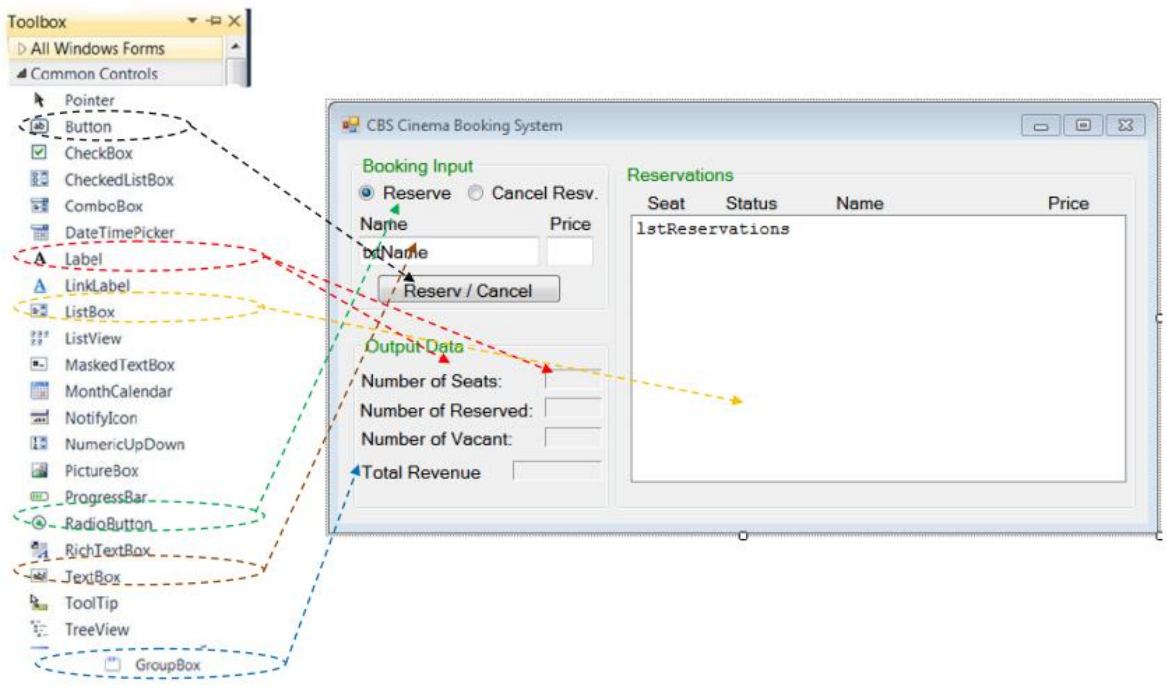
4. The Project

Create a new GUI which will then create also a solution for your project. Change the default class file name to MainForm.

5. Design of GUI

5.1 Draw and Design the GUI, using the Toolbox in Java, 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.

Control	Name	Purpose
ListBox	lstSeats	Showing all vacant/reserved seats in the cinema’s auditorium
GroupBox	grpInput, grpOutput	Grouping radio buttons and other input/output components.
RadioButton	rbtnReserve, rbtnCancel	Determines if you want to reserve or cancel a reservation i.e. make a seat booked or vacant.
TextBox	txtName, txtPrice	Takes input from the user.
Button	btnOK	Issues a reservation / cancellation.
Label	lblNumOfSeats, lblNumOfReservedSeats, etc..	Used for naming other components and for output. (Use Fixeded 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.



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.

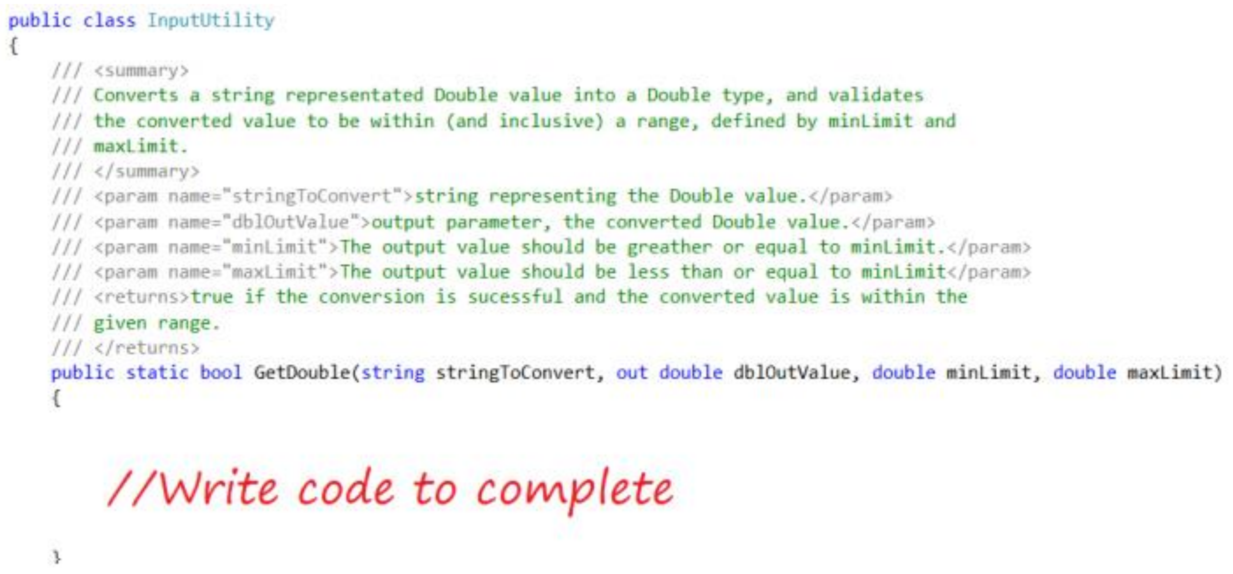
6.1 Create a new class; name it as InputUtility and save it as InputUtility.java.

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.



```
public class InputUtility
{
    /// <summary>
    /// Converts a string representated 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 greather or equal to minLimit.</param>
    /// <param name="maxLimit">The output value should be less than or equal to minLimit</param>
    /// <returns>true if the conversion is sucessful 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 parseDouble and parseInt methods to perform the conversions.

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.

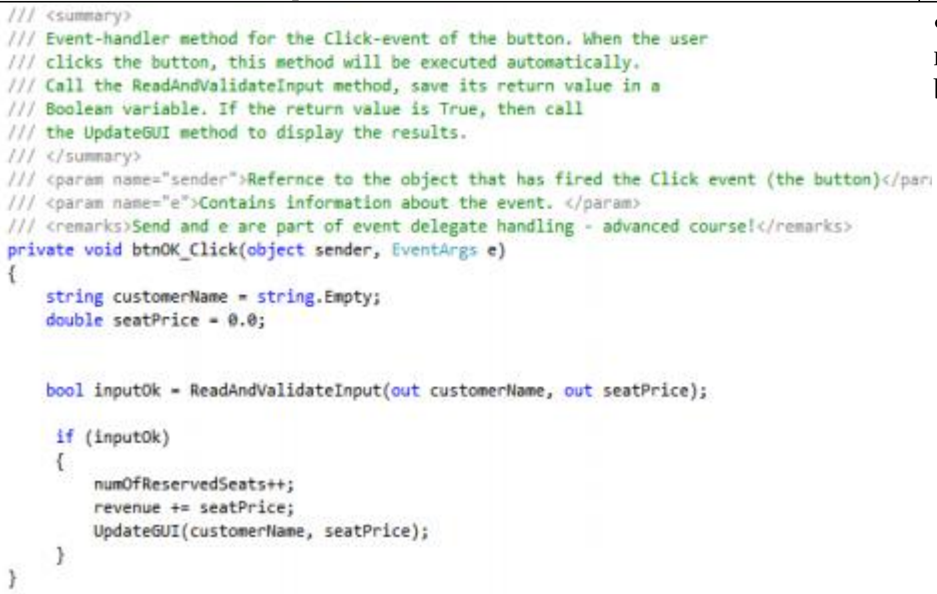
7.1.2 The price for the seat (double).

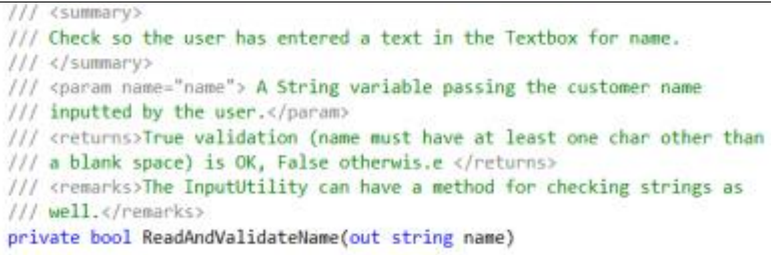
7.1.3 The choice of Reserve or Cancel Reservation (the Checked value of the RadioButtons is to be used in coding).

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

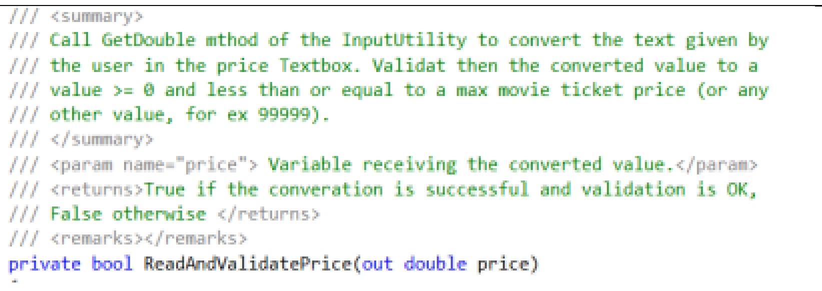
7.2 Output: To store number of reserved seats and the sum of the prices as a new clicking on the button takes place, you might need to declare a couple of instance variables, as exemplified in the code clip below.

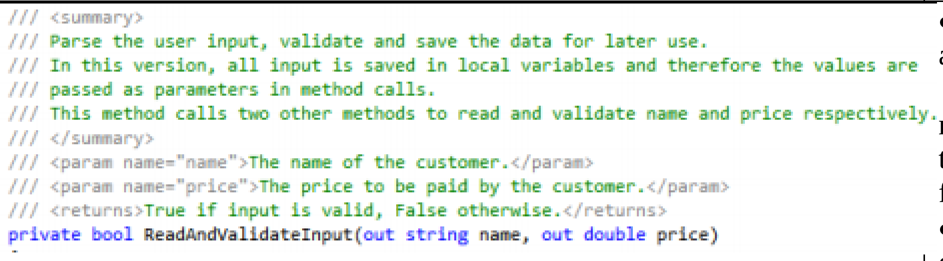
	Method and description	Purpose/Comments

1	
	`/// <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">Refernce to the object that has fired the Click event (the button)</par`
	`/// <param name="e">Contains information about the event. </param>`
	`/// <remarks>Send and e are part of event delegate handling - advanced course!</remarks>`
	`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);`
	`     }`
	`}`
		• Complete the methods as described below.

2	
	`/// <summary>`
	`/// Check so the user has entered a text in the Textbox for name.`
	`/// </summary>`
	`/// <param name="name"> A String variable passing the customer name`
	`/// inputted by the user.</param>`
	`/// <returns>True validation (name must have at least one char other than`
	`/// a blank space) is OK, False otherwis.e </returns>`
	`/// <remarks>The InputUtility can have a method for checking strings as`
	`/// well.</remarks>`
	`private bool 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
		• o return false. Tips: The method string.IsN

		ullOrEmpty comes to good help here.

3	
	`/// <summary>`
	`/// Call GetDouble mthod of the InputUtility to convert the text given by`
	`/// the user in the price Textbox. Validat then the converted value to a`
	`/// value >= 0 and less than or equal to a max movie ticket price (or any`
	`/// other value, for ex 99999).`
	`/// </summary>`
	`/// <param name="price"> Variable receiving the converted value.</param>`
	`/// <returns>True if the converation is successful and validation is OK,`
	`/// False otherwise </returns>`
	`/// <remarks></remarks>`
	`private bool 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. .

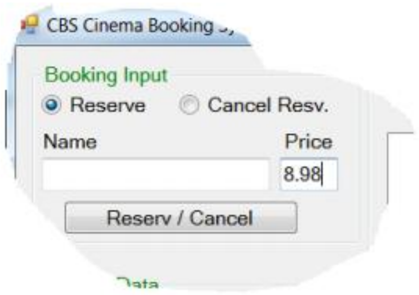
4	
	`/// <summary>`
	`/// Parse the user input, validate and save the data for later use.`
	`/// In this version, all input is saved in local variables and therefore the values are`
	`/// passed as parameters in method calls.`
	`/// This method calls two other methods to read and validate name and price respectively.`
	`/// </summary>`
	`/// <param name="name">The name of the customer.</param>`
	`/// <param name="price">The price to be paid by the customer.</param>`
	`/// <returns>True if input is valid, False otherwise.</returns>`
	`private bool ReadAndValidateInput(out string name, out double price)`
		• Call the methods (2) and (3).
		• 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

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:

```
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.



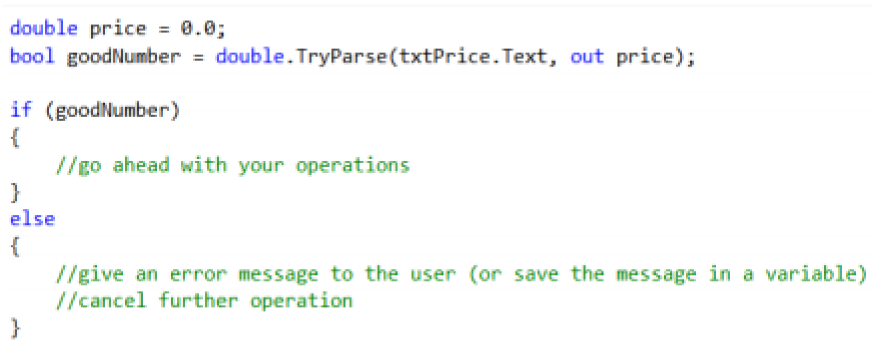
This means that the function both returns a Boolean value and the result of the attempted parsing.

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:

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.



```
double price = 0.0;
bool goodNumber = double.TryParse(txtPrice.Text, out price);

if (goodNumber)
{
    //go ahead with your operations
}
else
{
    //give an error message to the user (or save the message in a variable)
    //cancel further operation
}
```

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 unpredicted consequences, or no results and in most cases abnormal termination of the program. The parse 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.

Assigned on: 4 December, 2017

Submission Deadline: 13 December, 2017

This is a group assignment. The submission will be taken as per your project group.

Submission criteria:

1. Draw the UML diagram for your entire design and submit the hardcopy. (30 points)
2. Submit the softcopy of the project to the faculty and lab instructor. (70 points)

Note: During submission you will be interviewed shortly on your codes and design. Your marks will be valid only after this interview.