Course number : 420-CT2-AS

ORIENTED OBJECT PROGRAMMING

Teacher : Maftei Mihai

Weighting :	**30%**	Group :	**07194**
Points number:	**100**	Submit before :	**2023-04-18**
Presentation :	**2023-04-18 and 20**	Session :	**Winter 2023**

STATEMENT OF THE COMPETENCY

- To use object-oriented development approach (016T)

REQUIRED COMPETENCIES FOR THE PROJECT

1. To create an object model
2. Refine the object model.
3. To program a class
4. To ensure that the class functions correctly

DIRECTIVES

- **Open book and notes.**

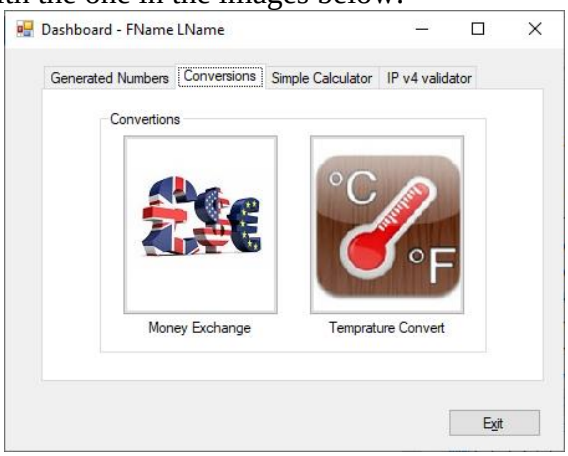
INSTRUCTIONS
This project has 7 sections

The application has 7 Forms, and 1 document evaluated like this:

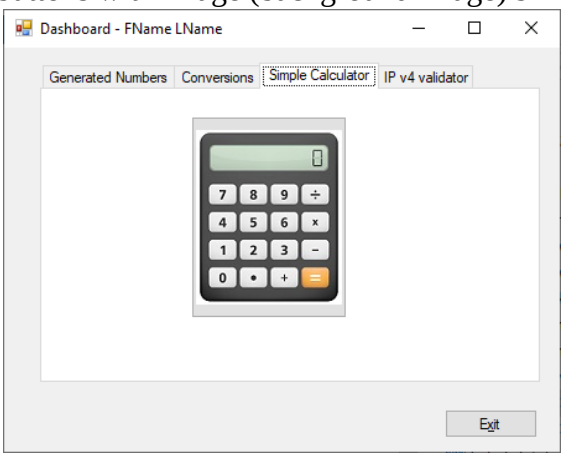
Form 0 Dashboard	**Form 1+2** Lotto Max + Lotto 649	**Form 3 + 4** Money Conversion + Temperature Conversion	**Form 5** Simple Calculator	**Form 6** IP4 Validator	**Documents** Technical and user document	**Total**
5 points	**20 points**	**15+15points**	**20points**	**10points**	**15 points**	**100 p**

Form 0 - The Dashboard (5 points)

Create a C# Form **frmDashboard**, similar with the one in the images below:



Each tab give access to diferent form applications. Use the provide images and create buttons with image (background image) similar with the images on this page.



The Exit button will confirm before exiting the application.

Identify yourself, enter the current date and have a short description for the application as a comment. By clicking on one of those buttons the application will open one of the following form applications:

Form 1 & 2 - The Lotto Max and 649 applications (20 points)

The form application that allows the user to generate 7 + 1 (frmMax) or 6 + 1 (frm649) random unique numbers for the Lotto Max (1-50) or for the Lotto 649 (1-49).

This is the code that generate random number from 1 to 49:

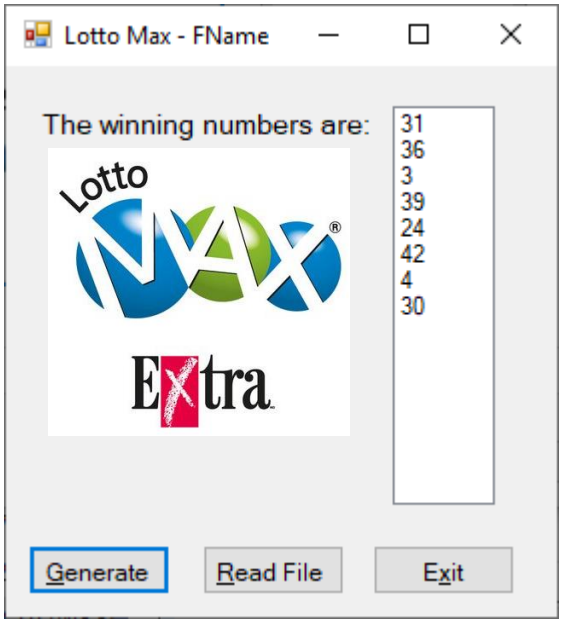
```
Random random = new Random();
int randomNumber = random.Next(1, 49); //random.Next(1, 50); for MAX
```

When **Generate** button is click, the 8 or 7 **unique** numbers should be generated and presented to the user into a multiline read only TextBox and save those numbers into a text file **LottoNbrs.txt** (one series of numbers per line, using comma to separate the numbers – similar like in the example below). Identify the name of the lottery, add the current date and time.

Example:

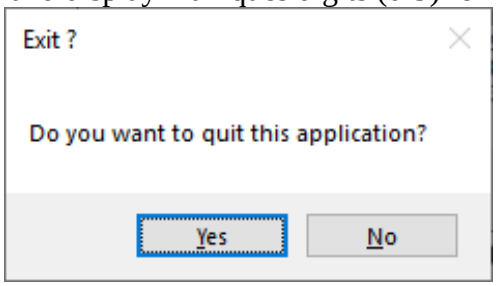
Max, 2023/3/20 2:17:36 PM, 31,36,3,39,24,42,4 Bonus 30
649, 2023/3/20 2:17:42 PM, 29,45,42,22,41,18 Bonus 11

Read and show the text file content using a message box when the button **Read File** is click. Add similar data information into the message box, add the appropriate title + your name as title. Use: FileStream, StreamReader, StreamWriter objects



7 + 1 = 8 unique numbers (1-50)

6 + 1 = 7 unique numbers (1-49), also generate and display 7 uniques digits (0-9) for each lottery, display them under the image



(Example of Message box for Exit)

Test the applications to be sure are displayed only unique numbers.

Form 3 - The Money Exchange application (15 points)

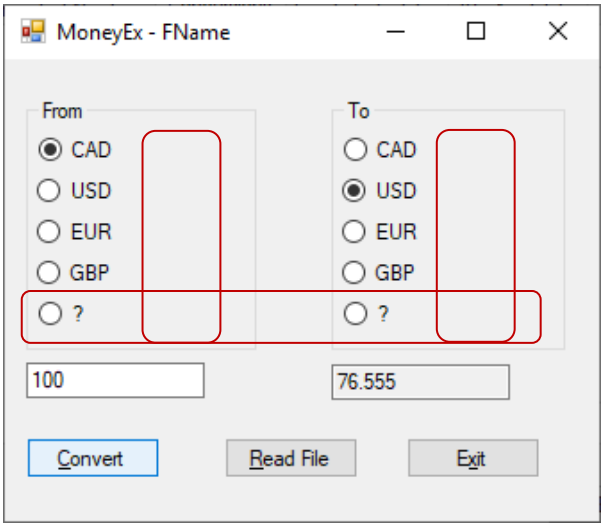
Add another currency of your choice, use the exchange factor, you find it on the Internet (save it as a comment + the date).

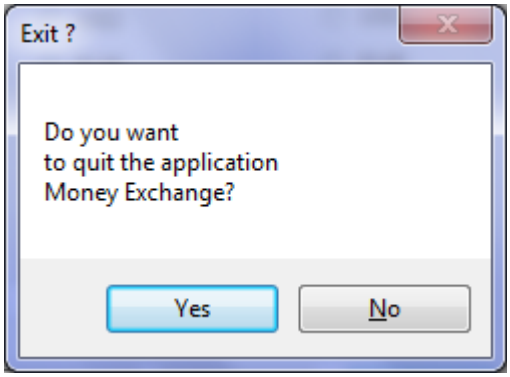
The choice of 4 + 1 exchange currencies should be presented by using radio buttons and allow the user to enter a valid amount into a text box (left), and presenting the converted amount into the read only TextBox (right).

Add another country of your choice on the fifth position. Search and add an flag image on the right side for each country.

Save the conversion into a text file **MoneyConv.txt** (one conversion per line. Identify the currency from/ to and add the current date and time.
Example:

100 CAD = 76 USD, 2022/7/29 9:57:33 AM
100 EUR = 85 USD, 2022/7/29 10:07:03 AM





Use: Try and Catch, Regular Expressions and FileStream, StreamReader, StreamWriter objects.

Read and show the text file content using a message box. Add pertinent information into the message box and add the appropriate title + your name as title.

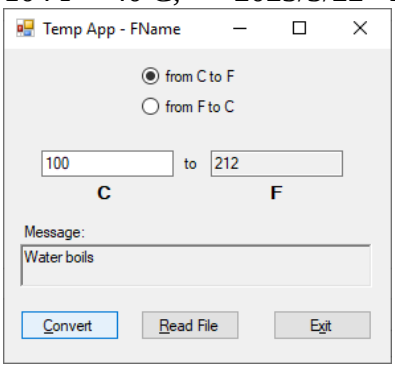
Before closing the form, calculate, and display the total time in seconds and minutes, the user was using that form (take the time when the form was loaded and the time of closing)

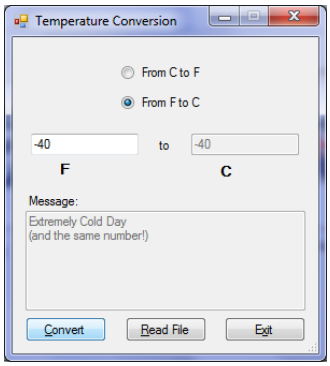
Form 4 - The Temperature application (15 points)

The choice of 2 possible temperature conversions should be presented to the user by using radio buttons and allow him to enter the valid value into a text box, and presenting the corresponding temperature into the read only TextBox. Into the Message read only text box, write the appropriate message description (see the table on the next page). Also save the temperature conversion into a text file **TempConv.txt** (one conversion per line. Identify the conversion from/ to and add the current date and time.

Example:

100 C = 212 F, 2023/3/22 01:01:33 PM Water Boils
104 F = 40 C, 2023/3/22 10:07:03 PM Hot Bath





Read and show the text file content using a message box. Add pertinent information into the message box and add the appropriate title + your name as title.

Use try and catch to validate data and show clear messages to the user.

Formula for temperature conversion:

Celsius to Fahrenheit: $(°C \times {}^{9}/_{5}) + 32 = °F$

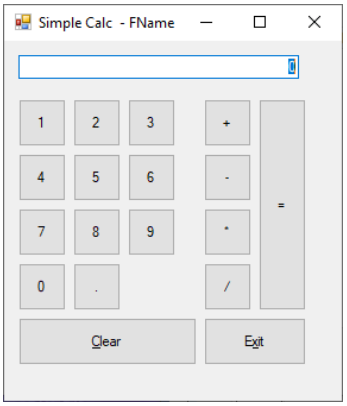
Fahrenheit to Celsius: $(°F - 32) \times {}^{5}/_{9} = °C$

Examples:

°C	°F	Message description
100	**212**	Water boils
40	**104**	Hot Bath
37	**98.6**	Body temperature
30	**86**	Beach weather
21	70	Room temperature
10	**50**	Cool Day
0	**32**	Freezing point of water
−18	0	Very Cold Day
−40	**−40**	Extremely Cold Day (and the same number!)
(**bold** are exact)		

Use: Try and Catch, Regular Expressions and FileStream, StreamReader, StreamWriter objects. Change the color for different temperatures (red for hot, green for comfortable temperatures, and blue for cold)

Form 5 - Simple calculator application (20 points)



You'll design a form that lets the user perform the operations provided by a basic calculator. You'll also create a class that performs the required basic operation and presenting the result into the read only TextBox. Also save all the arithmetical operations and theirs results into a text file **Calculator.txt** (one operation per line). Example: 3 + 5 = 8

You must convert entry data back and fore between decimal or double (data type for calculations) to the string data type (for presentation).

By clicking on one button from 0 to 9, you want to display the first or the second number into the read only text box, and, when you click on one of those 4 operation buttons, you need to have that value as a number to be used in further calculations, then, when the equal button is pressed, the result of the operation will be displayed. Clear button will clear all the fields' members of the Calculator class object, and the text box.

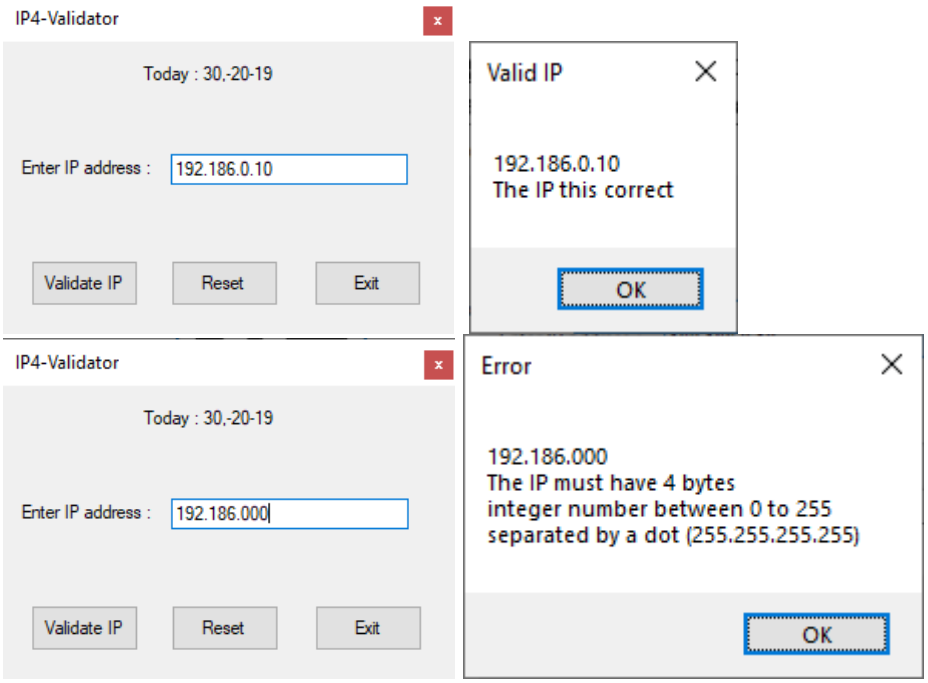
- If you want to be more specific about the Calculator class, you can provide this class design:

Private field	Description
currentValue	A decimal that stores the result currently displayed by the calculator.
operand1	A decimal that stores the value of the first operand.
operand2	A decimal that stores the value of the second operand.
op	A string type that stores the value of the operator
Constructor	**Description**
()	Creates a Calculator object with default values. The default value for the op field is Null.
Property	**Description**
CurrentValue	Gets the value of the currentValue field.
Method	**Description**
Add(displayValue**)**	Sets the operand1 and currentValue fields to the value that's passed to it and sets the op field to "+".
Subtract(displayValue**)**	Sets the operand1 and currentValue fields to the value that's passed to it and sets the op field to "-".
Multiply(displayValue**)**	Sets the operand1 and currentValue fields to the value that's passed to it and sets the op field to "*".
Divide(displayValue**)**	Sets the operand1 and currentValue fields to the value that's passed to it and sets the op field to "/".
Equals()	Performs the operation specified by the op field on the operand1 and operand2 fields, and stores the result in the operand1 field.
Equals(displayValue**)**	Sets the operand2 field to the value that's passed to it. Then, performs the operation specified by the op field on the operand1 and operand2 fields, and stores the result in the operand1 field.
Clear()	Sets the private fields to their default values.

Use: FileStream, StreamReader, StreamWriter objects

Form 6 - The IP4 Validator application (10 points)

Create and add to a new windows form with Visual Studio C#, name it **IP4-Validator.** Search for an appropriate image and create another button into the dash board (form 0). You'll design a form that lets the user perform the following operations using appropriate **String** and **DateTime** object methods (ToLongDateString(), Trim(),Split()), to create a similar form with the one in the following image:



You must present the current date in long format once the Form application open.
If the IP4 is a valid address, write it into a binary file, followed by the date and time.
Use: Try and Catch, Regular Expressions and FileStream and BinaryWriter objects.

Before closing the form, calculate, and display the total time in seconds and minutes, the user was using that form (take the time when the form was loaded and the time of closing)

Create a document using the template, with the print screens of your form applications, also present all the classes, the methods, that you create and use to build this application.(**15 points**)

Test your application functionalities with different sets of data, save the solution, compressit and send it with the results text files by LEA of Omnivox.

Thank you.