Course number : 420-CT2-AS

ORIENTED OBJECT PROGRAMMING

Teacher : Maftei Mihai

Weighting :	**30%**	Group :	**07224**
Points number:	**100**	Date :	**2020-04-17 before 14:00**
Duration :		Session :	**Winter 2020**

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.**

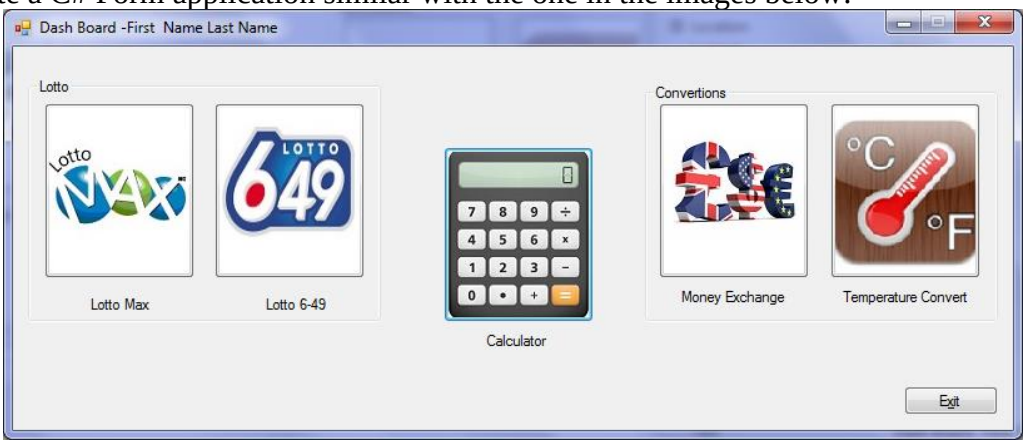
INSTRUCTIONS
This project has 6 sections

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

Form 0 Dash Board	**Form 1+2** Lotto Max + Lotto 649	**Form 3** Simple Calculator	**Form 4** Money Conversion	**Form 5 + 6** Temperature Conversion + IP4 Validator	**Document** Describing the application	**Total**
5 points	**20 points**	**20points**	**20points**	**10 + 10points**	**15 points**	**100 p**

Form 0 - The Dash board (5 points)

Create a C# Form application similar with the one in the images below:



The Exit button will confirm before exiting.

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:

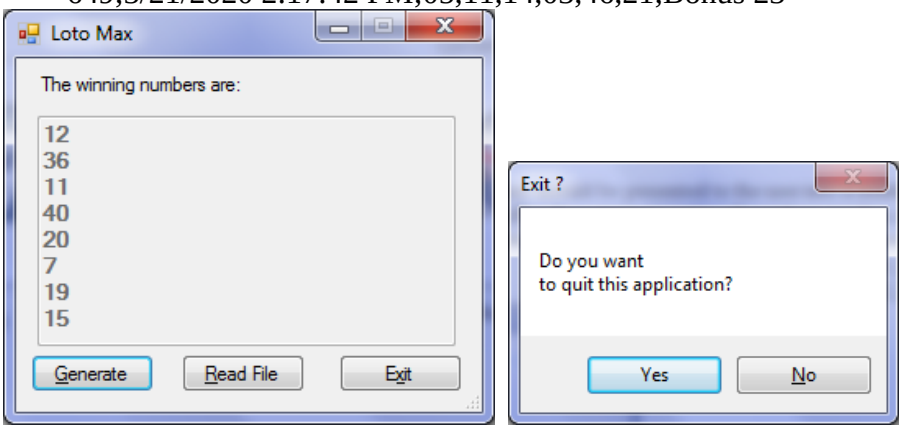
Form 1 and 2 - The Lotto Max and 649 applications (10 + 10 points)

Create a class **Lotto**, with overloaded constructors for this application. The form application allows the user to generate 7 + 1 random numbers out of 50 for the game Lotto **Max,**.and 6 + 1 random numbers out of 49 for the game Lotto **649**.

```
Random random = new Random();
int randomNumber = random.Next(1, 49);
```

The 8 or 7 unique numbers should be presented to the user into a multiline read only TextBox and also saved into a text file **LottoNbrs.txt**. Identify the name of the lottery and add the current date and time, followed by the numbers (separate them by semi column).

Ex.: Max;3/21/2020 2:17:36 PM;40,33,12,06,36,04,18;Bonus 29
649;3/21/2020 2:17:42 PM;05,11,14,03,46,21;Bonus 23



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. You can display the file content into text boxes (up to 50 lines/box).

Use: FileStream, StreamReader, StreamWriter objects

Form 3 - 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).
Ex.: You have to convert entry data back and fore between decimal or double (data type for calculations) to the string data type (for presentation).
So, by clicking on one button from 0 to 9, you want to present the first or the second number into the text box, and also, 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, and 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, and the text box.

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

Private field	Description
currentVal	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	An 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**
CurrentVal	Gets the value of the currentVal field.
Method	**Description**
Add(displayVal**)**	Sets the operand1 and currentVal fields to the value that's passed to it, and sets the op field to "+".
Subtract(displayVal**)**	Sets the operand1 and currentVal fields to the value that's passed to it, and sets the op field to "-".
Multiply(displayVal**)**	Sets the operand1 and currentVal fields to the value that's passed to it, and sets the op field to "*".
Divide(displayVal**)**	Sets the operand1 and currentVal 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(displayVal**)**	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.

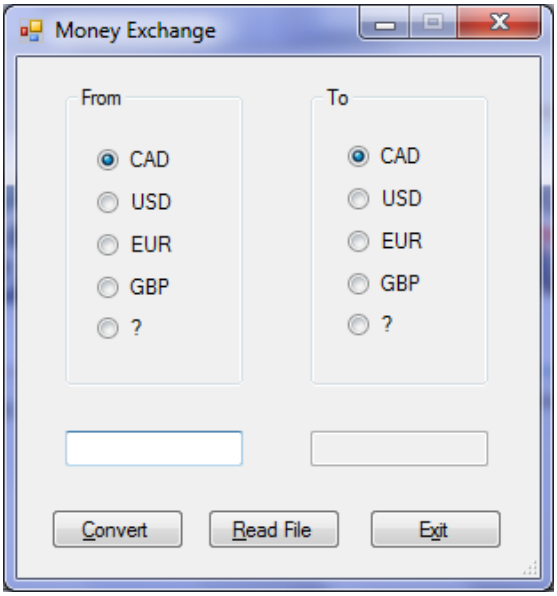
Use: FileStream, StreamReader, StreamWriter objects

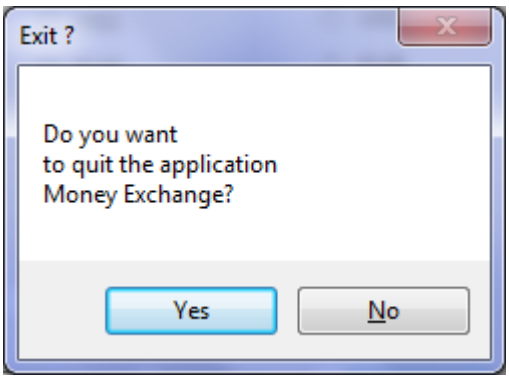
Form 4 - The Money Exchange application (20 points)

Create a class **Exchange** with constructors and methods that allow you to calculate and display the results for possible conversions between five different currencies. Add another currency of your choice. 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).

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

Ex.: 100 CAD = 78 USD, 4/01/2020 9:57:33 AM
100 EUR = 88 USD, 4/01/2020 10:07:03 AM





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

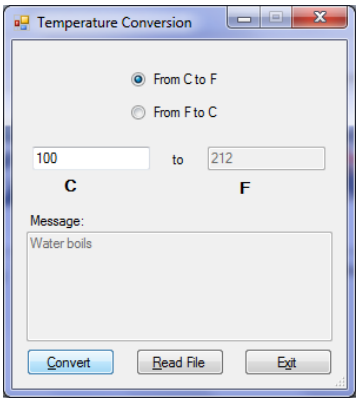
Read and show the text file content using a message box. Add pertinent information into the message box and add the appropriate title + your name as title. Make sure that you can display the file content into text boxes (up to 30 lines).

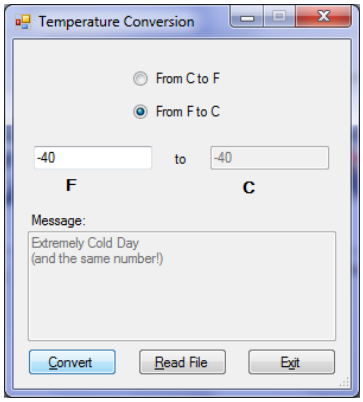
Form 5 - The Temperature application (10 points)

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) for the calculated temperatures. Save the temperature conversion into a text file **TempConvert.txt** (one conversion per line. Identify the conversion from/ to and add the current date and time.

Ex.: 100 C = 212 F, 4/02/2020 01:01:33 PM
104 F = 40 C, 4/02/2020 10:07:03 PM





Read and show the text file content using a message box if the file is not empty. Add pertinent information into the message box and add the appropriate title + your name as title. Change the color of the text depending the temperature. 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: $(°F - 32) \times {}^{5}/_{9} = °C$

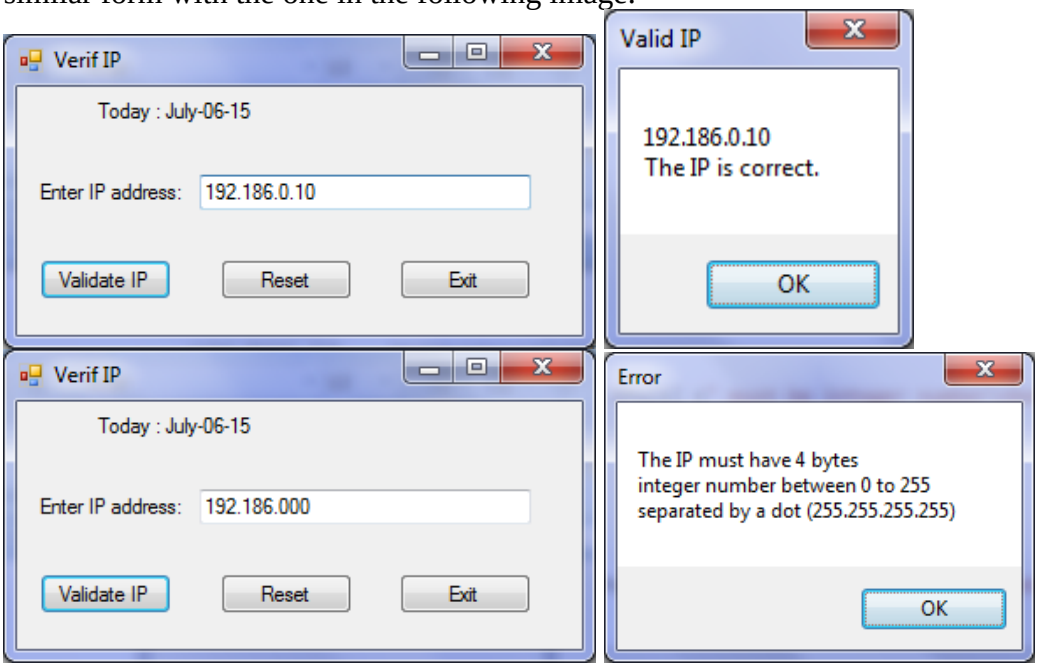
Examples:

°C	°F	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. You can display the file content into text boxes (up to 20 lines/box).

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 have to present the current date in long format once the Form application open. Save the date and the correct IPv4 addresses into a file. Create a button to display the file content into text boxes (up to 40 lines/box).

Create a document with print screens of your application, also present all the classes , and methods, that you use to build this application.(**10 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.