Course number : 420-CT2-AS

ORIENTED OBJECT PROGRAMMING

Teacher : Maftei Mihai

Weighting Section1:	**10% out of 30%**	Group :	**07222**
Points number:	**100**	Submit before :	**2023-11-10**
Weighting Section2:	**10% out of 30%**	Group :	**07222**
Points number:	**100**	Submit before :	**2023-11-24**
Weighting Section3:	**10% out of 30%**	Group :	**07222**
Points number:	**100**	Submit before :	**2023-12-08**
Presentation :	**2023-12-08 or before**	Session :	**Fall 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

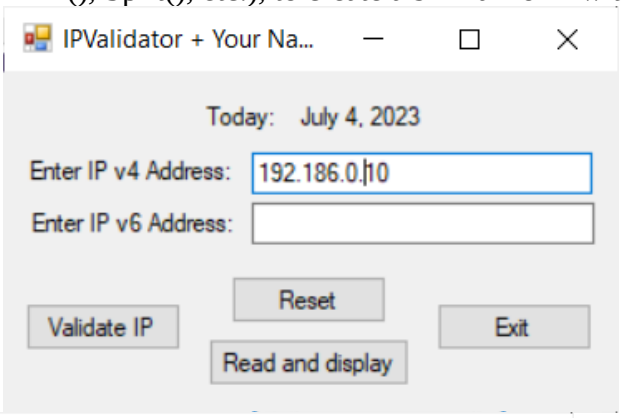
This project has 7 sections.

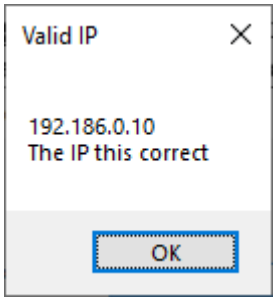
The application has 7 Forms, and 1 document evaluated like this:

Form 1 Dash Board	**Form 2 & 3** Lotto Max + Lotto 649	**Form 4 & 5** Money Conversion + Temperature Conversion	**Form 6** IPv4 & IPv6 Validator	**Form 7** Simple Calculator	**Docs** Technical and user document	**Presentation** **2023-12-08/11** (10/15 min.)	**Total**
5 points	**20 points**	**10+10points**	**10points**	**15points**	**15 points**	**15 points**	**100 p**

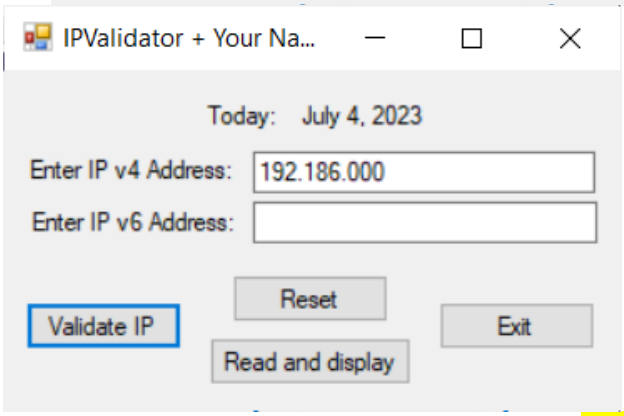
Section 3 - Form 6 - The IP4 & IP6 Validator application (10 points)

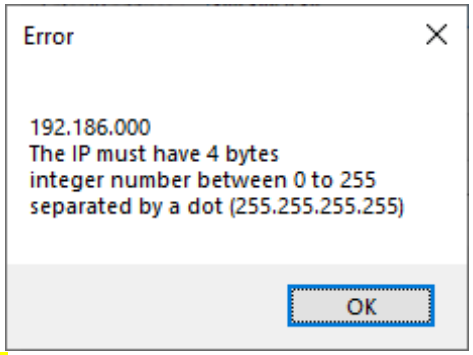
- Create and add to a new windows form with Visual Studio C#, name it **IP-Validator.** Use an appropriate image and create the button into the dashboard
- You'll design a form that lets the user perform the following operations using appropriate **String, Regex** and **DateTime** object methods (ToLongDateString(),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





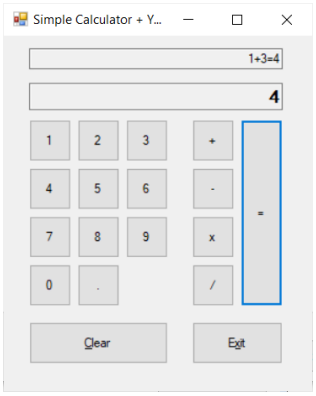
-





- Once you did the design, upload it as the first version of Project Section3.
- You must present the current date in long format once the Form application open.
- If the IP4/IP6 is a valid address (use abstraction and overriding), write it into a binary file, followed by the date and time, if not, an error message will be displayed (the right format for IP4 or IP6).
  Use: Try and Catch, Regular Expressions and FileStream and BinaryWriter, BinaryReader classes. Upload it as the 2nd version of Project Section3.

Section 3 -Form 7 - Simple calculator application



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) and **Calculator.xml**. Example: 1 + 3 = 4

<operation>1 + 3 = 4</operation>

Use abstraction and overriding. 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. Once you did the design, upload it as the 3rd version of Project Section3.

- 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, BinaryWriter, BinaryReader, XmlWriter, and XmlReader objects and methods

- Upload (zip) before 8-Dec – Section3 + the document Technical - User manual

The schedule for submiting your project:

- before 10-Nov – Section1
- before 24 Nov – Secton2
- before 08 Dec – Section3

Thank you.