|  |
| --- |
| College LaSalle |
| Project - Oriented Object Programming User and Technical Manual |
|  |
| Presented to: Mihai Maftei. |

|  |
| --- |
| Bonnie Ives de Castro Nunes  3/26/2023  Version: 2.0 |

1. **Start by adding a short description of your project, and the languages (technologies) used:**
2. Language: The code is made using C# language.
3. Tools (IDE): The tool used is Visual Studio 2022 with the Windows Forms App (.NET Framework).
4. **Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).**

## Number Generator Tag

1. If you click on tag Numbers Generator it will be shown the buttons for Lotto Max and Lotto 649 (Figure 1).

Graphical user interface, application

Description automatically generated

1. If you click Exit it will open a window “Exit?” which ask you if you want to exit the application. If you click on “Yes”, it will close the application; if you click on “No”, it will close only the “Exit?” window.

Graphical user interface, text, application, website

Description automatically generated

1. If you click on Lotto Max button, it will open a new window.

Graphical user interface, application

Description automatically generated

1. If you click on Generate, it will generate eight numbers in the text box and seven numbers below the picture.

Graphical user interface, application

Description automatically generated

1. If you click on Read file button, it will open a window with all the numbers from Lotto Max stored on the file LottoNmbr.txt. If you click on OK, it closes the window.

Text

Description automatically generated

1. If you click on Exit, it will execute the same procedure as in number 2.
2. If you are back to the same window in number 1 and click on Lotto 649 button, it will open new window.

Graphical user interface, application

Description automatically generated

1. If you click on Generate, it will generate seven numbers in the text box and seven numbers below the picture.

Graphical user interface, application

Description automatically generated

1. If you click on Read file button, it will open a window with all the numbers from Lotto 649 stored on the file LottoNmbr.txt. If you click on OK, it closes the window.

Text

Description automatically generated

1. If you click on Exit, it will execute the same procedure as in number 2.

## Convertors Tab

1. If you select the tab Convertors, the screen below will be shown.

Graphical user interface, application

Description automatically generated

1. If you click Money Exchange, a new window will open, as below.

Graphical user interface, application

Description automatically generated

1. If you type a number in the text box and select two different currencies (From and To), and click convert, you will have the value converted shown on the second text box.

Graphical user interface, application

Description automatically generated

1. If you click on Read File, a new window will open and will show you the register log of all conversions performed before.

Table

Description automatically generated with low confidence

1. If you click Exit, it will show the window below.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. If you click yes, it will show a window with the elapsed time on the application.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. Now, if you are back to the Convertors tab and click on Temperature Convert Button, the following window will pop up.

Graphical user interface, application

Description automatically generated

1. If you type a temperature and choose from C to F, it will perform a conversion from Celsius to Fahrenheit and show a message.

Graphical user interface, text, application

Description automatically generated

1. If you select from F to C, the labels below the textbox will switch

Graphical user interface, application

Description automatically generated

1. If you click on Read File a window will pop up with the register log of all conversions performed before.

Table

Description automatically generated

1. If you click exit, the same window from number 5 will be shown.

## Calculator

1. If you select the tab Calculator, you will be presented with the following design.

Graphical user interface, application

Description automatically generated

1. If you click on the calculator, a window will pop up showing you the calculator.

A screenshot of a computer

Description automatically generated with low confidence

1. If you click on the numbers, it will type each one individually (you can’t type directly on the text box).

A screenshot of a computer

Description automatically generated with medium confidence

1. If you select an operation, it will focus on the text box and reset to 0 the number on it.

A screenshot of a cell phone

Description automatically generated with medium confidence

1. If you type other numbers and click on “=”, it will show you the result of the operation.

A screenshot of a computer

Description automatically generated with low confidence

1. If you click on clear, the numbers will be reset to 0 and the result will be erased from the memory.
2. If you click on exit, the following window will pop up.

Graphical user interface, text, application, website

Description automatically generated

1. If you click on Yes, the calculator will be closed.

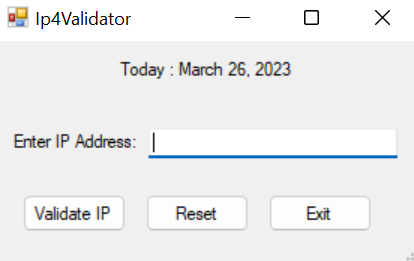
## IPv4 Validator

1. If you select the tab IPv4 Validator, you will have the option to this application.

Graphical user interface, application

Description automatically generated

1. If you click on the button IP, the windows below will pop up.



1. If you type the a right IP address into the text box and validate it you will have a pop up window showing the IP informed, the class of the IP and a message of validation.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. If you inform an invalid IP, it will pop up a window showing the IP informed and a message suggesting the right format of the IP address.

Graphical user interface, text, application

Description automatically generated

1. If you click on Reset, the text box will be cleared.
2. If you click on Exit, a window will pop up to confirm if the user wants to close the application.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. If the user chose on No, they return to the application.
2. If the user chose yes, other window pops up and shows the elapsed time using this application.

Graphical user interface, text, application, chat or text message

Description automatically generated

1. **Present the code of your application (forms).**

## Code for the main form

public partial class mainForm : Form

{

public mainForm()

{

InitializeComponent();

}

// When you click the button for Lotto Max, it will open a new window

// to generate the Lotto Max numbers

private void lottoMaxBtn\_Click(object sender, EventArgs e)

{

lottoMax lottoMax = new lottoMax();

lottoMax.ShowDialog();

}

// When you click the button for Lotto 649, it will open a new window

// to generate the Lotto Max numbers

private void lottoExtraBtn\_Click(object sender, EventArgs e)

{

lotto649 lotto649 = new lotto649();

lotto649.ShowDialog();

}

// This will be activated when you click the button for

// Money Exchange, opening a new window for it

private void moneyExchange\_Click(object sender, EventArgs e)

{

MoneyExchange moneyExchange = new MoneyExchange();

moneyExchange.ShowDialog();

}

// This will be activated when you click the button for

// Temperature Conversion, opening a new window for it

private void tempConvert\_Click(object sender, EventArgs e)

{

TempConv tempConv = new TempConv();

tempConv.ShowDialog();

}

private void button6\_Click(object sender, EventArgs e)

{

Calculator calculator = new Calculator();

calculator.ShowDialog();

}

private void ipVerifier\_Click(object sender, EventArgs e)

{

Ip4Validator ip4Validator = new Ip4Validator();

ip4Validator.ShowDialog();

}

private void exitBtn\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit all applications?",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

}

## Code for Lotto Max

public partial class lottoMax : Form

{

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream fileStream;

StreamWriter textOut;

StreamReader textIn;

// Declaring the structure to be used for showing the dates

DateTime date;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "LottoNbrs.txt";

// Declaring the random number

Random random;

// Declaring an array to store random numbers for print

// into the number box

int[] randomNumberBox;

public lottoMax()

{

InitializeComponent();

}

// ----------- WORKING WITH THE TEXT BOX GENERATING NUMBERS BETWEEN

// ----------- 1 AND 50

private void generateMax\_Click(object sender, EventArgs e)

{

// Starting to instantiating and declaring the variables for the numbers

random = new Random();

randomNumberBox = new int[8];

// Declaring the objects to manipulate the files

string filePath = pathDir + file;

fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write);

textOut = new StreamWriter(fileStream);

// Declaring the object to manipulate the dates

date = DateTime.Now;

// ------------ BLOCK FOR TEXT BOX ---------------------

// We must clear the text box for each iteration

textBoxMax.Clear();

// This first loop is used to generate eight numbers that

// will be printed into the text box

for (int i = 0; i < 8; i++)

{

// The number is generated and stored into the array

randomNumberBox[i] = random.Next(1, 50);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < i; j++)

{

// If the random number into the verified index is equal

// to the number generated, it will generate another

// random number until it doesn't repeat anymore

while (randomNumberBox[j] == randomNumberBox[i])

{

randomNumberBox[i] = random.Next(1, 50);

// If it is repeated, it will restart the counter

// into the for loop so it can check it again

j = 0;

}

}

// It will print into the label2 the concatenated numbers

textBoxMax.Text += randomNumberBox[i].ToString();

//textBox1.AppendText(randomNumberBox[i].ToString());

textBoxMax.AppendText(Environment.NewLine);

}

// ------------ BLOCK FOR LABEL ---------------------------

// We must clear the label for each iteration

labelMax7.Text = null;

// randomNumber is an array of 7 items wich will store

// the random number generated.

int[] randomNumber = new int[7];

// This for loop is used to generate seven random numbers

// and to display into the label

for (int i = 0; i < 7; i++)

{

// The number is generated and stored into the array

randomNumber[i] = random.Next(0, 7);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < i; j++)

{

randomNumber[i] = random.Next(0, 7);

}

// It will print into the label2 the concatenated numbers

labelMax7.Text += randomNumber[i].ToString() + " ";

}

// ---------------- BLOCK FOR FILES ----------------------------

// Writing into the the file

textOut.Write("Max, " + date.ToString(@"yyyy'/'M'/'dd - hh:mm:ss tt, "));

// This next loop is used to store the information inside

// the file

for (int i = 0; i < randomNumberBox.Length - 1; i++)

{

// Storing the numbers into the file

textOut.Write(randomNumberBox[i].ToString());

if (i != randomNumberBox.Length - 2)

{

textOut.Write(", ");

}

}

// Now it prints the Bonus number, which is the last of the

// serie randomNumber

textOut.Write(" Bonus " + randomNumberBox[randomNumberBox.Length - 1] + "\n");

// Closing the file

textOut.Close();

}

// Activated when read file is clicked

private void readFileMax\_Click(object sender, EventArgs e)

{

// Instantiating the object textIn which readt the file

// in the indicated path

string filePath = pathDir + file;

try

{

textIn = new StreamReader(filePath);

// Declaring an empty string list to store the read text

// in the file

string list = "";

// Setting the counter so the text box will display

// five registers per window

int counter = 0;

// Starting the loop and stoping when the index is equal to -1

while (textIn.Peek() != -1)

{

string row = textIn.ReadLine();

// When the string starts with Max, it will store into the string

// list all the row

if (row.StartsWith("Max") == true)

{

list += row.ToString() + "\n";

// Incrementing the counter

counter++;

}

if (counter == 5)

{

// The list will be shown into a new window

MessageBox.Show(list, "Numbers log - Lotto Max");

counter = 0;

list = "";

}

}

if (counter != 0)

{

// The list will be shown into a new window

MessageBox.Show(list, "Numbers log - Lotto Max");

}

textIn.Close();

}

catch (FileNotFoundException)

{

MessageBox.Show("File not created.");

File.Create(filePath).Close();

}

}

// Closes the application

private void exitMax\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application? ",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void lottoMax\_Load(object sender, EventArgs e)

{

string filePath = pathDir + file;

if (!Directory.Exists(pathDir))

{

Directory.CreateDirectory(pathDir);

}

}

}

## Code for Lotto 649

public partial class lotto649 : Form

{

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream fileStream;

StreamWriter textOut;

StreamReader textIn;

// Declaring the structure to be used for showing the dates

DateTime date;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "LottoNbrs.txt";

// Declaring the random number

Random random;

// Declaring an array to store random numbers for print

// into the number box

int[] randomNumberBox;

public lotto649()

{

InitializeComponent();

}

// ----------- WORKING WITH THE TEXT BOX GENERATING NUMBERS BETWEEN

// ----------- 1 AND 50

private void generateMax\_Click(object sender, EventArgs e)

{

// Starting to instantiating and declaring the variables for the numbers

random = new Random();

randomNumberBox = new int[7];

// Declaring the objects to manipulate the files

string filePath = pathDir + file;

fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write);

textOut = new StreamWriter(fileStream);

// Declaring the object to manipulate the dates

date = DateTime.Now;

// ------------ BLOCK FOR TEXT BOX ---------------------

// We must clear the text box for each iteration

textBox649.Clear();

// This first loop is used to generate eight numbers that

// will be printed into the text box

for (int i = 0; i < randomNumberBox.Length; i++)

{

// The number is generated and stored into the array

randomNumberBox[i] = random.Next(1, 50);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < i; j++)

{

// If the random number into the verified index is equal

// to the number generated, it will generate another

// random number until it doesn't repeat anymore

while (randomNumberBox[j] == randomNumberBox[i])

{

randomNumberBox[i] = random.Next(1, 50);

// If it is repeated, it will restart the counter

// into the for loop so it can check it again

j = 0;

}

}

// It will print into the label2 the concatenated numbers

textBox649.Text += randomNumberBox[i].ToString();

//textBox1.AppendText(randomNumberBox[i].ToString());

textBox649.AppendText(Environment.NewLine);

}

// ------------ BLOCK FOR LABEL ---------------------------

// We must clear the label for each iteration

label649\_7.Text = null;

// randomNumber is an array of 7 items wich will store

// the random number generated.

int[] randomNumber = new int[7];

// This for loop is used to generate seven random numbers

// and to display into the label

for (int i = 0; i < 7; i++)

{

// The number is generated and stored into the array

randomNumber[i] = random.Next(0, 7);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < i; j++)

{

randomNumber[i] = random.Next(0, 7);

}

// It will print into the label2 the concatenated numbers

label649\_7.Text += randomNumber[i].ToString() + " ";

}

// ---------------- BLOCK FOR FILES ----------------------------

// Writing into the the file

textOut.Write("649, " + date.ToString(@"yyyy'/'M'/'dd - hh:mm:ss tt, "));

// This next loop is used to store the information inside

// the file

for (int i = 0; i < randomNumberBox.Length - 1; i++)

{

// Storing the numbers into the file

textOut.Write(randomNumberBox[i].ToString());

if (i != randomNumberBox.Length - 2)

{

textOut.Write(", ");

}

}

// Now it prints the Bonus number, which is the last of the

// serie randomNumber

textOut.Write(" Bonus " + randomNumberBox[randomNumberBox.Length - 1] + "\n");

// Closing the file

textOut.Close();

}

// Activated when read file is clicked

private void readFile649\_Click(object sender, EventArgs e)

{

// Instantiating the object textIn which readt the file

// in the indicated path

string filePath = pathDir + file;

try

{

textIn = new StreamReader(filePath);

// Declaring an empty string list to store the read text

// in the file

string list = "";

// Setting the counter so the text box will display

// five registers per window

int counter = 0;

// Starting the loop and stoping when the index is equal to -1

while (textIn.Peek() != -1)

{

string row = textIn.ReadLine();

// When the string starts with Max, it will store into the string

// list all the row

if (row.StartsWith("649") == true)

{

list += row.ToString() + "\n";

// Incrementing the counter

counter++;

}

if (counter == 5)

{

// The list will be shown into a new window

MessageBox.Show(list, "Numbers log - Lotto Max");

counter = 0;

list = "";

}

}

if (counter != 0)

{

// The list will be shown into a new window

MessageBox.Show(list, "Numbers log - Lotto Max");

}

textIn.Close();

}

catch (FileNotFoundException)

{

MessageBox.Show("File not created.");

File.Create(filePath).Close();

}

}

// Closes the application

private void exit649\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application? ",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void lotto649\_Load(object sender, EventArgs e)

{

string filePath = pathDir + file;

if (!Directory.Exists(pathDir))

{

Directory.CreateDirectory(pathDir);

}

}

}

## Code for Lotto Max

public partial class lottoMax : Form

{

public lottoMax()

{

InitializeComponent();

}

private void button1\_Click(object sender, EventArgs e)

{

// ----------- WORKING WITH THE TEXT BOX GENERATING NUMBERS BETWEEN

// ----------- 1 AND 50

textBox1.Text = "";

// Declaring a file to store the information of the

// numbers. The bolean true is used to not overwrite

// the file instead of deleting it

StreamWriter file = new StreamWriter("LottoNbrs.txt", true);

// Storing some text and writing the actual date time

file.Write("Max, ");

file.Write(DateTime.Now.ToString("yyyy'/'M'/'dd \t hh:mm:ss tt") + ", ");

// For this first declaration, we are declaring the object

// random to generate the random numbers

Random random = new Random();

// randomNumber is an array of 7 items wich will store

// the random number generated.

int[] randomNumberBox = new int[8];

// This for loop is used to generate seven random numbers

// and to display into the label

for (int i = 0; i < 8; i++)

{

// The number is generated and stored into the array

randomNumberBox[i] = random.Next(1, 50);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < i; j++)

{

// If the random number into the verified index is equal

// to the number generated, it will generate another

// random number until it doesn't repeat anymore

while (randomNumberBox[j] == randomNumberBox[i])

{

randomNumberBox[i] = random.Next(1, 50);

// If it is repeated, it will restart the counter

// into the for loop so it can check it again

j = 0;

}

}

// It will print into the label2 the concatenated numbers

textBox1.Text += randomNumberBox[i].ToString();

//textBox1.AppendText(randomNumberBox[i].ToString());

textBox1.AppendText(Environment.NewLine);

}

// This next loop is used to store the information inside

// the file

for (int i = 0; i < randomNumberBox.Length - 1; i++)

{

// Storing the numbers into the file

file.Write(randomNumberBox[i].ToString());

if (i != randomNumberBox.Length - 2)

{

file.Write(", ");

}

}

// Now it prints the Bonus number, which is the last of the

// serie randomNumber

file.Write(" Bonus " + randomNumberBox[randomNumberBox.Length - 1] + "\n");

// ----------- WORKING WITH THE LABEL GENERATING NUMBERS BETWEEN

// ----------- 0 AND 9

label2.Text = "";

// randomNumber is an array of 7 items wich will store

// the random number generated.

int[] randomNumber = new int[7];

// This for loop is used to generate seven random numbers

// and to display into the label

for (int k = 0; k < 7; k++)

{

// The number is generated and stored into the array

randomNumber[k] = random.Next(0, 7);

// This for loop pass through the array from the last index

// to the first and checks if there is a repeated number

for (int j = 0; j < k; j++)

{

randomNumber[k] = random.Next(0, 7);

}

// It will print into the label2 the concatenated numbers

label2.Text += randomNumber[k].ToString() + " ";

}

// Close the file

file.Close();

file.Dispose();

}

private void textBox1\_TextChanged(object sender, EventArgs e)

{

}

private void button3\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application? ",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void button2\_Click(object sender, EventArgs e)

{

StreamReader reader = new StreamReader("LottoNbrs.txt");

string line = "";

string list = "";

while ((line = reader.ReadLine()) != null)

{

if (line.StartsWith("Max") == true)

{

list += line.ToString() + "\n";

}

}

MessageBox.Show(list, "Numbers log - Bonnie Nunes");

}

}

## Code for Money Exchange

public partial class MoneyExchange : Form

{

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream fileStream;

StreamWriter textOut;

StreamReader textIn;

// Declaring the time when the appliation is opened and

// declaring the objects to manipulate the time.

DateTime timeOpen;

DateTime timeClose;

TimeSpan timeSpan;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "MoneyConv.txt";

public MoneyExchange()

{

InitializeComponent();

}

private void convertMoney\_Click(object sender, EventArgs e)

{

// Declaring the objects to manipulate the file

string filePath = pathDir + file;

fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write);

textOut = new StreamWriter(fileStream);

// The exchanges were consulted on Wednesday 15th March 2023

// The exchanges where normalizated all based on CAD (set as 1)

// then consulted the exchange of 1 CAD to the other currency

// for example, considering 1 CAD we have 0.73 USD

double toCurrency = 0;

double inCurrency = 0;

double amount = 0;

string currencyCodeIn = "";

string currencyCodeOut = "";

double finalAmount = 0;

// Removing all blank spaces

exchangeIn.Text = exchangeIn.Text.Trim().Replace(" ", "").Replace(",", "");

// Verifying the conversion, if an invalid entry is used

// the Exception message will be displayed

try

{

amount = Convert.ToDouble(exchangeIn.Text);

}

catch (Exception exp)

{

MessageBox.Show(exp.Message);

exchangeIn.Focus();

exchangeIn.Text = null;

}

// ------------------- BLOCK TO PERFORM THE EXCHANGE -----------

// Verifying the format of the amount using Regex

// It will acept as much digits as informed, but

// when using decimal numbers it will accept only

// two digits

Regex regex = new Regex(@"^([\d]+)[(\.)]?(\d{2})?$");

// If the number is in the correct format, then this happens

if (regex.IsMatch(amount.ToString()))

{

// This set of conditionals verifies the currency you want to exchange from

// The set of radio buttons on the group From

if (radioFromCAD.Checked == true)

{

inCurrency = 1.00;

currencyCodeIn = "CAD";

}

else if (radioFromUSD.Checked == true)

{

inCurrency = 0.73;

currencyCodeIn = "USD";

}

else if (radioFromEUR.Checked == true)

{

inCurrency = 0.69;

currencyCodeIn = "EUR";

}

else if (radioFromGBP.Checked == true)

{

inCurrency = 0.60;

currencyCodeIn = "GBP";

}

else if (radioFromBRL.Checked == true)

{

inCurrency = 3.84;

currencyCodeIn = "BRL";

}

// This set of conditionals verifies the currency you want to exchange to

// That is, this is the currency you convert to when you click

// on the radio buttons in the group To

if (radioToCAD.Checked == true)

{

toCurrency = 1.00;

currencyCodeOut = "CAD";

}

else if (radioToUSD.Checked == true)

{

toCurrency = 0.73;

currencyCodeOut = "USD";

}

else if (radioToEUR.Checked == true)

{

toCurrency = 0.69;

currencyCodeOut = "EUR";

}

else if (radioToGBP.Checked == true)

{

toCurrency = 0.60;

currencyCodeOut = "GBP";

}

else if (radioToBRL.Checked == true)

{

toCurrency = 3.84;

currencyCodeOut = "BRL";

}

// Calculating the final amount using the formula

// based on proportions

finalAmount = Math.Round((amount \* (toCurrency / inCurrency)), 2);

// Shows the result into the second text box

exchangeOut.Text = finalAmount.ToString();

exchangeIn.Focus();

// ---------------- BLOCK TO WORK WITH THE FILE

// Writing into the file

textOut.Write(amount.ToString() + " " + currencyCodeIn + " = "

+ finalAmount.ToString() + " " + currencyCodeOut + ", " +

DateTime.Now.ToString("yyyy'/'M'/'dd - hh:mm:ss tt") + "\n");

// Closing the file

textOut.Close();

}

else

{

// Message box if the exception happens

// Suggesting the right format to the user

MessageBox.Show("Incorrect format.\nFormat examples: 123, 0.23, 123.30");

exchangeIn.Text = "0";

exchangeIn.Focus();

}

}

// When the application is opened, the time starts to count

private void MoneyExchange\_Load(object sender, EventArgs e)

{

// If the directory do not exists, we create it here

string filePath = pathDir + file;

timeOpen= DateTime.Now;

if (!Directory.Exists(pathDir))

{

Directory.CreateDirectory(pathDir);

}

}

// Method to read the file and to display it content in a window

private void readExchange\_Click(object sender, EventArgs e)

{

// Instantiating the object textIn which readt the file

// in the indicated path

string filePath = pathDir + file;

try

{

textIn = new StreamReader(filePath);

// Declaring an empty string list to store the read text

// in the file

string list = "";

// Setting the counter

int counter = 0;

// Starting the loop and stoping when the index is equal to -1

while (textIn.Peek() != -1)

{

string row = textIn.ReadLine();

list += row.ToString() + "\n";

counter++;

if (counter == 5)

{

// The list will be shown into a new window

MessageBox.Show(list, "Exchange log");

counter = 0;

list = "";

}

}

if (counter != 0)

{

// The list will be shown into a new window

MessageBox.Show(list, "Exchange log");

counter = 0;

list = "";

}

textIn.Close();

}

catch (FileNotFoundException)

{

MessageBox.Show("File not created.");

File.Create(filePath).Close();

}

}

// Closing the application

private void exitExchange\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application:\nMoney Exchange?",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

// The time when closed is stored into timeClose

timeClose = DateTime.Now;

// Time span is the difference between the time when

// closed and the time is open

timeSpan = timeClose - timeOpen;

MessageBox.Show("Elapsed time: " + timeSpan.ToString(@"mm\:ss"), "Session closed");

this.Close();

}

}

}

## Code for Temperature Convertor

public partial class TempConv : Form

{

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream fileStream;

StreamWriter textOut;

StreamReader textIn;

// Declaring the structure to be used for showing the dates

DateTime date;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "TempConv.txt";

// Declaring the temperatures

double temperatureIn;

double temperatureOut;

// Declaring a string to the message

string message;

string unityIn;

string unityOut;

public TempConv()

{

InitializeComponent();

}

private void buttonConvert\_Click(object sender, EventArgs e)

{

// Returning the font to its regular style to remove

// former styles used on this application

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Regular);

// Declaring the objects to manipulate the files

string filePath = pathDir + file;

fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write);

textOut = new StreamWriter(fileStream);

// Reseting the temperatures

temperatureIn = 0;

temperatureOut = 0;

// ------------------ BLOCK TO CONVER FROM C TO F -----------

// Checking the conversions

try

{

// Removing all the blank spaces

textBoxIn.Text = textBoxIn.Text.Trim().Replace(" ", "");

// Checking the conversion from string to double

temperatureIn = Convert.ToDouble(textBoxIn.Text);

}

catch (Exception ex)

{

// If the string is invalid, it will show

// the message declaring the error

MessageBox.Show(ex.Message);

textBoxOut.Text = "";

textBoxOut.Focus();

}

if (radioFromCtoF.Checked == true)

{

// Calulating the temperature that will be displayed

// into the out text box

temperatureOut = Math.Round(((temperatureIn \* ((double)9 / 5)) + 32), 0);

textBoxOut.Text = temperatureOut.ToString();

// Storing the temperature into a string

unityIn = "C";

unityOut = "F";

}

// ------------------ BLOCK TO CONVER FROM F TO C -----------

else if (radioFromFtoC.Checked == true)

{

// Calulating the temperature that will be displayed

// into the out text box

temperatureOut = Math.Round((temperatureIn - 32) \* ((double)5 / 9), 1);

textBoxOut.Text = temperatureOut.ToString();

// Storing the temperature into a string

unityIn = "F";

unityOut = "C";

}

// ---------------- BLOCK TO WORK WITH THE MESSAGE ---------

// Printing the text for the results into the text box.

// I will base the print on celsius temperature.

if (radioFromCtoF.Checked == true)

{

temperatureOut = temperatureIn;

}

// Conditionals to show the message on each interval

if (temperatureOut > 100)

{

message = "Water boiling.";

}

else if (temperatureOut == 100)

{

message = "Water boils.";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < 100 && temperatureOut > 40)

{

message = "TOO hot for bath.";

}

else if (temperatureOut == 40)

{

message = "Hot bath.";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < 40 && temperatureOut > 37)

{

message = "Fever, maybe?";

}

else if (temperatureOut == 37)

{

message = "Body temperature";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < 37 && temperatureOut > 21)

{

message = "Air conditioner recomended.";

}

else if (temperatureOut == 21)

{

message = "Room temperature.";

}

else if (temperatureOut < 21 && temperatureOut > 10)

{

message = "Starting to get cold, right?";

}

else if (temperatureOut == 10)

{

message = "Cool day";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < 10 && temperatureOut > 0)

{

message = "Welcome to canadian Summer.";

}

else if (temperatureOut == 0)

{

message = "Very cold day.";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < 0 && temperatureOut > -40)

{

message = "Welcome to canadian Fall";

}

else if (temperatureOut == -40)

{

message = "Extremely Cold Day (and the same number!)";

textBoxMessage.Font = new Font(textBoxMessage.Font, FontStyle.Bold);

}

else if (temperatureOut < -40)

{

message = "The winter is coming.";

}

// --------------- BLOCK TO MANIPULATE THE FILE ------------

// Writing the text into the file

textOut.Write(temperatureIn + unityIn + " = " + temperatureOut +

unityOut + ", " + DateTime.Now.ToString("yyyy'/'M'/'dd - hh:mm:ss tt - ")

+ message + "\n");

textOut.Close();

// Showing the message into the text box

textBoxMessage.Text = message;

}

// Changing the label

private void radioFromCtoF\_CheckedChanged(object sender, EventArgs e)

{

// changing the label when the radio button from C to F is

// checked

labelIn.Text = "C";

labelOut.Text = "F";

}

// Changing the label

private void radioFromFtoC\_CheckedChanged(object sender, EventArgs e)

{

// changing the label when the radio button from F to C is

// checked

labelIn.Text = "F";

labelOut.Text = "C";

}

// Block to read the file and to show it in a new window

private void buttonReadFile\_Click(object sender, EventArgs e)

{

string filePath = pathDir + file;

try

{// Instantiating the object textIn which readt the file

// in the indicated path

textIn = new StreamReader(filePath);

// Declaring an empty string list to store the read text

// in the file

string list = "";

// Setting the counter

int counter = 0;

// Starting the loop and stoping when the index is equal to -1

while (textIn.Peek() != -1)

{

string row = textIn.ReadLine();

list += "-- " + row.ToString() + "\n";

counter++;

if (counter == 5)

{

// The list will be shown into a new window

MessageBox.Show(list, "Temperature log");

counter = 0;

list = "";

}

}

if (counter != 0)

{

// The list will be shown into a new window

MessageBox.Show(list, "Exchange log");

counter = 0;

list = "";

}

textIn.Close();

}

catch (FileNotFoundException)

{

MessageBox.Show("File not created.");

File.Create(filePath).Close();

}

}

// Activating when click on exit

private void buttonExit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application? ",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

private void TempConv\_Load(object sender, EventArgs e)

{

// If the directory do not exists, we create it here

string filePath = pathDir + file;

if (!Directory.Exists(pathDir))

{

Directory.CreateDirectory(pathDir);

}

}

}

## Code for Calculator

public partial class Calculator : Form

{

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream fileStream;

StreamWriter textOut;

StreamReader textIn;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "Calculator.txt";

// Declaring the numbers

CalculatorOperations number1;

double number2;

// Declaring strings

string operation;

public Calculator()

{

InitializeComponent();

}

// If the number 1 is clicked, then it will generate the number

// 1, if the number 0 is already on the text box, it will

// substitute it by 1, if not, it will concatenate the

// number 1

private void one\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "1";

}

else

{

textBox1.Text = textBox1.Text + "1";

}

}

// If the number 2 is clicked, then it will generate the number

// 2, if the number 0 is already on the text box, it will

// substitute it by 2, if not, it will concatenate the

// number 2

private void two\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "2";

}

else

{

textBox1.Text = textBox1.Text + "2";

}

}

// I don't need to repeat it, but the same from 1 and 2 is valid

// here

private void three\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "3";

}

else

{

textBox1.Text = textBox1.Text + "3";

}

}

// I don't need to repeat it, but the same from 1 and 2 is valid

// here

private void four\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "4";

}

else

{

textBox1.Text = textBox1.Text + "4";

}

}

// again...

private void five\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "5";

}

else

{

textBox1.Text = textBox1.Text + "5";

}

}

// Keep going

private void six\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "6";

}

else

{

textBox1.Text = textBox1.Text + "6";

}

}

// Almost there...? No... (Too much click events)

private void seven\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "7";

}

else

{

textBox1.Text = textBox1.Text + "7";

}

}

// LOL!!!!

private void eight\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "8";

}

else

{

textBox1.Text = textBox1.Text + "8";

}

}

// NUMBER NEIN!

private void nine\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "9";

}

else

{

textBox1.Text = textBox1.Text + "9";

}

}

// Finaly zero, amen!

private void zero\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "0";

}

else

{

textBox1.Text = textBox1.Text + "0";

}

}

// Now something different. If you have zero in the text box

// It will generate 0. / If the textbox already contains a dot

// it will add nothing / else, it will add only the dot.

private void point\_Click(object sender, EventArgs e)

{

if (textBox1.Text == "0")

{

textBox1.Text = "0.";

}

else if (textBox1.Text.Contains(".") == true)

{

textBox1.Text += "";

}

else

{

textBox1.Text = textBox1.Text + ".";

}

}

// When you load the calculator, the textbox will

// have the number 0 already there, the textbox will be focused

// and the option ReadOnly will be activated.

private void Calculator\_Load(object sender, EventArgs e)

{

textBox1.Text = "0";

textBox1.Focus();

textBox1.ReadOnly = true;

}

private void clear\_Click(object sender, EventArgs e)

{

number1 = new CalculatorOperations(0);

number2 = 0;

textBox1.Text = "0";

textBox1.Focus();

}

private void addition\_Click(object sender, EventArgs e)

{

try

{

number1 = new CalculatorOperations(Convert.ToDouble(textBox1.Text));

textBox1.Text = "0";

operation = "+";

textBox1.Focus();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void subtraction\_Click(object sender, EventArgs e)

{

try

{

number1 = new CalculatorOperations(Convert.ToDouble(textBox1.Text));

textBox1.Text = "0";

operation = "-";

textBox1.Focus();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void multiplication\_Click(object sender, EventArgs e)

{

try

{

number1 = new CalculatorOperations(Convert.ToDouble(textBox1.Text));

textBox1.Text = "0";

operation = "\*";

textBox1.Focus();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void division\_Click(object sender, EventArgs e)

{

try

{

number1 = new CalculatorOperations(Convert.ToDouble(textBox1.Text));

textBox1.Text = "0";

operation = "/";

textBox1.Focus();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

// Click on equal event, meaning that it will show us the

// result of the chosen operation

private void equal\_Click(object sender, EventArgs e)

{

number2 = Convert.ToDouble(textBox1.Text);

// Declaring the objects to manipulate the file

string filePath = pathDir + file;

fileStream = new FileStream(filePath, FileMode.Append, FileAccess.Write);

textOut = new StreamWriter(fileStream);

// The result is reseted to zero when click on equal

double result = 0;

// Using a switch execute do operation with the two

// informed numbers

switch (operation)

{

case "+":

{

result = number1.Addition(number2);

textBox1.Text = (result).ToString();

break;

}

case "-":

{

result = number1.Subtraction(number2);

textBox1.Text = (result).ToString();

break;

}

case "\*":

{

result = number1.Multiplication(number2);

textBox1.Text = (result).ToString();

break;

}

case "/":

{

result = number1.Division(number2);

textBox1.Text = (result).ToString();

break;

}

default:

{

textBox1.Text = "0";

break;

}

}

// Line to write in the file the operation executed

textOut.Write(number1.InNumber.ToString() + " " + operation

+ " " + number2 + " = " + result + "\n");

number1 = new CalculatorOperations(result);

textOut.Close();

}

// Closing the application

private void exit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application?",

"Exit Simple Calculator?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

}

## Code for Calculator Operations Class

internal class CalculatorOperations

{

// Private field

private double inNumber;

// Property

public double InNumber

{

get { return inNumber; }

set { inNumber = value; }

}

public CalculatorOperations(double number)

{

this.InNumber = number;

}

// Methods

public double Addition(double otherNumber)

{

return InNumber + otherNumber;

}

public double Subtraction(double otherNumber)

{

return InNumber - otherNumber;

}

public double Multiplication(double otherNumber)

{

return InNumber \* otherNumber;

}

public double Division(double otherNumber)

{

if (otherNumber == 0)

{

MessageBox.Show("Cannot divide by zero");

return 0;

}

else

{

return InNumber / otherNumber;

}

}

}

## Code for IPv4 Validator

public partial class Ip4Validator : Form

{

// Declaring the object to store the time

// Time open store the time when the application is open

DateTime timeOpen = DateTime.Now;

// Declaring the structures to be used into the classes

// that will manipulate the generated files

FileStream binaryStream;

BinaryWriter binaryOut;

BinaryReader binaryIn;

// Declaring the structure to be used for showing the dates

DateTime date;

// Declaring the path to write the file

string pathDir = @".\Files\";

string file = "BinaryIPv4.txt";

public Ip4Validator()

{

InitializeComponent();

}

// As the application is open, the date time will be displayed

// on the window in the empty label above the text box

private void Ip4Validator\_Load(object sender, EventArgs e)

{

// Set to verify if the directory already exists.

string filepath = pathDir + file;

if (!Directory.Exists(pathDir))

{

Directory.CreateDirectory(pathDir);

}

// Set to print the actual time

string loadTime;

loadTime = DateTime.Now.ToLongDateString();

labelShow.Text = "Today : " + loadTime;

}

private void buttonValidate\_Click(object sender, EventArgs e)

{

string pathBinary = pathDir + file;

binaryStream = new FileStream(pathBinary, FileMode.Append, FileAccess.Write);

binaryOut = new BinaryWriter(binaryStream);

// Declaring variables

// starting reading the text into the text box

string ipInput = textBoxIn.Text;

// each octet (in decimal form) will be displayed

// into an array of octet

string[] octets;

// declaring the message of validation as an empty string

string messageValidation = "";

// Declaring message of IP class as an empty string

string messageClassIP = "";

// Whn the input is informed, it will be trimmed to remove

// unecessary white spaces and the single white spaces

// remaining will be replaced by an empty string

ipInput = ipInput.Trim().Replace(" ", "");

// Then it will display the corrected entry

textBoxIn.Text = ipInput;

// the input will be split using a dot as a separator

// and stored inside the array octets

octets = ipInput.Split('.');

// The regex object is declared here

Regex ipValidation = new Regex(@"^([1-9]|[1-9][0-9]|1[0-9][0-9]|2[0-5][0-5])\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-5][0-5])\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-5][0-5])\.([0-9]|[1-9][0-9]|1[0-9][0-9]|2[0-5][0-5])$");

// Testing if the object matches with the input

if (ipValidation.IsMatch(ipInput) == true)

{

messageValidation = "This IP is correct";

// The set of if will verify the first octet to

// show the class of the IP

if (Convert.ToInt16(octets[0]) > 0 && Convert.ToInt16(octets[0]) < 128)

{

messageClassIP = "Class A IP";

}

else if (Convert.ToInt16(octets[0]) > 128 && Convert.ToInt16(octets[0]) < 192)

{

messageClassIP = "Class B IP";

}

else if (Convert.ToInt16(octets[0]) > 192 && Convert.ToInt16(octets[0]) < 224)

{

messageClassIP = "Class C IP";

}

else if (Convert.ToInt16(octets[0]) > 224 && Convert.ToInt16(octets[0]) < 240)

{

messageClassIP = "Class C IP";

}

else

{

messageClassIP = "Class E IP";

}

// The message box will show the ip input, the class of the IP,

// and the message of validation

MessageBox.Show(ipInput + "\n" + messageClassIP +

"\n" + messageValidation, "Valid IP");

// The ip input is written here considering as a string

binaryOut.Write(ipInput);

}

else

{

// The message box shows the user the correct entry format.

MessageBox.Show(ipInput + "\n" + "The IP must have 4 bytes\ninteger numbers" +

"between 1 to 255 on the first interval, and between 0 to 255 on the other" +

"intervals\nseparated by a dot (255.255.255.255)", "Error");

}

// The binary file is closed

binaryOut.Close();

}

// When the button reset is pressed, the text box is emptied

private void buttonReset\_Click(object sender, EventArgs e)

{

textBoxIn.Text = string.Empty;

textBoxIn.Focus();

}

// When exit button is pressed, it will ask the user if

// they want to leave the application, and it will

// show the elapsed time

private void buttonExit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit this application:\nIPv4 Validator?",

"Exit?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

// The elapsed time is calculated by considering

// the time when the application is loaded

// subtracted to the time the application

// is closed.

DateTime timeClose = DateTime.Now;

TimeSpan timeSpan = timeClose - timeOpen;

MessageBox.Show("Elapsed time: " + timeSpan.ToString(@"mm\:ss"), "Session closed");

this.Close();

}

}

}

1. **Present the classes and/or methods that you create or you did use in the project.**

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
| 1. CalculatorOperations | This class is used to store a double number and use it to perform simple arithmetic operations |
| 1. CalculatorOperations() | Is a constructor to store a double and give access to private variables of the CalculatorOperations class |
| 1. Addition(double number) | Is a method receive an external number to perform an addition between this number and the number in the properties of the class |
| 1. Subtraction(double number) | Is a method receive an external number to perform an subtraction between this number and the number in the properties of the class |
| 1. Multiplication(double number) | Is a method receive an external number to perform an multiplication between this number and the number in the properties of the class |
| 1. Division(double number) | Is a method receive an external number to perform an division between this number and the number in the properties of the class |

1. **Present the difficulties that you have, what was the hardest and the easiest part of your project.**

The easiest part is to create the design of the forms. The hardest part was to check if the number was used already so I could generate the unique set of numbers. Also, it was hard to display the lines of the text using a text box.

Until this moment, I couldn’t find out how to change the colors of the string shown on the text boxes.