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

|  |
| --- |
| Francisco Ricardo Andraschko  4/15/2023  Version: 5.0 |

1. **Start by adding a short description of your project, and the languages (technologies) used:**
2. Language: C#, Windows Forms .Net Framework
3. Tools (IDE): Microsoft Visual Studio Community 2022 (64-bit) - Version 17.5.2
4. **Present the print screens of yours forms and have a detailed description of the functionalities (step by step).**

MainForm: The main form provide access to the features:

* 1. Numbers Generators

Graphical user interface, application

Description automatically generated

* 1. Conversions

Graphical user interface, application

Description automatically generated

* 1. Calculator

Graphical user interface, application

Description automatically generated

* 1. IP v4 Validator

Graphical user interface, application

Description automatically generated

On this screen, there are two ways to access the desired resource, by clicking on the tabs or on the menu bar.

1. Graphical user interface, application, Word

   Description automatically generatedNumbers Generator: this form provide access to generator of numbers to Lotto Max and Lotto 649

Graphical user interface, application

Description automatically generated

* 1. Lotto Max: on this screen there are 3 buttons as follow:

1. Generate: This button generates 7 random numbers and 1 bonus random number, all unique, between 1 and 50. The results are placed in a label and text box, after which the values are written in the last line of the text file named “LottoMax.txt”

Graphical user interface, application

Description automatically generated Tela de computador com texto preto sobre fundo branco

Descrição gerada automaticamente

1. Read file: this button read a text file named “LottoMax.txt” saved into the folder “\files\” and exhibit in the form showmodal.

Text

Description automatically generated

1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form.

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

Description automatically generated

* 1. Lotto 649: on this screen there are 3 buttons:

1. Generate: this button generates 6 random numbers and 1 bonus random number, all of them are unique, between 1 and 49. The results are placed in a label and text box, after which the values ​​are written to the last line in the text file named “Lotto649.txt”.

Graphical user interface, application

Description automatically generated Text

Description automatically generated

1. Read file: this button read a text file named “Lotto649.txt” saved into the folder “\files\” and exhibit in the form show modal.

Text

Description automatically generated

1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form.

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

Description automatically generated

1. Conversions: this form provides two options to conversions
   1. Money Exchange: this form provides a money exchange from / to five exchange currencies: CAD (Canadian Dollar), USD (US Dollar), EUR (Euro), GBP (British Pound Sterling) and BRL (Brazilian Real). This form has 2 fields (one editable: where the amount “from” was selected and one read-only: where the amount converted is shown) and 3 buttons:

Graphical user interface, application, Word

Description automatically generated Text

Description automatically generated

* + 1. Convert: this button uses the values of respective exchange currencies selected and calculate the respective amount in the “to” selected, showing in the textbox (read-only), after which the values ​​are written to the last line in the text file named “MoneyEx.txt”.

Graphical user interface, application, Word

Description automatically generated

* + 1. this button read a text file named “MoneyEx.txt” saved into the folder “\files\” and exhibit in the new screen.

Graphical user interface, text

Description automatically generated

* + 1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form.

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

Description automatically generated

* 1. Temperature Convert: this form provides a temperature convert, from C to F or from F to C. This form has 2 fields (one editable: where the value “from” was inputted and one read-only: where the value converted is shown) and 3 buttons:

Graphical user interface, application

Description automatically generated Text

Description automatically generated

1. Convert: this button uses the values of respective temperature (C or F) selected and calculate the respective value in the other temperature (C to F or F to C), showing in the textbox (read-only), after which the values ​​are written to the last line in the text file named “TempConv.txt”

Graphical user interface, application

Description automatically generated

1. Read file: this button read a text file named “TempConv.txt” saved into the folder “\files\” and exhibit in the new screen.

Text

Description automatically generated

1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form

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

Description automatically generated

1. Calculator: this form provides a simple calculator and its mainly math operators. Its operation is:

By clicking in the respective number, the first number is shown in the textbox. Then is necessary to click on the operation button and after the second number (also shown in the textbox Historic) at the end click on the button Equal or repeat the process to include other operations in sequence.

A screenshot of a computer

Description automatically generated with medium confidence Graphical user interface, application

Description automatically generated

* + 1. Add: add two decimal numbers

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

* + 1. Sub: subtract two decimal numbers

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

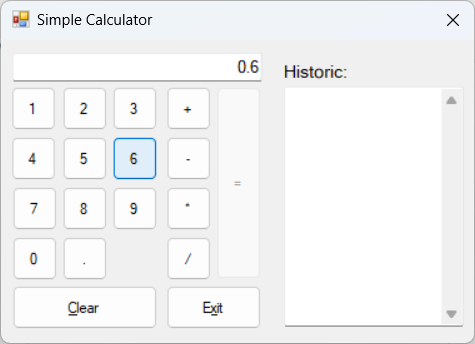
Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

* + 1. Mul: multiply two decimal numbers

A screenshot of a computer

Description automatically generated with medium confidence 

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

* + 1. Div: divide two decimal numbers, since de second (after click on “/” button) is greater than 0.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a cell phone

Description automatically generated with medium confidenceGraphical user interface, text, application

Description automatically generated A screenshot of a computer

Description automatically generated with medium confidence

* + 1. Clear: clean the textbox and the operators, restarting to initial state.

A screenshot of a computer

Description automatically generated with medium confidence A screenshot of a computer

Description automatically generated with medium confidence

* + 1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form

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

Description automatically generated

1. IP v4 Validator: this form validate the IP v.4 format using REGEX, showing the user if it is a correct number or not. If applicable, this informations are saved into the binary file. Also has in this form a time control, letting get to user know how long he is online on the form.

Graphical user interface, application

Description automatically generated Text

Description automatically generated

* + 1. Validate IP: this button calls the function to validate with Regex code if the value inputted is a valid IP or not. If valid, it is saved at end of the binary file, add the current time. If not, it is shown to user, nothing happens into the binary file.

Graphical user interface, application

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

Text

Description automatically generated

Graphical user interface, application

Description automatically generated Graphical user interface, text, application

Description automatically generated

* + 1. Reset: this button clears the text and focus the cursor on the textbox.

Graphical user interface, application

Description automatically generated Graphical user interface, application

Description automatically generated

* + 1. Exit: this button calls the function responsible to close this form, but it needs a user confirmation to close the form. After close the form, it is shown a message to the total time spent on the form, in a format MM:SS.

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

Description automatically generated Graphical user interface, text, application, chat or text message

Description automatically generated

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

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinFormsFinalProject

{

public partial class MainForm : Form

{

public MainForm()

{

InitializeComponent();

}

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

{

LottoMAX obj = new LottoMAX();

obj.ShowDialog();

}

private void btnLotto649\_Click(object sender, EventArgs e)

{

Lotto649 obj = new Lotto649();

obj.ShowDialog();

}

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

{

string title = "Application";

if (MessageBox.Show("Do you want to quit the main \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

Application.Exit();

}

}

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

{

//MoneyEx obj = new MoneyEx();

MoneyConv obj = new MoneyConv();

obj.ShowDialog();

}

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

{

TempConv obj = new TempConv();

obj.ShowDialog();

}

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

{

SimpleCalculator obj = new SimpleCalculator();

obj.ShowDialog();

}

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

{

IP4\_Validator obj = new IP4\_Validator();

obj.ShowDialog();

}

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

{

string dir = @".\files\";

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

}

}

}

1. LottoMax

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.RegularExpressions;

using System.Collections;

using System.IO;

using static System.Net.Mime.MediaTypeNames;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using System.Runtime.InteropServices.ComTypes;

using System.Runtime.Remoting.Messaging;

namespace WinFormsFinalProject

{

public partial class LottoMAX : Form

{

public LottoMAX()

{

InitializeComponent();

}

// Data to txt file

string dir = @".\files\";

string file = "LottoMax.txt";

string name = "Lotto Max";

public string randSeq(int range)

{

Random random = new Random();

string tempString = ""; // variable to return

List<int> numbers = Enumerable.Range(1, 50).ToList(); // Adding the random numbers in the array.

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

{

int index = random.Next(numbers.Count);

int randomNumber = numbers[index];

tempString += randomNumber.ToString() + "\t"; // add TAB after the random number

numbers.RemoveAt(index); // Remove all the repeating numbers.

}

string txt = tempString.Remove(tempString.Length - 1, 1); // remove last TAB or , and return the 'range' random numbers

return (txt);

}

public string randControl(int range)

{

Random random = new Random();

string tempString = ""; // variable to return

List<int> numbers = Enumerable.Range(1, 9).ToList(); // Adding the random numbers in the array.

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

{

int index = random.Next(numbers.Count);

int randomNumber = numbers[index];

tempString += randomNumber.ToString() + " "; // add TAB after the random number

//numbers.RemoveAt(index); // Remove all the repeating numbers.

}

return tempString;

}

public void createFile(string dir, string file, string name, string txt)

{

string path = Path.Combine(dir, file);

DateTime currentDate = DateTime.Now;

txt = txt.Replace("\t", ",");

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

else

{

StreamWriter sw = new StreamWriter(path, true); //Pass the filepath and filename to the StreamWriter Constructor

sw.WriteLine($"{name}, {currentDate}, {txt.Remove(txt.LastIndexOf(","))}, Bonus " + txt.Substring(txt.LastIndexOf(",") + 1)); //Write a line of text

sw.Close(); //Close the file

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex.Message,"Error " + name);

}

}

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

{

int range = 8; // qty of random numbers

label2.Text = "";

label2.Text = randControl(range); // label2 = "Range" random numbers

textBox1.Text = "";

textBox1.Text = randSeq(range); // textBox1 = "Range" random numbers

string txt = textBox1.Text;

createFile(dir, file, name, txt); //Create text file.

}

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

{

string title = "Lotto Max";

if (MessageBox.Show("Do you want to quit the application \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

}

}

private void showMsgModal(string title, string msg)

{

ShowMsg msgModal = new ShowMsg();

msgModal.Text = title;

msgModal.textBox1.Text = (msg.Trim());

msgModal.ShowDialog();

}

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

{

string path = Path.Combine(dir, file);

string title = "Lotto Max";

string textToPrint = "";

FileStream stream = null;

byte counter = 0;

int numexhibit = 10;

try

{

using (stream = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(stream))

{

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

{

string line = reader.ReadLine();

textToPrint += line + "\r\n";

}

// call the modal form, putting the text and title.

showMsgModal(title, textToPrint);

reader.Close();

}

}

catch (IOException ex)

{

MessageBox.Show("IO Exception\n" + ex.Message);

}

finally

{

if (stream != null)

stream.Close();

}

}

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

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

}

}

}

1. Lotto649

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace WinFormsFinalProject

{

public partial class Lotto649 : Form

{

public Lotto649()

{

InitializeComponent();

}

// Data to txt file

string dir = @".\files\";

string file = "Lotto649.txt";

string name = "Lotto 649";

public string randSeq(int range)

{

Random random = new Random();

string tempString = ""; // variable to return

List<int> numbers = Enumerable.Range(1, 49).ToList(); // Adding the random numbers in the array.

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

{

int index = random.Next(numbers.Count);

int randomNumber = numbers[index];

tempString += randomNumber.ToString() + "\t"; // add TAB after the random number

numbers.RemoveAt(index); // Remove all the repeating numbers.

}

string txt = tempString.Remove(tempString.Length - 1, 1); // remove last TAB or , and return the 'range' random numbers

return (txt);

}

public string randControl(int range)

{

Random random = new Random();

string tempString = ""; // variable to return

List<int> numbers = Enumerable.Range(1, 9).ToList(); // Adding the random numbers in the array.

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

{

int index = random.Next(numbers.Count);

int randomNumber = numbers[index];

tempString += randomNumber.ToString() + ""; // add TAB after the random number

numbers.RemoveAt(index); // Remove all the repeating numbers.

}

return tempString;

}

public void createFile(string dir, string file, string name, string txt)

{

DateTime currentDate = DateTime.Now;

txt = txt.Replace("\t", ",");

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

else

{

string path = Path.Combine(dir, file);

StreamWriter sw = new StreamWriter(path, true); //Pass the filepath and filename to the StreamWriter Constructor

sw.WriteLine($"{name}, {currentDate}, {txt.Remove(txt.LastIndexOf(","))}, Bonus " + txt.Substring(txt.LastIndexOf(",") + 1)); //Write a line of text

sw.Close(); //Close the file

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex.Message, "Error " + name);

}

}

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

{

int range = 7; // qty of random numbers

label2.Text = "";

label2.Text = randControl(range); // label2 = "Range" random numbers

textBox1.Text = "";

textBox1.Text = randSeq(range); // textBox1 = "Range" random numbers

string txt = textBox1.Text;

createFile(dir, file, name, txt); //Create text file.

}

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

{

string title = "Lotto 649";

if (MessageBox.Show("Do you want to quit the application \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

}

}

private void showMsgModal(string title, string msg)

{

ShowMsg msgModal = new ShowMsg();

msgModal.Text = title;

msgModal.textBox1.Text = (msg.Trim());

msgModal.ShowDialog();

}

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

{

string path = Path.Combine(dir, file);

string title = "Lotto 649";

string textToPrint = "";

FileStream stream = null;

byte counter = 0;

int numexhibit = 10;

try

{

using (stream = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(stream))

{

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

{

string line = reader.ReadLine();

textToPrint += line + "\r\n";

}

// call the modal form, putting the text and title.

showMsgModal(title, textToPrint);

reader.Close();

}

}

catch (IOException ex)

{

MessageBox.Show("IO Exception\n" + ex.Message);

}

finally

{

if (stream != null)

stream.Close();

}

}

}

}

1. MoneyConv

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Xml.Linq;

namespace WinFormsFinalProject

{

public partial class MoneyConv : Form

{

public MoneyConv()

{

InitializeComponent();

}

// Data to txt file

string dir = @".\files\";

string file = "MoneyConv.txt";

string name = "Money Exchange";

//format to save in the txt file

//100 CAD = 76 USD, 2022/7/29 9:57:33 AM

public void createFile(string dir, string file, string name, string txt)

{

DateTime currentDate = DateTime.Now;

try

{

string path = Path.Combine(dir, file);

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

else

{

StreamWriter sw = new StreamWriter(path, true); //Pass the filepath and filename to the StreamWriter Constructor

sw.WriteLine($"{txt}, {currentDate}"); //Write a line of text

sw.Close();

}

//Close the file

}

catch (Exception e)

{

Console.WriteLine("Exception: " + e.Message);

}

finally

{

Console.WriteLine("Executing finally block.");

}

}

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

{

// default select

rbCAD1.Checked = true;

rbCAD2.Checked = true;

}

private void rbCheckedChanged(object sender, EventArgs e)

{

// clear the Convert value alwways radiobutton checked changed

textBox2.Text = "";

}

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

{

/\* Rates in 2023-03-28

CAD: 1.00 --> 0.79 USD

CAD: 1.00 --> 0.68 EUR

CAD: 1.00 --> 0.60 GBP

CAD: 1.00 --> 3.79 BRL

\*/

decimal CAD = 1.00m;

decimal USD = 0.79m;

decimal EUR = 0.68m;

decimal GBP = 0.60m;

decimal BRL = 3.79m;

decimal amount = 0;

decimal fMoney = 0;

decimal sMoney = 0;

string txt = ""; // text file

string fText = ""; // text from money

string tText = ""; // text to money

try

{

if (textBox1.Text != "")

{

if (decimal.Parse(textBox1.Text) > 0)

{

// Running every groupbox control

foreach (RadioButton radioButton1 in groupBox1.Controls.OfType<RadioButton>())

{

// check if radio button is checked

if (radioButton1.Checked)

{

// select the value to convertion

switch (radioButton1.Text)

{

case "CAD":

{

fMoney = CAD;

fText = radioButton1.Text;

break;

}

case "USD":

{

fMoney = USD;

fText = radioButton1.Text;

break;

}

case "EUR":

{

fMoney = EUR;

fText = radioButton1.Text;

break;

}

case "GBP":

{

fMoney = GBP;

fText = radioButton1.Text;

break;

}

case "BRL":

{

fMoney = BRL;

fText = radioButton1.Text;

break;

}

default:

break;

}

}

}

foreach (RadioButton radioButton2 in groupBox2.Controls.OfType<RadioButton>())

{

if (radioButton2.Checked)

{

// select the value to convertion

switch (radioButton2.Text)

{

case "CAD":

{

sMoney = CAD;

tText = radioButton2.Text;

break;

}

case "USD":

{

sMoney = USD;

tText = radioButton2.Text;

break;

}

case "EUR":

{

sMoney = EUR;

tText = radioButton2.Text;

break;

}

case "GBP":

{

sMoney = GBP;

tText = radioButton2.Text;

break;

}

case "BRL":

{

sMoney = BRL;

tText = radioButton2.Text;

break;

}

default:

break;

}

}

}

// convert = ( FROM \* textbox1 ) / TO

amount = (Convert.ToDecimal(textBox1.Text) \* sMoney) / fMoney;

// round value 0.00 format

amount = decimal.Round(amount, 2);

// put the convert value into textbox2

textBox2.Text = amount.ToString();

// txt = initial value + initial money = converted value + final money

txt = textBox1.Text + " " + fText + " = " + textBox2.Text + " " + tText;

createFile(dir, file, name, txt); //Create text file.

}

else

{

MessageBox.Show("Input a positive value to convert", "Convert Value", MessageBoxButtons.OK);

textBox1.Focus();

}

}

else

{

MessageBox.Show("Input a positive value to convert", "Convert Value", MessageBoxButtons.OK);

textBox1.Focus();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex.Message);

}

}

private void showMsgModal(string title, string msg)

{

ShowMsg msgModal = new ShowMsg();

msgModal.Text = title;

msgModal.textBox1.Text = (msg.Trim());

msgModal.ShowDialog();

}

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

{

string path = Path.Combine(dir, file);

string title = "Money Exchange";

string textToPrint = "";

FileStream stream = null;

byte counter = 0;

int numexhibit = 10;

try

{

using (stream = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(stream))

{

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

{

string line = reader.ReadLine();

textToPrint += line + "\r\n";

}

// call the modal form, putting the text and title.

showMsgModal(title, textToPrint);

reader.Close();

}

}

catch (IOException ex)

{

MessageBox.Show("IO Exception\n" + ex.Message);

}

finally

{

if (stream != null)

stream.Close();

}

}

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

{

string title = "Money Exchange";

if (MessageBox.Show("Do you want to quit the application \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

}

}

}

}

1. TempConv

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SqlTypes;

using System.Drawing;

using System.Linq;

using System.Security.Policy;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Net.Mime.MediaTypeNames;

using System.Xml.Linq;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

using static System.Windows.Forms.VisualStyles.VisualStyleElement.Button;

using System.IO;

using System.Text.RegularExpressions;

namespace WinFormsFinalProject

{

public partial class TempConv : Form

{

public TempConv()

{

InitializeComponent();

}

// Data to txt file

string dir = @".\files\";

string file = "TempConv.txt";

string name = "Temperature Conversion";

// format to save in the txt file

// 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

public void createFile(string dir, string file, string name, string txt, string msg)

{

DateTime currentDate = DateTime.Now;

try

{

string path = Path.Combine(dir, file);

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

else

{

StreamWriter sw = new StreamWriter(path, true); //Pass the filepath and filename to the StreamWriter Constructor

sw.WriteLine($"{txt}, {currentDate} {msg}"); //Write a line of text

sw.Close();

}

//Close the file

}

catch (Exception e)

{

Console.WriteLine("Exception: " + e.Message);

}

finally

{

Console.WriteLine("Executing finally block.");

}

}

private void rbCheckedChanged(object sender, EventArgs e)

{

// clear the Convert value alwways radiobutton checked changed

textBox1.Text = "";

lbMessage.Text = "";

lbValueTo.Text = "";

if (radioButton1.Checked)

{

lbTextFrom.Text = "C";

lbTextTo.Text = "F";

}

else if (radioButton2.Checked)

{

lbTextFrom.Text = "F";

lbTextTo.Text = "C";

}

}

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

{

decimal c = 0;

decimal f = 0;

Regex formatNumber = new Regex(@"^[+-]?\d+(\.\d+)?$");

try

{

//validate if format number is: +NN, -NN or NN, w/ or w/o decimal numbers

if ( formatNumber.IsMatch(textBox1.Text))

{

if (radioButton1.Checked)

{

c = decimal.Parse(textBox1.Text);

f = (c \* 9 / 5) + 32;

f = decimal.Round(f,1);

lbValueTo.Text = f.ToString();

switch (f)

{

case var n when (n > 100 && n != 104 && n != 212):

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Scalding Water";

lbValueTo.ForeColor = Color.Red;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 212:

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Water boils";

lbValueTo.ForeColor = Color.Red;

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

break;

}

case 104:

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Hot Bath";

lbValueTo.ForeColor = Color.Red;

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

break;

}

case var n when (n >= 60 && n < 100 && n != 70 && n != 86 && n != 98.6m):

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Comfortable Temperature";

lbValueTo.ForeColor = Color.Green;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 98.6m:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Body Temperature";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case 86:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Beach Water";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case 70:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Room Temperature";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case var n when (n < 60 && n != 50 && n != 32 && n != 0 && n != -40):

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Cold Day";

lbValueTo.ForeColor = Color.Blue;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 50:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Cool Day";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case 32:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Freezing point of water";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case 0:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Very Cold Day";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case -40:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Extremaly Cold Day\n(and the same number!)";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

default:

{

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

}

}

else if (radioButton2.Checked)

{

f = decimal.Parse(textBox1.Text);

c = (f - 32) \* 5 / 9;

c = decimal.Round(c, 1);

lbValueTo.Text = c.ToString();

switch (c)

{

case var n when (n > 40 && n != 100):

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Scalding Water";

lbValueTo.ForeColor = Color.Red;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 100:

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Water boils";

lbValueTo.ForeColor = Color.Red;

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

break;

}

case 40:

{

lbMessage.ForeColor = Color.Red;

lbMessage.Text = "Hot Bath";

lbValueTo.ForeColor = Color.Red;

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

break;

}

case var n when(n > 15 && n < 40 && n != 37 && n != 30 && n != 21):

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Comfortable Temperature";

lbValueTo.ForeColor = Color.Green;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 37:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Body Temperature";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case 30:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Beach Water";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case 21:

{

lbMessage.ForeColor = Color.Green;

lbMessage.Text = "Room Temperature";

lbValueTo.ForeColor = Color.Green;

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

break;

}

case var n when (n < 15 && n != 10 && n != 0 && n != -40):

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Cold Day";

lbValueTo.ForeColor = Color.Blue;

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

case 10:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Cool Day";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case 0:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Freezing point of water";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case (-18):

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Very Cold Day";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

case -40:

{

lbMessage.ForeColor = Color.Blue;

lbMessage.Text = "Extremaly Cold Day\n(and the same number!)";

lbValueTo.ForeColor = Color.Blue;

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

break;

}

default:

{

lbValueTo.Font = new Font(textBox1.Font, FontStyle.Regular);

break;

}

}

}

else

{

MessageBox.Show("Input a value to convert", "Convert Value", MessageBoxButtons.OK);

textBox1.Focus();

}

// txt = initial value + C/F = converted value + F/C

string txt = textBox1.Text + " " + lbTextFrom.Text + " = " + lbValueTo.Text + " " + lbTextFrom.Text;

string msg = lbMessage.Text.Replace("\n", " ");

createFile(dir, file, name, txt, msg); //Create text file.

}

else

{

MessageBox.Show("Input a value to convert", "Convert Value", MessageBoxButtons.OK);

textBox1.Focus();

}

}

catch (Exception ex)

{

MessageBox.Show("Exception: " + ex.Message);

}

}

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

{

string title = "Temperature Conversion";

if (MessageBox.Show("Do you want to quit the main \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

}

}

private void showMsgModal(string title, string msg)

{

ShowMsg msgModal = new ShowMsg();

msgModal.Text = title;

msgModal.textBox1.Text = (msg);

msgModal.ShowDialog();

}

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

{

string path = Path.Combine(dir, file);

string title = "Temperature Conversion";

string textToPrint = "";

FileStream stream = null;

try

{

using (stream = new FileStream(path, FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(stream))

{

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

{

string line = reader.ReadLine();

// inc the text, each one per new line

textToPrint += line + "\r\n";

}

// call the modal form, putting the text and title.

showMsgModal(title, textToPrint);

reader.Close();

}

}

catch (IOException ex)

{

MessageBox.Show("IO Exception\n" + ex.Message);

}

finally

{

if (stream != null)

stream.Close();

}

}

}

}

1. SimpleCalculator

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.Linq;

using System.Reflection;

using System.Text;

using System.Windows.Forms;

namespace WinFormsFinalProject

{

public partial class SimpleCalculator : Form

{

private Calculator calculator;

// variable operation --> private, because has another op in Calculator.cs

private string op;

bool decimalPoint = false;

// there's the first caracter of string

bool firstNumber = true;

// it is the second operation

bool secondOperand = false;

// Data to txt file

string dir = @".\files\";

string file = "Calculator.txt";

string name = "Simple Calculator";

public SimpleCalculator()

{

InitializeComponent();

// new obj type Calculator

calculator = new Calculator();

// initializing operation

op = "";

// if has decimal point, inicializing no (false)

decimalPoint = false;

// if first number (true)

firstNumber = true;

// used to keyboard press

KeyPreview = true;

}

public void enableOpBtn(bool enable)

{

if (enable)

{

btnDiv.Enabled = true;

btnMul.Enabled = true;

btnSub.Enabled = true;

btnSum.Enabled = true;

}

else

{

btnSum.Enabled = false;

btnDiv.Enabled = false;

btnMul.Enabled = false;

btnSub.Enabled = false;

}

}

// onclick for numbers buttons (0..9)

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

{

Button btn = (Button)sender;

try

{

if (op == "" && firstNumber)

{

textBox1.Text = btn.Text;

// now, is not first number

firstNumber = false;

// enable buttons

enableOpBtn(true);

}

else

{

textBox1.Text += btn.Text;

}

if (secondOperand)

{

// disable buttons

enableOpBtn(false);

//enable equal button

btnEqual.Enabled = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

private void btnPointClick(object sender, EventArgs e)

{

Button btn = (Button)sender;

try

{

// if has decimal point, ignore

if (!decimalPoint)

{

if (op == "" && firstNumber)

{

textBox1.Text = "0.";

// now, is not first number

firstNumber = false;

// enable buttons

enableOpBtn(true);

}

else

{

textBox1.Text += btn.Text;

}

// now has decimal point

decimalPoint = true;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

// onclick for operations buttons (+ - \* /)

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

{

Button btn = (Button)sender;

try

{

if (op == "")

{

//op = btn.Text;

calculator.Op = btn.Text;

calculator.Operand1 = decimal.Parse(textBox1.Text);

textBox2.Text += textBox1.Text + "\r\n" + calculator.Op;

textBox1.Text = "";

// control if secondOperator is eneble to insert

secondOperand = true;

decimalPoint = false;

// disable buttons

enableOpBtn(false);

btnEqual.Enabled = false;

}

else

{

// disable buttons

enableOpBtn(false);

//disaable equal button

btnEqual.Enabled = false;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

try

{

if (decimal.Parse(textBox1.Text) == 0)

{

MessageBox.Show("Cannot divide by zero", name, MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

// operand2

calculator.Operand2 = Convert.ToDecimal(textBox1.Text);

// call the Equals to calculate the operaation

calculator.Equals();

// put the result into textbox1, rouding to 10 decimals

textBox1.Text = Math.Round(calculator.CurrentValue, 10).ToString();

// show operations into textbox2

string txt = "";

txt = calculator.Operand2 + "\r\n=" + Math.Round(calculator.CurrentValue, 10) + "\r\n--------\r\n";

textBox2.AppendText(txt);

// save the operations into txt file

saveOnFile(calculator.Operand1, calculator.Op, calculator.Operand2);

// clear the operation

op = "";

// there's no number (will be the first)

firstNumber = true;

// there's no decimal point

decimalPoint = false;

// disable buttons

enableOpBtn(false);

btnEqual.Enabled = false;

// flag there's no second operand

secondOperand = false;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message, "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

//Clear();

calculator.Clear();

textBox1.Text = "";

textBox2.Text = "";

op = "";

// there's no number

firstNumber = true;

secondOperand = false;

decimalPoint = false;

// read only

textBox1.ReadOnly = true;

textBox2.ReadOnly = true;

//buttons disabled

enableOpBtn(false);

btnEqual.Enabled = false;

}

private void saveOnFile(decimal printOp1, string printOp, decimal printOp2)

{

string path = Path.Combine(dir, file);

string line = $"{printOp1.ToString()} {printOp} {printOp2.ToString()} = {calculator.CurrentValue}";

try

{

using (StreamWriter sw = new StreamWriter(path, true))

{

sw.WriteLine(line);

}

}

catch (Exception ex)

{

MessageBox.Show("Error: " + ex.Message);

}

}

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

{

//Clear();

calculator.Clear();

textBox1.Text = "";

textBox2.Text = "";

op = "";

// there's no number

firstNumber = true;

// read only

textBox1.ReadOnly = true;

textBox2.ReadOnly = true;

//buttons disabled

enableOpBtn(false);

btnEqual.Enabled = false;

}

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

{

string title = "Simple Calculator";

if (MessageBox.Show("Do you want to quit the application \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

}

}

private void SimpleCalculator\_KeyPress(object sender, KeyPressEventArgs e)

{

//switch (e.KeyChar)

//{

// case '1':

// {

// // Simulate a button click

// button1.PerformClick();

// break;

// }

// case '2':

// {

// // Simulate a button click

// button2.PerformClick();

// break;

// }

// case '3':

// {

// // Simulate a button click

// button3.PerformClick();

// break;

// }

// case '4':

// {

// // Simulate a button click

// button4.PerformClick();

// break;

// }

// case '5':

// {

// // Simulate a button click

// button5.PerformClick();

// break;

// }

// case '6':

// {

// // Simulate a button click

// button6.PerformClick();

// break;

// }

// case '7':

// {

// // Simulate a button click

// button7.PerformClick();

// break;

// }

// case '8':

// {

// // Simulate a button click

// button8.PerformClick();

// break;

// }

// case '9':

// {

// // Simulate a button click

// button9.PerformClick();

// break;

// }

// case '0':

// {

// // Simulate a button click

// button0.PerformClick();

// break;

// }

// case '.':

// {

// // Simulate a button click

// btnPoint.PerformClick();

// break;

// }

// case '+':

// {

// // Simulate a button click

// btnSum.PerformClick();

// break;

// }

// case '-':

// {

// // Simulate a button click

// btnSum.PerformClick();

// break;

// }

// case '\*':

// {

// // Simulate a button click

// btnSum.PerformClick();

// break;

// }

// case '/':

// {

// // Simulate a button click

// btnSum.PerformClick();

// break;

// }

// case (char)13:

// {

// // Simulate a button click

// btnEqual.PerformClick();

// break;

// }

//}

}

}

}

1. IP4-Validator

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.IO;

using System.IO.Pipes;

using System.Linq;

using System.Runtime.InteropServices.ComTypes;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsFinalProject

{

public partial class IP4\_Validator : Form

{

private DateTime formLoadTime;

// Data to txt file

string dir = @".\files\";

string file = "IP4-Validate.bin";

public IP4\_Validator()

{

InitializeComponent();

}

private bool ValidIP(string ip)

{

Regex myRegex = new Regex(@"^(25[0-5]|2[0-4]\d|[0-1]?\d?\d)(\.(25[0-5]|2[0-4]\d|[0-1]?\d?\d)){3}$");

return myRegex.IsMatch(ip);

}

private void IP4\_Validator\_Load(object sender, EventArgs e)

{

// datetime today

// DateTime currentDate = DateTime.Now;

// write text + current date

label1.Text += DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

;

// start time when Load Form

formLoadTime = DateTime.Now;

timer1.Start();

}

private void IP4\_Validator\_FormClosing(object sender, FormClosingEventArgs e)

{

}

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

{

textBox1.Text = textBox1.Text.Trim();

try

{

string title = "alid IP";

if (ValidIP(textBox1.Text.Trim()))

{

MessageBox.Show(textBox1.Text.Trim() + "\nThe IP is correct", "V"+title, MessageBoxButtons.OK);

createBinaryFile(dir, file, (textBox1.Text.Trim() + "|" + Regex.Replace(label1.Text.Trim(),"Today is: ","")));

}

else

{

MessageBox.Show(textBox1.Text + "\nThe IP must have 4 bytes\ninteger number between 0 to 255\nsepareted by a dot (255.255.255.255)", "Inv" + title, MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

catch (Exception ex)

{

Console.WriteLine("Exception: " + ex.Message);

}

}

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

{

string title = "IP4\_Validator";

if (MessageBox.Show("Do you want to quit the application \n" + title + "?", "Exit " + title, MessageBoxButtons.YesNo, MessageBoxIcon.Question).ToString() == "Yes")

{

this.Close();

// after close the form, stop the counter and show the total

timer1.Stop();

MessageBox.Show(label3.Text, "Total time today");

}

}

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

{

textBox1.Text = string.Empty;

textBox1.Focus();

}

private void timer1\_Tick(object sender, EventArgs e)

{

// stop time when Closing Form

DateTime formCloseTime = DateTime.Now;

// diff stop & start time

TimeSpan timeSpent = formCloseTime - formLoadTime;

int totalSec = (int)timeSpent.TotalSeconds;

int totalMin = (int)timeSpent.TotalMinutes;

label3.Text = string.Format("Total time today: {0:D2}:{1:D2}", totalMin, totalSec % 60);

}

public void createBinaryFile(string dir, string file, string txt)

{

// create filepath

string path = Path.Combine(dir, file);

// declarating the filestream and writer as binary

FileStream fs = null;

BinaryWriter binaryOut = null;

try

{

if (!File.Exists(path))

{

fs = new FileStream(path, FileMode.OpenOrCreate);

}

fs = new FileStream(path, FileMode.Append, FileAccess.Write);

// create the output stream for a binary file that exists

binaryOut = new BinaryWriter(fs);

// write the fields into text file

binaryOut.Write(txt);

// close the output stream for the text file

binaryOut.Close();

fs.Close();

}

catch (IOException ex)

{

MessageBox.Show("Error \n" + ex.Message);

}

finally

{

if (fs != null)

{

binaryOut.Close();

fs.Close();

}

}

}

}

}

1. ShowModal

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace WinFormsFinalProject

{

public partial class ShowMsg : Form

{

public ShowMsg()

{

InitializeComponent();

}

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

{

this.DialogResult = DialogResult.OK;

this.Close();

}

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

{

//string msg = "";

textBox1.Multiline = true;

textBox1.ScrollBars = ScrollBars.Vertical;

}

}

}

1. Class Calculator

using System;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Windows.Forms.VisualStyles.VisualStyleElement;

namespace WinFormsFinalProject

{

internal class Calculator

{

//private fields

private decimal currentValue;

private decimal operand1;

private decimal operand2;

private string op;

//properties

public decimal CurrentValue

{

get { return currentValue; }

set { currentValue = value; }

}

public decimal Operand1

{

get { return operand1; }

set { operand1 = value; }

}

public decimal Operand2

{

get { return operand2; }

set { operand2 = value; }

}

public string Op

{

get { return op; }

set { op = value; }

}

//Methods

public decimal Add()

{

currentValue = operand1 + operand2;

return currentValue;

}

public decimal Sub()

{

currentValue = operand1 - operand2;

return currentValue;

}

public decimal Mul()

{

currentValue = operand1 \* operand2;

return currentValue;

}

public decimal Div()

{

currentValue = Math.Round(operand1 / operand2, 16);

return currentValue;

}

public void Equals()

{

switch (Op)

{

case "+":

{

Add();

break;

}

case "-":

{

Sub();

break;

}

case "\*":

{

Mul();

break;

}

case "/":

{

Div();

break;

}

}

}

public void Clear()

{

currentValue = 0;

operand1 = 0;

operand2 = 0;

op = null;

}

}

}

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

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
| 1. string randSeq(int range) | Generate a random sequence of number to use into textbox (to play lotto), parameter range is the quantity of unique numbers are generated. |
| 1. string randControl(int range) | Generate a random sequence of number to use into the label (to control), parameter range is the quantity of numbers are generated. |
| 1. void createFile(string dir, string file, string name, string txt) | Used to generate and save the content into the text file. |
| 1. void showMsgModal(string title, string msg) | Used to create a Showmodal Form, to show the content read of txt files. |
| 1. void rbCheckedChanged(object sender, EventArgs e) | A general method called by all RadioButtons, instead of each calling its own. This is used to simplify manipulation and control of changes between radiobuttons. |
| 1. bool ValidIP(string ip) | Used to validate by REGEX, the IP V4 inserted in the textbox. |
| 1. void createBinaryFile(string dir, string file, string txt) | Used to generate and save the content into the binary file. |
| 1. void enableOpBtn(bool enable) | Used to enable/disable the operations buttons |
| 1. void btnNum\_Click(object sender, EventArgs e) | This method are used for all numbers buttons, instead of each calling its own. This is used to simplify manipulation and control of changes between buttons the same function. |
| 1. void btnOp\_Click(object sender, EventArgs e) | This method are used for all operations buttons, instead of each calling its own. This is used to simplify manipulation and control of changes between buttons the same function. |
| 1. void saveOnFile(decimal printOp1, string printOp, decimal printOp2) | Used to generate and save the operations from calculator into the text file. |
| 1. Class Calculator | In this class there are all the methods to implement the Calculator’s operations (Add,Sub, Mul, Div). |
| 1. decimal Add() | Used to provide the sum operation |
| 1. decimal Sub() | Used to provide the subtration operation |
| 1. decimal Mul() | Used to provide the multiplication operation |
| 1. decimal Div() | Used to provide the divide operation |
| 1. void Equals() | After inserted both operands, this is used to provide the final result of operation. This call Add(), Sub(), Mul(), Div(). |
| 1. void Clear() | Used to clean the operand1, operand2, operator and currentValue, used into the class calculator. |
| 1. decimal CurrentValue | Used to access the current value, result the manipulations from the operations, from the Form Simple Calculator |
| 1. decimal Operand1 | Used to access the value from operand1 |
| 1. decimal Operand2 | Used to access the value from operand2 |
| 1. string Op | Used to access the value from op (operation) |

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

In general, the difficulties encountered were due to the lack of programming experience.

Items highlighted by difficulty were:

- In Calculator, I tried to use the keyboard to activate the numbers, but this disabled the read-only function of the textbox, instead of just executing the shortcut for the button.

- Saving in TXT and BIN files was also difficult, mainly due to the text formatting.

- In the display, Read File, the screen display via messagebox was also not possible in a friendly way, as it was necessary to display the file's content in portions. For that I created a showmodal form with a multiline and scroll textbox.

- When converting currencies, the selection between radiobox was also difficult, as they are different objects, so their control is not automatic, requiring the creation of a new object to manipulate them.

- In the temperature conversion, the difficulty was due to the textbox object not accepting the change of text color and bold formatting, being necessary to use a label component.

1. **Content version**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Version | Type of Content | Form / File | Description | Date |
| 1 | Create | Main | Create form main, including TabControl |  |
| Create | Technical Manual | Create and write first descriptions |  |
| 2 | Create | LottoMax | Create form LottoMax and its funcionalities |  |
| Create | Lotto649 | Create forma Lotto649 Convertion and its funcionalities |  |
| 2.1 | Create | MoneyConv | Create forma Money Convertion and its funcionalities |  |
| Correction | Main | Changed the text showed in each messagebox |  |
| Correction | LottoMax | Changed the text showed in each messagebox |  |
| Correction | Lotto649 | Changed the text showed in each messagebox |  |
| 2.2 | Create | TempConv | Create forma Temperature Convertion and its funcionalities |  |
| Create | ShowModal | Create form Show Modal to substitute MessageBox |  |
| Correction | Main | Position on screen, centered |  |
| Correction | Lotto Max | Position on screen, center with parent |  |
| Correction | Lotto 649 | Position on screen, center with parent |  |
| Correction | MoneyConv | Position on screen, center with parent |  |
| Correction | TempConv | Position on screen, center with parent |  |
| Correction | Main | FormBorderStyle: FixedSingle |  |
| Correction | Lotto Max | FormBorderStyle: FixedSingle |  |
| Correction | Lotto 649 | FormBorderStyle: FixedSingle |  |
| Correction | MoneyConv | FormBorderStyle: FixedSingle |  |
| Correction | TempConv | FormBorderStyle: FixedSingle |  |
| 3.0 | Create | SimpleCalculator | Create form SimpleCalculator and its functionalities, with the prior modifications yet. |  |
| 4.0 | Create | IP4\_Validator | Create form IP4\_Validator and its functionalities, including saving on Binary file and showing the time online on this form, with the prior modifications yet. | 2023-04-05 |
| Correction | SimpleCalculator | Div Function was writing wrong value into txt file. | 2023-04-05 |
| 4.1 | Correction | SimpleCalculator | Error: after the second operand, if clicked at a new operation button, it ignores the first operand and the first operation. | 2023-04-11 |
| Correction | TempConv | Error: inputting values, if the signal – or + is after the value, will be working | 2023-04-11 |
| Correction | Main | Rename buttons: btnCalculator and btnIP | 2023-04-11 |
| Add | SimpleCalculator | Add a textbox to show calculation history | 2023-04-11 |
| Correction | SimpleCalculator | Clean the textbox after concluding the operation | 2023-04-11 |
| 4.2 | Correction | LottoMax | Error Message in the Try/Catch | 2023-04-13 |
| Correction | Lotto649 | Error Message in the Try/Catch | 2023-04-13 |
| 5.0 | Correction | Calculator | Enable buttons after clear(); | 2023-04-15 |
| Change | LottoMax | Change the use of MessageBox to Form ShowModal to exhibition of Read File | 2023-04-15 |
| Change | Lotto649 | Change the use of MessageBox to Form ShowModal to exhibition of Read File | 2023-04-15 |
| Change | MoneyConv | Change the use of MessageBox to Form ShowModal to exhibition of Read File | 2023-04-15 |
| Finalization | Technical Manual | Edit and Finalization the text. | 2023-04-15 |