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

|  |
| --- |
| Your name: Kathleen Forgiarini da Silva  3/24/2023  Version: 1.9 |

1. **Start by adding a short description of your project, and the languages (technologies) used:**
2. Language: C#
3. Tools (IDE): Visual Studio 2022
4. **Present the print screens of yours forms, and have a detailed description of the functionalities (step by step).**
5. When initiate the application, you will see the dashboard. Interface gráfica do usuário, Aplicativo

   Descrição gerada automaticamente
6. If you click on Exit button, it will show a message to confirm if you want to exit.

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

1. On tab “Generated Numbers”, it has two buttons with images, one button for Lotto Max and another one for Lotto649.
2. When click on Lotto Max button, it will open a dialog.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Generate button, it will generate 7 ~~unique~~ (I changed in this version, because it does not need to be unique anymore) random numbers from 0-9 under the image, and another 8 random unique numbers from 1-49 in the textbox. In addition, a data will be written to a .txt file, with the name of the lottery, the date, and the numbers.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

1. Clicking on Read File button, it will read the file generated and show the data in a Message Box, 10 records at a time.

Texto

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask if you want to exit this form. If yes, it will return to the dashboard.

Interface gráfica do usuário, Aplicativo, Site

Descrição gerada automaticamente

1. When click on Lotto649 button, it will open a dialog.

Interface gráfica do usuário, Aplicativo, Word

Descrição gerada automaticamente

1. Clicking on Generate button, it will generate 7 ~~unique~~ (I changed in this version, because it does not need to be unique anymore) random numbers from 0-9 under the image, and another 8 random unique numbers from 1-49 in the textbox. In addition, a data will be written to a .txt file, with the name of the lottery, the date, and the numbers.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

1. Clicking on Read File button, it will read the file generated and show the data in a Message Box, 10 records at a time.

Texto

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask if you want to exit this form. If yes, it will return to the dashboard.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto, Site

Descrição gerada automaticamente

1. On tab “Conversions”, it has two buttons with images, one button for Money Exchange and another one for Temperature Convert.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Money Exchange button, it will open the form to convert money.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask if you want to quit. If yes, it will show how much time you spent in this form.

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Convert button it will validate the value entered and the selected currencies, and display the result. In addition, it will save the conversions into a .txt file.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Read File button, it will read the file generated and show the data in a Message Box, at least 10 records at a time.

Texto

Descrição gerada automaticamente

1. On tab “Conversions”, clicking on temperature button, it will open a form to convert temperatures.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask if you want to quit.

Interface gráfica do usuário, Texto, Aplicativo

Descrição gerada automaticamente

1. If you select the option “from C to F” and press Convert, it will calculate the temperature you entered in Celsius to Fahrenheit. If you select the option “from F to C” (the labels of the textboxes will change) and press Convert, it will calculate the temperature you entered in Fahrenheit to Celsius. Depending on the temperature inserted, a message will appear. Every conversion made will be saved in a text file called TempConv.txt.

Also, the “Message” textbox should be read-only, but with this property, the color doesn’t work, so I changed the textbox for a label.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Read File button, it will read the file generated and show the data in a Message Box, 10 records at a time.

Texto

Descrição gerada automaticamente

1. On tab “Simple Calculator”, it has one button with image to open the calculator.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on the calculator, it will open a form with a calculator.

Interface gráfica do usuário, Texto

Descrição gerada automaticamente

1. As we click on the buttons, the textbox automatically changes, doing the operations. Also, it will save each operation into the file Calculator.txt.

Tela de computador com fundo branco

Descrição gerada automaticamenteInterface gráfica do usuário

Descrição gerada automaticamenteTela de computador com fundo branco

Descrição gerada automaticamente com confiança médiaTela de computador com texto preto sobre fundo branco

Descrição gerada automaticamenteInterface gráfica do usuário, Texto

Descrição gerada automaticamente

Texto

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask you if you want to exit.

Interface gráfica do usuário, Texto, Aplicativo, Site

Descrição gerada automaticamente

1. On tab “IP v4 validator”, it has one button with image to open the validator of IPs.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on the image, it will open a form for IP Validator. When it is loaded, it will show the current date.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Exit button, it will ask if you want to exit and show you the time you spent in this form.

Interface gráfica do usuário, Site

Descrição gerada automaticamenteInterface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

1. Clicking on Reset button, it will erase any entered text.
2. Clicking on Validate button, it will test the IP entered in the textbox, and show if it is a valid IP or not. If it is a valid IP, it will save it in a binary file.

Interface gráfica do usuário, Aplicativo

Descrição gerada automaticamente

Interface gráfica do usuário, Texto, Aplicativo, chat ou mensagem de texto

Descrição gerada automaticamente

Tela de computador

Descrição gerada automaticamente

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

**MAIN.CS**

public partial class Main : Form

{

public Main()

{

InitializeComponent();

}

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

{

if (MessageBox.Show("Are you sure you want to exit?", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

Application.Exit();

}

}

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

{

}

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

{

LottoMax obj = new LottoMax();

obj.ShowDialog();

}

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

{

Lotto649 obj = new Lotto649();

obj.ShowDialog();

}

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

{

MoneyExchange obj = new MoneyExchange();

obj.ShowDialog();

}

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

{

Temperature obj = new Temperature();

obj.ShowDialog();

}

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

{

Calculator obj = new Calculator();

obj.ShowDialog();

}

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

{

IPValidator obj = new IPValidator();

obj.ShowDialog();

}

}

**LOTTOMAX.CS**

public partial class LottoMax : Form

{

public LottoMax()

{

InitializeComponent();

}

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

{

if (MessageBox.Show("Do you want to quit?", "Exit Lotto Max", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

string path = @".\files\LottoNbrs.txt";

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

{

Random labelRandom = new Random();

int labelRandomNumber = 0;

List<int> labelUniqueNumbers = new List<int>();

string labelNumbers = "";

while (labelUniqueNumbers.Count < 7)

{

labelRandomNumber = labelRandom.Next(0, 10);

labelUniqueNumbers.Add(labelRandomNumber);

labelNumbers += labelRandomNumber;

}

maxLabel.Text = labelNumbers;

Random boxRandom = new Random();

int boxRandomNumber = 0;

List<int> boxUniqueNumbers = new List<int>();

string boxNumbers = "";

while (boxUniqueNumbers.Count < 8)

{

boxRandomNumber = boxRandom.Next(1, 50);

if (!boxUniqueNumbers.Contains(boxRandomNumber))

{

boxUniqueNumbers.Add(boxRandomNumber);

boxNumbers += boxRandomNumber.ToString() + "\t\t";

}

}

maxTextbox.Text = boxNumbers;

//Text File

string dir = @".\files\";

FileStream fileStream = null;

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

string lotteryName = "Max";

string dateTimeString = DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

int bonusNumber = boxUniqueNumbers[7];

writer.Write(lotteryName + ", " + dateTimeString + ", ");

for (int i = 0; i < boxUniqueNumbers.Count - 1; i++)

{

writer.Write(boxUniqueNumbers[i]);

if (i != boxUniqueNumbers.Count - 1)

{

writer.Write(",");

}

}

writer.WriteLine(" Bonus " + bonusNumber);

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

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

{

FileStream fileStream = null;

StreamReader reader = null;

string textToPrint = "";

int counter = 0;

string title = "Lottery Numbers by Kathleen Forgiarini";

try

{

fileStream = new FileStream(path, FileMode.Open, FileAccess.Read);

reader = new StreamReader(fileStream);

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

{

textToPrint += reader.ReadLine() + "\n";

counter++;

if (counter == 10)

{

MessageBox.Show(textToPrint, title);

textToPrint = "";

counter = 0;

}

}

if (counter != 0)

{

MessageBox.Show(textToPrint, title);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

}

**LOTTO649.CS**

public partial class Lotto649 : Form

{

public Lotto649()

{

InitializeComponent();

}

private void l649Exit\_Click(object sender, EventArgs e)

{

if (MessageBox.Show("Do you want to quit?", "Exit Lotto649", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

string path = @".\files\LottoNbrs.txt";

private void l649Generate\_Click(object sender, EventArgs e)

{

Random labelRandom = new Random();

int labelRandomNumber = 0;

List<int> labelUniqueNumbers = new List<int>();

string labelNumbers = "";

while (labelUniqueNumbers.Count < 7)

{

labelRandomNumber = labelRandom.Next(0, 10);

labelUniqueNumbers.Add(labelRandomNumber);

labelNumbers += labelRandomNumber;

}

l649Label.Text = labelNumbers;

Random boxRandom = new Random();

int boxRandomNumber = 0;

List<int> boxUniqueNumbers = new List<int>();

string boxNumbers = "";

while (boxUniqueNumbers.Count < 7)

{

boxRandomNumber = boxRandom.Next(1, 49);

if (!boxUniqueNumbers.Contains(boxRandomNumber))

{

boxUniqueNumbers.Add(boxRandomNumber);

boxNumbers += boxRandomNumber.ToString() + "\t\t";

}

}

l649Textbox.Text = boxNumbers;

//Text File

string dir = @".\files\";

FileStream fileStream = null;

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

string lotteryName = "649";

int bonusNumber = boxUniqueNumbers[6];

string dateTimeString = DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

writer.Write(lotteryName + ", " + dateTimeString + ", ");

for (int i = 0; i < boxUniqueNumbers.Count - 1; i++)

{

writer.Write(boxUniqueNumbers[i]);

if (i != boxUniqueNumbers.Count - 1)

{

writer.Write(",");

}

}

writer.WriteLine(" Bonus " + bonusNumber);

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

private void l649Read\_Click(object sender, EventArgs e)

{

FileStream fileStream = null;

StreamReader reader = null;

string textToPrint = "";

int counter = 0;

string title = "Lottery Numbers by Kathleen Forgiarini";

try

{

fileStream = new FileStream(path, FileMode.Open, FileAccess.Read);

reader = new StreamReader(fileStream);

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

{

textToPrint += reader.ReadLine() + "\n";

counter++;

if (counter == 10)

{

MessageBox.Show(textToPrint, title);

textToPrint = "";

counter = 0;

}

}

if (counter != 0)

{

MessageBox.Show(textToPrint, title);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

}

**MONEYEXCHANGE.CS**

public partial class MoneyExchange : Form

{

private const double cadRate = 1.0;

private const double usdRate = 0.7289;

private const double eurRate = 0.6845;

private const double gbpRate = 0.5997;

private const double brlRate = 3.8265;

private DateTime startTime;

private DateTime endTime;

public MoneyExchange()

{

InitializeComponent();

}

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

{

if (MessageBox.Show("Do you want \nto quit the application \nMoney Exchange?", "Exit ?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

string path = @".\files\MoneyConv.txt";

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

{

double amount;

double result;

string amountStr = fromTextbox.Text.Trim();

if (Regex.IsMatch(amountStr, @"^\d{1,3}(,\d{3})\*(\.\d+)?$"))

{

try

{

amount = Convert.ToDouble(fromTextbox.Text);

}

catch (FormatException)

{

MessageBox.Show("Invalid input amount.");

return;

}

string from = GetSelectedCurrency(cad, usd, eur, gbp, brl);

if (from == null)

{

MessageBox.Show("Please select a FROM currency.");

return;

}

string to = GetSelectedCurrency(toCad, toUsd, toEur, toGbp, toBrl);

if (to == null)

{

MessageBox.Show("Please select a TO currency.");

return;

}

switch (to)

{

case "CAD":

result = amount \* cadRate / GetCurrencyRate(from);

break;

case "USD":

result = amount \* usdRate / GetCurrencyRate(from);

break;

case "EUR":

result = amount \* eurRate / GetCurrencyRate(from);

break;

case "GBP":

result = amount \* gbpRate / GetCurrencyRate(from);

break;

case "BRL":

result = amount \* brlRate / GetCurrencyRate(from);

break;

default:

MessageBox.Show("Invalid currency selection.");

return;

}

toTextbox.Text = result.ToString("0.00");

fromTextbox.Focus();

//Text File

FileStream fileStream = null;

try

{

string dir = @".\files\";

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

string dateTimeString = DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

writer.WriteLine($"{amount} {from} = {Math.Round(result)} {to}, {dateTimeString}");

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

else

{

MessageBox.Show("Invalid amount entered. Please enter a valid amount with a comma or dot as decimal separator.");

}

}

private string GetSelectedCurrency(params RadioButton[] radioButtons)

{

foreach (RadioButton rb in radioButtons)

{

if (rb.Checked)

{

return rb.Text;

}

}

return null;

}

private double GetCurrencyRate(string currency)

{

switch (currency)

{

case "CAD":

return cadRate;

case "USD":

return usdRate;

case "EUR":

return eurRate;

case "GBP":

return gbpRate;

case "BRL":

return brlRate;

default:

return 0.0;

}

}

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

{

string message = "";

FileStream fileStream = null;

StreamReader reader = null;

string textToPrint = "";

int counter = 0;

string title = "Money Exchange by Kathleen Forgiarini";

try

{

fileStream = new FileStream(path, FileMode.Open, FileAccess.Read);

reader = new StreamReader(fileStream);

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

{

textToPrint += reader.ReadLine() + "\n";

counter++;

if (counter == 10)

{

MessageBox.Show(textToPrint, title);

textToPrint = "";

counter = 0;

}

}

if (counter != 0)

{

MessageBox.Show(textToPrint, title);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

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

{

startTime = DateTime.Now;

}

private void MoneyExchange\_FormClosing(object sender, FormClosingEventArgs e)

{

endTime = DateTime.Now;

TimeSpan interval = endTime - startTime;

int totalSeconds = Convert.ToInt32(interval.TotalSeconds);

int totalMinutes = totalSeconds / 60;

totalSeconds %= 60;

if (totalMinutes >= 2)

{

MessageBox.Show($"You used the money exchange form for {totalMinutes} minutes and {totalSeconds} seconds", "Time Spent");

} else if (totalMinutes == 0)

{

MessageBox.Show($"You used the money exchange form for {totalSeconds} seconds", "Time Spent");

} else

{

MessageBox.Show($"You used the money exchange form for {totalMinutes} minute and {totalSeconds} seconds", "Time Spent");

}

}

}

**TEMPERATURE.CS**

public partial class Temperature : Form

{

public Temperature()

{

InitializeComponent();

}

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

{

if (cTof.Checked)

{

labelFrom.Text = "C";

labelTo.Text = "F";

}

else

{

labelFrom.Text = "F";

labelTo.Text = "C";

}

fromTextbox.Text = "";

toTextbox.Text = "";

message.Text = "";

}

string path = @".\files\TempConv.txt";

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

{

string valueStr = fromTextbox.Text.Trim();

double result = 0;

double inTemperature;

string outMessage = "";

string dir = @".\files\";

if (Regex.IsMatch(valueStr, @"^-?\d{1,3}(\.\d{1,2})?(°[CF])?$"))

{

try

{

inTemperature = Convert.ToDouble(fromTextbox.Text);

}

catch (FormatException)

{

MessageBox.Show("Invalid temperature.");

return;

}

if (cTof.Checked)

{

result = inTemperature \* 9 / 5 + 32;

switch (inTemperature)

{

case 100:

message.ForeColor = Color.Red;

outMessage = "Water boils";

break;

case 40:

message.ForeColor = Color.Red;

outMessage = "Hot Bath";

break;

case 37:

message.ForeColor = Color.Green;

outMessage = "Body temperature";

break;

case 30:

message.ForeColor = Color.Green;

outMessage = "Beach weather";

break;

case 21:

message.ForeColor = Color.Green;

outMessage = "Room remperature";

break;

case 10:

message.ForeColor = Color.Blue;

outMessage = "Cool Day";

break;

case 0:

message.ForeColor = Color.Blue;

outMessage = "Freezing point of water";

break;

case -18:

message.ForeColor = Color.Blue;

outMessage = "Very Cold Day";

break;

case -40:

message.ForeColor = Color.Blue;

outMessage = "Extremely Cold Day \t\t\t\t\t(and the same number!)";

break;

default:

outMessage = "";

break;

}

FileStream fileStream = null;

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

string dateTimeString = DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

writer.WriteLine($"{inTemperature} Celsius = {Math.Round(result, 1)} Fahrenheit, {dateTimeString} {Regex.Replace(outMessage, @"\s+", " ")}");

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

else

{

result = (inTemperature - 32) \* 5 / 9;

switch (inTemperature)

{

case 212:

message.ForeColor = Color.Red;

outMessage = "Water boils";

break;

case 104:

message.ForeColor = Color.Red;

outMessage = "Hot Bath";

break;

case 98.6:

message.ForeColor = Color.Green;

outMessage = "Body temperature";

break;

case 86:

message.ForeColor = Color.Green;

outMessage = "Beach weather";

break;

case 70:

message.ForeColor = Color.Green;

outMessage = "Room remperature";

break;

case 50:

message.ForeColor = Color.Blue;

outMessage = "Cool Day";

break;

case 32:

message.ForeColor = Color.Blue;

outMessage = "Freezing point of water";

break;

case 0:

message.ForeColor = Color.Blue;

outMessage = "Very Cold Day";

break;

case -40:

message.ForeColor = Color.Blue;

outMessage = "Extremely Cold Day \t\t\t\t\t(and the same number!)";

break;

default:

outMessage = "";

break;

}

FileStream fileStream = null;

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

string dateTimeString = DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt");

writer.WriteLine($"{inTemperature} Fahrenheit = {Math.Round(result, 1)} Celsius, {dateTimeString} {Regex.Replace(outMessage, @"\s+", " ")}");

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

toTextbox.Text = Math.Round(result,2).ToString();

message.Text = outMessage;

fromTextbox.Focus();

}

else

{

MessageBox.Show("Invalid value entered. Please enter a valid temperature using dot as decimal separator.");

}

}

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

{

if (MessageBox.Show("Do you want to quit the temperature converter?", "Exit ?", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

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

{

string message = "";

FileStream fileStream = null;

StreamReader reader = null;

string textToPrint = "";

int counter = 0;

string title = "Temperature Converter by Kathleen Forgiarini";

try

{

fileStream = new FileStream(path, FileMode.Open, FileAccess.Read);

reader = new StreamReader(fileStream);

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

{

textToPrint += reader.ReadLine() + "\n";

counter++;

if (counter == 10)

{

MessageBox.Show(textToPrint, title);

textToPrint = "";

counter = 0;

}

}

if (counter != 0)

{

MessageBox.Show(textToPrint, title);

}

reader.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

}

**CALCULATOR.CS**

public partial class Calculator : Form

{

private decimal currentValue;

private decimal operand1;

private decimal operand2;

private string op;

private bool isOperationClicked = false;

public Calculator()

{

InitializeComponent();

Clear();

}

public decimal CurrentValue

{

get { return Math.Round(currentValue,8); }

}

public void Add(decimal displayValue)

{

operand1 = displayValue;

currentValue = displayValue;

op = "+";

}

public void Subtract(decimal displayValue)

{

operand1 = displayValue;

currentValue = displayValue;

op = "-";

}

public void Multiply(decimal displayValue)

{

operand1 = displayValue;

currentValue = displayValue;

op = "\*";

}

public void Divide(decimal displayValue)

{

operand1 = displayValue;

currentValue = displayValue;

op = "/";

}

public void Equals()

{

switch (op)

{

case "+":

currentValue = operand1 + operand2;

break;

case "-":

currentValue = operand1 - operand2;

break;

case "\*":

currentValue = operand1 \* operand2;

break;

case "/":

if (operand2 == 0)

{

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

}

else

{

currentValue = operand1 / operand2;

}

break;

}

string path = @".\files\Calculator.txt";

string dir = @".\files\";

FileStream fileStream = null;

try

{

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

StreamWriter writer = new StreamWriter(fileStream);

writer.WriteLine($"{operand1} {op} {operand2} = {CurrentValue}");

writer.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again. \n" + ex.Message);

}

finally

{

if (fileStream != null) fileStream.Close();

}

operand1 = currentValue;

}

public void Equals(decimal displayValue)

{

operand2 = displayValue;

Equals();

}

public void Clear()

{

currentValue = 0;

operand1 = 0;

operand2 = 0;

op = null;

textBox.Text = "0";

}

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

{

if (MessageBox.Show("Do you want to quit the calculator?", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

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

{

Clear();

}

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

{

if (!textBox.Text.Contains("."))

{

textBox.Text += ".";

isOperationClicked = false;

}

}

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

{

if (isOperationClicked)

{

MessageBox.Show("Select another number");

}

else

{

Add(decimal.Parse(textBox.Text));

textBox.Text = "+";

isOperationClicked = true;

}

}

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

{

if (isOperationClicked)

{

MessageBox.Show("Select another number");

}

else

{

Subtract(decimal.Parse(textBox.Text));

textBox.Text = "-";

isOperationClicked = true;

}

}

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

{

if (isOperationClicked)

{

MessageBox.Show("Select another number");

}

else

{

Multiply(decimal.Parse(textBox.Text));

textBox.Text = "\*";

isOperationClicked = true;

}

}

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

{

if (isOperationClicked)

{

MessageBox.Show("Select another number");

}

else

{

Divide(decimal.Parse(textBox.Text));

textBox.Text = "/";

isOperationClicked = true;

}

}

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

{

try

{

Equals(decimal.Parse(textBox.Text));

textBox.Text = CurrentValue.ToString();

operand1 = 0;

operand2 = 0;

op = null;

}

catch

{

MessageBox.Show("An error occured, please review the operations entered!", "Error");

Clear();

}

}

private void btn1\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "1";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "1";

}

else

{

textBox.Text += "1";

}

}

private void btn2\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "2";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "2";

}

else

{

textBox.Text += "2";

}

}

private void btn3\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "3";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "3";

}

else

{

textBox.Text += "3";

}

}

private void btn4\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "4";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "4";

}

else

{

textBox.Text += "4";

}

}

private void btn5\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "5";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "5";

}

else

{

textBox.Text += "5";

}

}

private void btn6\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "6";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "6";

}

else

{

textBox.Text += "6";

}

}

private void btn7\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "7";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "7";

}

else

{

textBox.Text += "7";

}

}

private void btn8\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "8";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "8";

}

else

{

textBox.Text += "8";

}

}

private void btn9\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "9";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "9";

}

else

{

textBox.Text += "9";

}

}

private void btn0\_Click(object sender, EventArgs e)

{

if (isOperationClicked)

{

textBox.Text = "0";

isOperationClicked = false;

}

else if (textBox.Text == "0")

{

textBox.Text = "0";

}

else

{

textBox.Text += "0";

}

}

}

**IPVALIDATOR.CS**

public partial class IPValidator : Form

{

public IPValidator()

{

InitializeComponent();

}

private DateTime startTime;

private DateTime endTime;

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

{

if (MessageBox.Show("Do you want to quit the IP4 Validator?", "Exit", MessageBoxButtons.YesNo).ToString() == "Yes")

{

this.Close();

}

}

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

{

labelDate.Text += "Today: " + DateTime.Today.ToLongDateString().ToString();

startTime = DateTime.Now;

}

string pathBinary = @"C:.\files\BIPValidator.dat";

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

{

string ip = textBox.Text.Trim();

Regex ipRegex = new Regex(@"^((25[0-5]|(2[0-4]|1\d|[1-9]|)\d)\.?\b){4}$");

try

{

if (ip == "")

{

MessageBox.Show("Enter an IP to validate!");

} else

{

if (ipRegex.IsMatch(ip))

{

MessageBox.Show(ip + "\nThe IP is correct", "Valid IP");

FileStream fileStream = null;

try

{

string dir = @".\files\";

if (!Directory.Exists(dir))

{

Directory.CreateDirectory(dir);

}

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

BinaryWriter binaryOut = new BinaryWriter(fileStream);

binaryOut.Write(textBox.Text.Trim());

binaryOut.Write(DateTime.Now.ToString("yyyy/MM/dd h:mm:ss tt"));

binaryOut.Close();

fileStream.Close();

}

catch (Exception ex)

{

MessageBox.Show("An error occurred while writing to the file. Try again!\n" + ex.Message, "Error");

}

finally

{

if (fileStream != null) fileStream.Close();

}

}

else

{

MessageBox.Show(ip + "\nThe IP must have 4 bytes \ninteger number between 0 to 255 \nseparated by a dot (255.255.255.255)", "Error");

}

}

textBox.Focus();

}

catch (Exception ex)

{

MessageBox.Show("An error occured, try again!\n" + ex.Message);

}

}

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

{

textBox.Text = "";

textBox.Focus();

}

private void IPValidator\_FormClosing(object sender, FormClosingEventArgs e)

{

endTime = DateTime.Now;

TimeSpan interval = endTime - startTime;

int totalSeconds = Convert.ToInt32(interval.TotalSeconds);

int totalMinutes = totalSeconds / 60;

totalSeconds %= 60;

if (totalMinutes >= 2)

{

MessageBox.Show($"You used the IP4 validator for {totalMinutes} minutes and {totalSeconds} seconds", "Time Spent");

}

else if (totalMinutes == 0)

{

MessageBox.Show($"You used the IP4 validator for {totalSeconds} seconds", "Time Spent");

}

else

{

MessageBox.Show($"You used the IP4 validator for {totalMinutes} minute and {totalSeconds} seconds", "Time Spent");

}

}

}

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

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
| 1. Main | This class has the dashboard |
| 1. mainExit\_Click | A method inside Main class to close the application |
| 1. lottoMax\_Click | A method inside Main class to open Lotto Max form |
| 1. lotto649\_Click | A method inside Main class to open Lotto649 form |
| 1. money\_Click | A method inside Main class to open MoneyExchange form |
| 1. temperature\_Click | A method inside Main class to open Temperature converter form |
| 1. calculator\_Click | A method inside Main class to open Calculator form |
| 1. ipValidator\_Click | A method inside Main class to open IP Validator form |
| 1. LottoMax | This class has the Lotto Max form |
| 1. maxExit\_Click | A method inside LottoMax class to exit the form |
| 1. maxGenerate\_Click | A method inside LottoMax to generate random numbers and save it to a .txt file |
| 1. maxRead\_Click | A method inside LottoMax to read the .txt file generated and show the data in a Message Box |
| 1. Lotto649 | This class has the Lotto649 form |
| 1. l649Exit\_Click | A method inside the Lotto649 class to exit the form |
| 1. l649Generate\_Click | A method inside the Lotto649 to generate random numbers and save it to a .txt file |
| 1. l649Read\_Click | A method inside Lotto649 to read the .txt file generated and show the data in a Message Box |
| 1. MoneyExchange | This class has the Money Exchange form |
| 1. exit\_Click | A method inside MoneyExchange class to exit the form |
| 1. convert\_Click | A method inside MoneyExchange class to convert the amount entered into the selected currency and save the conversions in a txt file |
| 1. GetSelectedCurrency | A method inside MoneyExchange class, to get the value of the selected radio button |
| 1. GetCurrencyRate | A method inside MoneyExchange class, to get the exchange factor depending on which radio button is selected |
| 1. read\_Click | A method inside MoneyExchange class, to read the .txt file generated and show the data in a Message Box |
| 1. MoneyExchange\_Load | A method inside MoneyExchange class, to get the current time when the form is loaded |
| 1. MoneyExchange\_FormClosing | A method inside MoneyExchange class, to get the current time and calculate the difference between the time the form was loaded and closed, and display the result to the user |
| 1. Temperature | This class has the Temperature form |
| 1. cTof\_CheckedChanged | A method inside Temperature class, to check which radio button is selected. It changes the text of the labels and set the textboxes to empty |
| 1. convert\_Click | A method inside Temperature class, to get the value entered, the option selected, and do the conversion from C to F and from F to C. Also, save all the conversions into a text file |
| 1. exit\_Click | A method inside Temperature class to exit the form |
| 1. read\_Click | A method inside Temperature class, to read the .txt file generated and show the data in a Message Box |
| 1. Calculator | This class has the Calculator form |
| 1. CurrentValue | A property inside the Calculator class to return the value of the private decimal currentValue, rounding to 8 decimal places |
| 1. Add | A method inside Calculator class to save the current value of the textbox, and save the operation selected |
| 1. Subtract | A method inside Calculator class to save the current value of the textbox, and save the operation selected |
| 1. Multiply | A method inside Calculator class to save the current value of the textbox, and save the operation selected |
| 1. Divide | A method inside Calculator class to save the current value of the textbox, and save the operation selected |
| 1. Equals | A method inside Calculator class that take the operation selected and the operands, and do the calculation. In addition, it saves each operation to a text file called Calculator.txt |
| 1. Equals(displayValue) | A method inside Calculator class that takes the current value of the textbox and save it to the second operand, and calls the method Equals. |
| 1. Clear | A method inside Calculator class that set all the values to its default |
| 1. exit\_Click | A method inside Calculator class to exit the form |
| 1. clear\_Click | A button event inside Calculator class to call the Clear method |
| 1. btnDot\_Click | A button event inside Calculator class representing the dot (.) |
| 1. btnAdd\_Click | A button event inside Calculator class representing the plus sign (+), that calls the method Add |
| 1. btnSub\_Click | A button event inside Calculator class representing the minus sign (-), that calls the method Subtract |
| 1. btnMul\_Click | A button event inside Calculator class representing the multiply signal (\*), that calls the method Multiply |
| 1. btnDiv\_Click | A button event inside Calculator class representing the divide signal (/), that calls the method Subtract |
| 1. btnEqual\_Click | A button event inside Calculator class that calls the Equals(displayValue) function, passing the current value of the textbox as a parameter |
| 1. btn1\_Click | A button event inside Calculator class for when click on number 1 |
| 1. btn2\_Click | A button event inside Calculator class for when click on number 2 |
| 1. btn3\_Click | A button event inside Calculator class for when click on number 3 |
| 1. btn4\_Click | A button event inside Calculator class for when click on number 4 |
| 1. btn5\_Click | A button event inside Calculator class for when click on number 5 |
| 1. btn6\_Click | A button event inside Calculator class for when click on number 6 |
| 1. btn7\_Click | A button event inside Calculator class for when click on number 7 |
| 1. btn8\_Click | A button event inside Calculator class for when click on number 8 |
| 1. btn9\_Click | A button event inside Calculator class for when click on number 9 |
| 1. btn0\_Click | A button event inside Calculator class for when click on number 0 |
| 1. IPValidator | This class has the IP Validator form |
| 1. exit\_Click | A method inside IPValidator class to exit the form |
| 1. IPValidator\_Load | A method inside IPValidator class, to get the current time when the form is loaded |
| 1. IPValidator\_FormClosing | A method inside IPValidator class, to get the current time and calculate the difference between the time the form was loaded and closed, and display the result to the user |
| 1. reset\_Click | A button event inside IPValidator class, to clear the textbox |
| 1. validate\_Click | A button event inside IPValidator class, to validate the IP entered and show appropriated messages. If the IP is valid, it will save in a binary file |

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

* It was easy to design every layouts and connect forms.
* In the generated numbers forms, I had a little bit of hard time figuring out how to get only unique numbers to display.
* I had some difficult to create the text file and read it, because it was my first time using the objects FileStream, StreamReader and StreamWriter.
* In money exchange, I had some difficult to get the value of the selected radio buttons because I have never used this input before. But it was ok to do the math conversions and display the results.
* In temperature converter, I had some troubles changing the color, because its my first time using it. It asks to change the color of the text depending on the message, also it asked for the textbox multiline to be readonly, but if it is readonly, the color method doesn’t work. Besides that, it was easy to do the temperature converter.
* In the calculator, it was a lot of work to take the click event of each button and perform the operations properly in a smooth way, handling all possible errors. But it was not hard, because we learned how to do it in class.
* In the IP Validator, it was hard to create the Regex to validate the IP, because it is big, and a lot of characters is confusing.