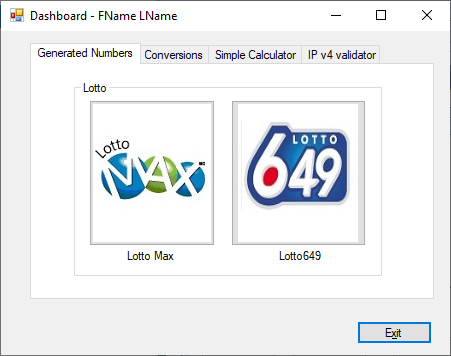
|  |
| --- |
| College LaSalle |
| Project - Oriented Object Programming User and Technical Manual |
|  |
| Presented to: Mihai Maftei. |

|  |
| --- |
| Meet Hingu  12/6/2024  Version last |

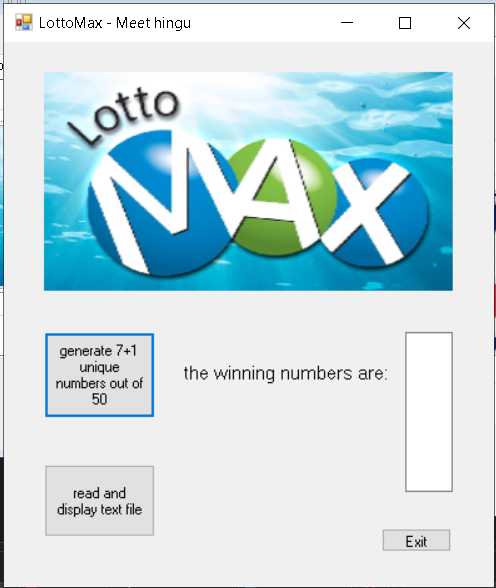
1. **Start by adding a short description of your project, and the languages (technologies) used:**

This first section of the project is about lottery and generating random numbers and display them.

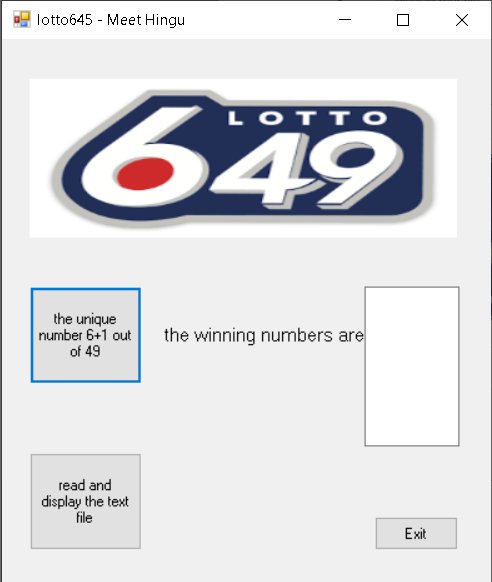
1. Language: C#
2. Used tool(s): VS2022 Windows Form (.Net Framework), MS Word, AI (ChatGPT)
3. **Present the print screens of yours forms and have a detailed description of the functionalities (step by step).**



1. If you click on tab LottoMax



1. If you click on tab Lotto649



**Lottery Number Generation**

In your project, you could create a feature where users can generate random Lotto Max or Lotto 6/49 numbers for their tickets:

* **Lotto Max**: Generate 7 random numbers from 1 to 50.
* **Lotto 6/49**: Generate 6 random numbers from 1 to 49. You could implement this in a feature like "Generate Numbers for Lotto Max" or "Generate Numbers for Lotto 6/49" where users can choose the lottery type and have the system randomly select numbers for them.

**Lottery Results**

You could allow users to input their numbers, then compare them to the winning numbers and show whether they’ve won or not. For example:

* Users input their 6 or 7 numbers.
* Display the result, showing how many numbers they matched and if they won a prize.

**Tracking and Saving Results**

If you need to save the conversion results or lottery data into a text file (as you mentioned in previous requests), you could create a logging mechanism:

* **Lotto Results Log**: Save the results of each lottery drawing, including the draw date and winning numbers, into a text file or database.
* **User Entries**: Save user-selected numbers or their results in a file like LottoEntries.txt.

**3.Money Conversion**

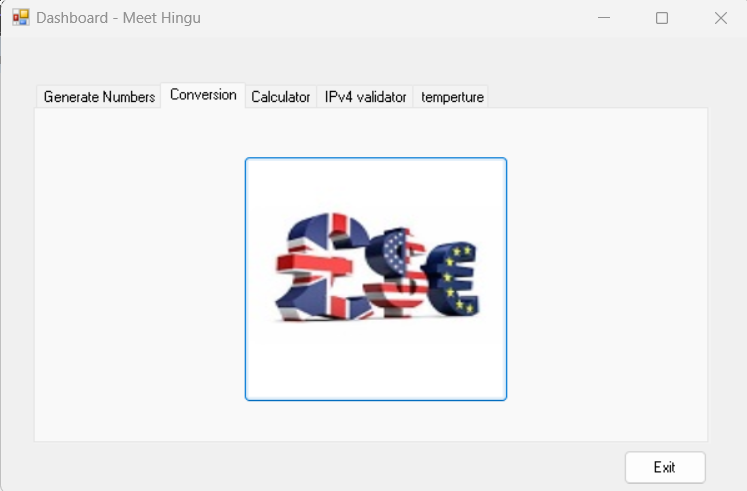
Money conversion is a process that allows the exchange of one currency into another based on the current exchange rates. In the context of your project, it seems that you're implementing a **money conversion tool** where users can input an amount in one currency (e.g., CAD, USD, EUR) and get the equivalent amount in multiple other currencies, such as USD, EUR, GBP, INR, and even Bitcoin (BTC).

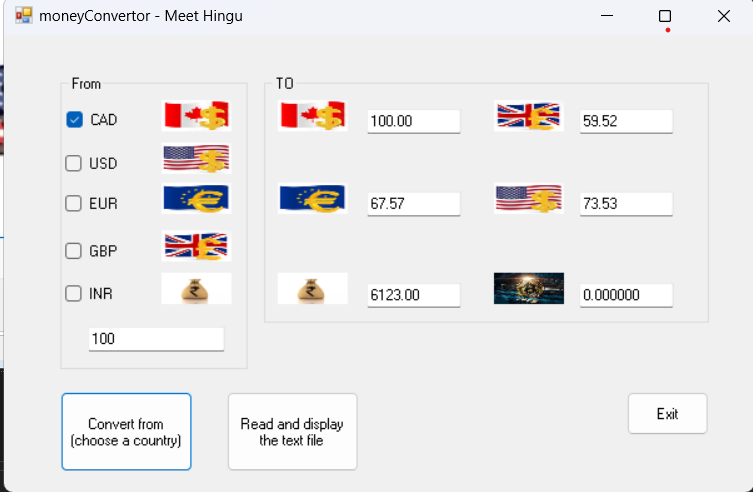
The tool could be useful for:

* Travelers who need to quickly convert their home currency to foreign currencies.
* Investors or businesses dealing with multiple currencies.
* People who want to track the value of cryptocurrencies against traditional currencies.

#### Key Features:

1. **Multiple Currencies**: The tool can convert between several currencies like **Canadian Dollar (CAD)**, **US Dollar (USD)**, **Euro (EUR)**, **British Pound (GBP)**, **Indian Rupee (INR)**, and **Bitcoin (BTC)**.
2. **Conversion Rates**: The conversion is done using predefined exchange rates that represent how much one unit of a given currency is worth in another currency. For example, 1 CAD might be worth 0.7353 USD.
3. **User Input**: The user inputs a base amount (in a given currency) and chooses the target currencies.
4. **Real-time Conversion**: Upon entering an amount and selecting the conversion type, the program can output the equivalent values in all selected currencies.
5. **Saving Conversion Results**: After each conversion, the system can save the results, including the amount entered and the converted amounts, with the current date and time, into a text file for future reference.
6. **User Interface**:
   * **Input Box**: For entering the base currency amount.
   * **Text Boxes**: For displaying the converted values in other currencies.
   * **Buttons**: To trigger the conversion process and display or save the results.
   * **Message Box**: To show results of the conversion or an error message if the input is invalid.

****

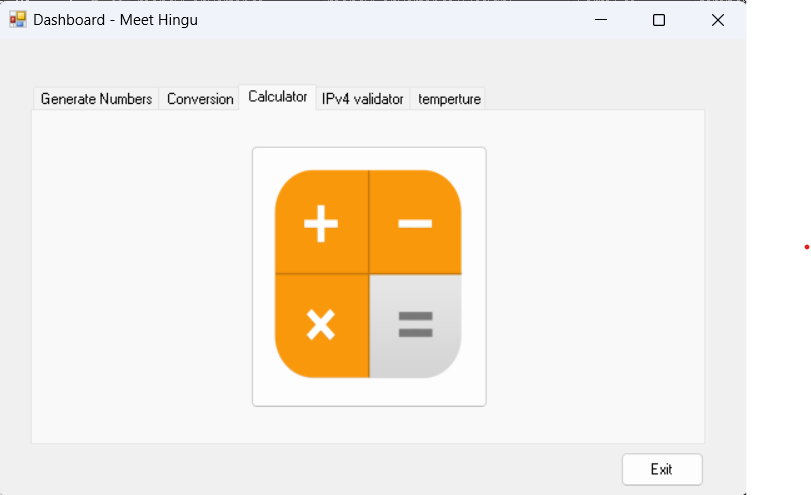
****

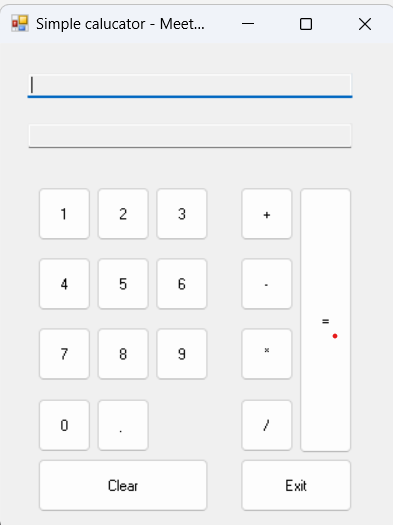
**4.Calculator**

A **calculator** is a device or application that performs mathematical operations on numbers. It allows users to input numbers and operations to calculate results based on predefined rules (such as addition, subtraction, multiplication, and division). In the context of a software calculator, it functions as a digital tool that mimics traditional handheld calculators or even advanced scientific calculators with additional functions.

#### Key Features:

1. **Basic Arithmetic Operations**:
   * **Addition (+)**: Adds two or more numbers.
   * **Subtraction (-)**: Subtracts one number from another.
   * **Multiplication (×)**: Multiplies two numbers.
   * **Division (÷)**: Divides one number by another.
2. **User Interface**:
   * **Number Buttons**: The user can enter numbers (0-9) through a set of buttons.
   * **Operation Buttons**: Buttons for basic operations like +, -, ×, ÷, and = to perform the calculations.
   * **Clear Button (C)**: Resets the input, clearing the current operation or result.
   * **Result Display**: A text box or label where the user sees the result of the operation.

****

****

5.IPv4 and IPv6

#### **IPv4 (Internet Protocol version 4)**

**IPv4** is the fourth version of the Internet Protocol (IP), and it is one of the core protocols used in the Internet and networking systems. It was developed in the early 1980s and has since been the dominant protocol for addressing and routing packets of data over networks.

##### **Key Features of IPv4:**

1. **32-bit Addressing**:
   * IPv4 uses a 32-bit address format, which allows for a total of 2^32 unique IP addresses (approximately 4.3 billion addresses).
   * These addresses are represented in **dotted decimal notation**, where the 32 bits are split into four octets (8 bits each), and each octet is written in decimal form separated by periods (e.g., 192.168.1.1).

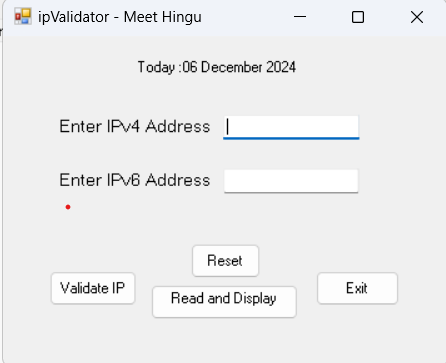
#### **IPv6 (Internet Protocol version 6)**

**IPv6** is the successor to IPv4 and was developed to address the limitations of IPv4, especially the exhaustion of IP addresses. IPv6 is the latest version of the Internet Protocol and is designed to support the growing number of devices on the Internet.

##### **Key Features of IPv6:**

1. **128-bit Addressing**:
   * IPv6 uses a 128-bit address format, which provides a vast address space of 2^128 unique IP addresses. This is equivalent to about 340 undecillion (3.4×10^38) possible addresses, ensuring there will be enough addresses for the foreseeable future.



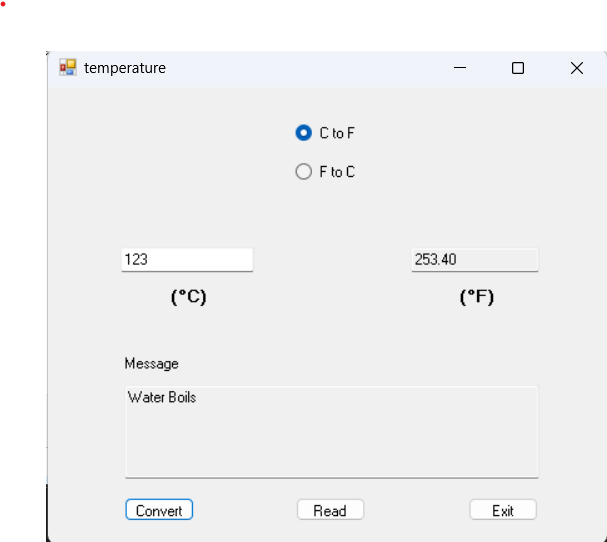
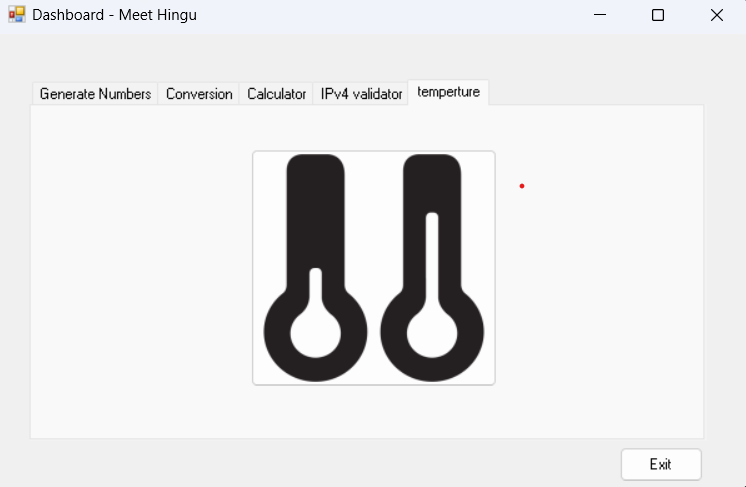


**5.Temperature**

### ****Temperature Conversion - Description****

The **temperature conversion** system is designed to convert temperatures between different units, such as Celsius and Fahrenheit. Temperature conversion is a common task in fields such as science, engineering, and everyday life, where you may need to switch between different measurement scales depending on the context (e.g., weather forecasting, cooking, or industrial processes).

#### **Common Temperature Scales:**

1. **Celsius (°C)**:
   * The Celsius scale is widely used around the world, especially in countries that do not use the Imperial system (i.e., most countries other than the U.S.).
   * It is based on the freezing point of water (0°C) and the boiling point of water (100°C) at standard atmospheric pressure.
   * The scale is linear, with a 1°C difference corresponding to the same temperature difference as 1°F (Fahrenheit).
2. **Fahrenheit (°F)**:
   * The Fahrenheit scale is commonly used in the United States and some Caribbean nations.
   * The freezing point of water is 32°F, and the boiling point is 212°F (at standard atmospheric pressure).
   * ****Unlike Celsius, which has 100 intervals between freezing and boiling points, Fahrenheit has 180 intervals between freezing and boiling points.
3. **Present the code of your application (forms).**

**Dashboard**

… your code goes here

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;

// Name Meet Hingu

// date 06/12/2024

// desc : main form.

namespace meet\_project

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

}

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

{

lotto649 obj = new lotto649();

obj.Show();

}

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

{

lottoMax obj = new lottoMax();

obj.Show();

}

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

{

Application.Exit();

}

private void button4\_Click(object sender, EventArgs e)

{

ipv4 obj = new ipv4();

obj.Show();

}

private void button5\_Click(object sender, EventArgs e)

{

moneyConvertor obj = new moneyConvertor();

obj.Show();

}

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

{

calculatorForm obj = new calculatorForm();

obj.Show();

}

private void button7\_Click(object sender, EventArgs e)

{

temperature obj = new temperature();

obj.Show();

}

}

}

**LotoMAX**

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;

//name Meet Hingu

//date 06/12/2024

//desc

namespace meet\_project

{

public partial class lottoMax : Form

{

public lottoMax()

{

InitializeComponent();

}

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

{

Random random = new Random();

string randomNumbers = "";

string lotteryName = "Max"; // Example lottery name

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

{

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

randomNumbers += randomNumber.ToString() + Environment.NewLine;

}

int extraNumber = new Random().Next(1, 50); // Generate an extra number

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

// Format the result

string result = $"{lotteryName}, {timestamp}, {string.Join(",", randomNumbers)} Extra {extraNumber}";

// Display in the TextBox

textBox1.Text += randomNumbers + Environment.NewLine;

// Save to file

SaveToFile("LottoNbrs.txt", result);

}

private void SaveToFile(string filePath, string content)

{

using (FileStream fs = new FileStream(filePath, FileMode.Append, FileAccess.Write))

using (StreamWriter writer = new StreamWriter(fs))

{

writer.WriteLine(content);

}

}

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

{

// Display a message box with Yes and No options

var dialogResult = MessageBox.Show(

"Are you sure you want to exit?",

"Exit Confirmation",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question

);

// Check the user's response

if (dialogResult == DialogResult.No)

{

// If the user clicks "No", cancel the form closing

e.Cancel = true;

}

else if (dialogResult == DialogResult.Yes)

{

// If the user clicks "Yes", proceed with the form closing

e.Cancel = false;

}

}

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

{

this.Close();

}

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

{

}

private string ReadFromFile(string filePath)

{

using (FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read))

using (StreamReader reader = new StreamReader(fs))

{

return reader.ReadToEnd();

}

}

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

{

string fileContent;

using (FileStream fs = new FileStream("LottoNbrs.txt", FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(fs))

{

fileContent = reader.ReadToEnd();

}

// If file is empty, show a message

if (string.IsNullOrWhiteSpace(fileContent))

{

MessageBox.Show("No data available in the file.", "File Content", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// Split content into lines

string[] lines = fileContent.Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);

// Prepare the output message

string message = "Lotto Numbers:\n";

foreach (var line in lines)

{

message += line + "\n";

}

// Display the result in a message box

MessageBox.Show(message, "File Content", MessageBoxButtons.OK, MessageBoxIcon.Information);

textBox1.Text = "";

}

}

}

… your code goes here

Loto649

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;

// Name Meet Hingu

// date 06/12/2024

// desc : lotto649 form.

namespace meet\_project

{

public partial class lotto649 : Form

{

public lotto649()

{

InitializeComponent();

}

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

{

Random random = new Random();

string randomNumbers = "";

string lotteryName = "Lotto649"; // Example lottery name

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

{

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

randomNumbers += randomNumber.ToString() + Environment.NewLine;

}

int extraNumber = new Random().Next(1, 50); // Generate an extra number

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

// Format the result

string result = $"{lotteryName}, {timestamp}, {string.Join(",", randomNumbers)} Extra {extraNumber}";

// Display in the TextBox

textBox1.Text += randomNumbers + Environment.NewLine;

// Save to file

SaveToFile("Lotto649Nbrs.txt", result);

}

private string ReadFromFile(string filePath)

{

using (FileStream fs = new FileStream(filePath, FileMode.Open, FileAccess.Read))

using (StreamReader reader = new StreamReader(fs))

{

return reader.ReadToEnd();

}

}

private void SaveToFile(string filePath, string content)

{

using (FileStream fs = new FileStream(filePath, FileMode.Append, FileAccess.Write))

using (StreamWriter writer = new StreamWriter(fs))

{

writer.WriteLine(content);

}

}

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

{

string fileContent;

using (FileStream fs = new FileStream("Lotto649Nbrs.txt", FileMode.OpenOrCreate, FileAccess.Read))

using (StreamReader reader = new StreamReader(fs))

{

fileContent = reader.ReadToEnd();

}

// If file is empty, show a message

if (string.IsNullOrWhiteSpace(fileContent))

{

MessageBox.Show("No data available in the file.", "File Content", MessageBoxButtons.OK, MessageBoxIcon.Information);

return;

}

// Split content into lines

string[] lines = fileContent.Split(new string[] { Environment.NewLine }, StringSplitOptions.RemoveEmptyEntries);

// Prepare the output message

string message = "Lotto Numbers:\n";

foreach (var line in lines)

{

message += line + "\n";

}

// Display the result in a message box

MessageBox.Show(message, "File Content", MessageBoxButtons.OK, MessageBoxIcon.Information);

textBox1.Text = "";

}

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

{

// Display a message box with Yes and No options

var dialogResult = MessageBox.Show(

"Are you sure you want to exit?",

"Exit Confirmation",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question

);

// Check the user's response

if (dialogResult == DialogResult.No)

{

// If the user clicks "No", cancel the form closing

e.Cancel = true;

}

else if (dialogResult == DialogResult.Yes)

{

// If the user clicks "Yes", proceed with the form closing

e.Cancel = false;

}

}

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

{

this.Close();

}

}

}

Money Conversion

… your code goes here

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.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

using static System.Resources.ResXFileRef;

// Name Meet Hingu

// date 06/12/2024

// desc : moneyConvertor form.

namespace meet\_project

{

public partial class moneyConvertor : Form

{

public moneyConvertor()

{

InitializeComponent();

}

public class MoneyConv

{

// Conversion factors relative to CAD

private readonly double[] exchangeRates = { 1.00, 0.7353, 0.6757, 0.5952, 61.23, 0.000023 }; // INR and Bitcoin rates

private readonly string[] currencyCodes = { "CAD", "USD", "EUR", "GBP", "INR", "BTC" };

public bool IsValidAmount(string amount)

{

// Validate input format (0.00 to 999.99)

string pattern = @"^\d{1,3}(\.\d{1,2})?$";

return Regex.IsMatch(amount, pattern);

}

public double[] ConvertAmount(double baseAmount)

{

double[] results = new double[exchangeRates.Length];

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

{

results[i] = Math.Round(baseAmount \* exchangeRates[i], 2);

}

return results;

}

public void SaveConversions(string baseCurrency, double baseAmount, double[] conversions)

{

string filePath = "MoneyConversions.txt";

using (StreamWriter writer = new StreamWriter(filePath, true))

{

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

string log = $"{dateTime}, {baseAmount} {baseCurrency} = ";

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

{

log += $"{conversions[i]} {currencyCodes[i]}";

if (i < conversions.Length - 1) log += "; ";

}

writer.WriteLine(log);

}

}

public string ReadConversions()

{

string filePath = "MoneyConversions.txt";

if (!File.Exists(filePath))

return "No conversion records found.";

using (StreamReader reader = new StreamReader(filePath))

{

return reader.ReadToEnd();

}

}

}

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

{

MoneyConv converter = new MoneyConv();

string amountText = textBox1.Text.Trim();

if (!converter.IsValidAmount(amountText))

{

MessageBox.Show("Invalid amount! Enter a value between 0.00 and 999.99.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

double baseAmount = double.Parse(amountText);

double[] conversions = converter.ConvertAmount(baseAmount);

// Display results in respective TextBoxes

textboxCAD.Text = conversions[0].ToString("F2");

textboxUSA.Text = conversions[1].ToString("F2");

textboxEUR.Text = conversions[2].ToString("F2");

textboxGBP.Text = conversions[3].ToString("F2");

textboxINR.Text = conversions[4].ToString("F2"); // INR

textboxBTC.Text = conversions[5].ToString("F6");

}

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

{

string amountInput = textBox1.Text.Trim();

// Check if the input is not empty

if (string.IsNullOrEmpty(amountInput))

{

MessageBox.Show("Please enter a value before clicking the button.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

MoneyConv converter = new MoneyConv();

string amountText = textBox1.Text.Trim();

double baseAmount = double.Parse(amountText);

double[] conversions = converter.ConvertAmount(baseAmount);

// Display results in respective TextBoxes

textboxCAD.Text = conversions[0].ToString("F2");

textboxUSA.Text = conversions[1].ToString("F2");

textboxEUR.Text = conversions[2].ToString("F2");

textboxGBP.Text = conversions[3].ToString("F2");

textboxINR.Text = conversions[4].ToString("F2"); // INR

textboxBTC.Text = conversions[5].ToString("F6");

string message = $"Conversion results:\n" +

$"2024/12/06 ({DateTime.Now:hh:mm:ss tt}), {baseAmount} CAD = " +

$"{conversions[0]} CAD; {conversions[1]} USD; {conversions[2]} EUR; " +

$"{conversions[3]} GBP; {conversions[4]} INR; {conversions[5]} BTC";

MessageBox.Show(message, "Conversion Complete", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

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

{

if (button3.clicked)

{

}

}

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

{

// Display a message box with Yes and No options

var dialogResult = MessageBox.Show(

"Are you sure you want to exit?",

"Exit Confirmation",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question

);

// Check the user's response

if (dialogResult == DialogResult.No)

{

// If the user clicks "No", cancel the form closing

e.Cancel = true;

}

else if (dialogResult == DialogResult.Yes)

{

// If the user clicks "Yes", proceed with the form closing

e.Cancel = false;

}

}

}

}

IPv4 & IP

… your code goes here

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.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

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

// Name Meet Hingu

// date 06/12/2024

// desc : ipv4 and ipv6 form.

namespace meet\_project

{

public partial class ipv4 : Form

{

public ipv4()

{

InitializeComponent();

}

public class IPv4IPv6Conv

{

// Validate IPv4

public bool IsValidIPv4(string ip)

{

string pattern = @"^(\d{1,3}\.){3}\d{1,3}$";

if (Regex.IsMatch(ip, pattern))

{

string[] parts = ip.Split('.');

foreach (string part in parts)

{

if (int.Parse(part) > 255) return false;

}

return true;

}

return false;

}

// Validate IPv6

public bool IsValidIPv6(string ip)

{

string pattern = @"^([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}$";

return Regex.IsMatch(ip, pattern);

}

public void SaveIP(string type, string ip)

{

string filePath = "IPRecords.txt";

using (FileStream fs = new FileStream(filePath, FileMode.Append))

using (BinaryWriter writer = new BinaryWriter(fs))

{

writer.Write($"{type}: {ip}");

writer.Write(DateTime.Now.ToString());

}

}

}

private void ipv4\_Load(object sender, EventArgs e)

{

label3.Text = "Today :" + DateTime.Now.ToLongDateString();

}

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

{

// Display a message box with Yes and No options

var dialogResult = MessageBox.Show(

"Are you sure you want to exit?",

"Exit Confirmation",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question

);

// Check the user's response

if (dialogResult == DialogResult.No)

{

// If the user clicks "No", cancel the form closing

e.Cancel = true;

}

else if (dialogResult == DialogResult.Yes)

{

// If the user clicks "Yes", proceed with the form closing

e.Cancel = false;

}

}

private void button4\_Click(object sender, EventArgs e)

{

this.Close();

}

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

{

textBox1.Clear();

textBox2.Clear();

MessageBox.Show("Data has been Reset.");

textBox1.Focus();

}

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

{

string ipv4 = textBox1.Text.Trim(); // IPv4 input

string ipv6 = textBox2.Text.Trim(); // IPv6 input

try

{

IPv4IPv6Conv converter = new IPv4IPv6Conv();

if (converter.IsValidIPv4(ipv4))

{

converter.SaveIP("IPv4", ipv4);

MessageBox.Show($"{ipv4} \n This ip is correct !");

}

else if (converter.IsValidIPv6(ipv6))

{

converter.SaveIP("IPv6", ipv6);

MessageBox.Show($"{ipv6} \n This IP is Correct !");

}

else if(!converter.IsValidIPv4(ipv4))

{

MessageBox.Show($"{ipv4} \n this ip must have 4 bytes integer number between 0 to 255 separated by a dot(255.255.255.255)");

}

else if (!converter.IsValidIPv4(ipv6))

{

MessageBox.Show($"{ipv6} \n this ip must have 16 bytes integer number and text between 0 to 255 and a to f consecutevaly separated by a colan");

}

}

catch (Exception ex)

{

MessageBox.Show($"Error: {ex.Message}");

}

}

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

{

string ipv4 = textBox1.Text.Trim();

string ipv6 = textBox2.Text.Trim();

// Check if both fields are empty

if (string.IsNullOrEmpty(ipv4) && string.IsNullOrEmpty(ipv6))

{

MessageBox.Show("Both fields are empty. Please enter an IP address.", "Input Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

else

{

// Display the entered IP addresses in a message box

string message = "Entered IP Addresses:\n";

if (!string.IsNullOrEmpty(ipv4))

{

message += $"IPv4: {ipv4}\n";

}

if (!string.IsNullOrEmpty(ipv6))

{

message += $"IPv6: {ipv6}\n";

}

MessageBox.Show(message, "Entered IP Addresses", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

}

Calculator

… your code goes here

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;

// Name Meet Hingu

// date 06/12/2024

//desc : the form for calculator

namespace meet\_project

{

public partial class calculatorForm : Form

{

private Calculator calculator = new Calculator();

public calculatorForm()

{

InitializeComponent();

}

public class Calculator

{

public decimal operand1;

public decimal operand2;

public string operation;

public decimal CurrentValue { get; private set; }

public void SetOperand1(decimal value)

{

operand1 = value;

}

public void SetOperand2(decimal value)

{

operand2 = value;

}

public void SetOperation(string op)

{

operation = op;

}

public void Calculate()

{

switch (operation)

{

case "+":

CurrentValue = operand1 + operand2;

break;

case "-":

CurrentValue = operand1 - operand2;

break;

case "\*":

CurrentValue = operand1 \* operand2;

break;

case "/":

CurrentValue = operand2 != 0 ? operand1 / operand2 : 0;

break;

}

}

}

private void button17\_Click(object sender, EventArgs e)

{

textBox2.Text = " ";

textBox1.Text = "";

}

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

{

}

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

{

// Display a message box with Yes and No options

var dialogResult = MessageBox.Show(

"Are you sure you want to exit?",

"Exit Confirmation",

MessageBoxButtons.YesNo,

MessageBoxIcon.Question

);

// Check the user's response

if (dialogResult == DialogResult.No)

{

// If the user clicks "No", cancel the form closing

e.Cancel = true;

}

else if (dialogResult == DialogResult.Yes)

{

// If the user clicks "Yes", proceed with the form closing

e.Cancel = false;

}

}

private void button18\_Click(object sender, EventArgs e)

{

this.Close();

}

// = button

private void button16\_Click(object sender, EventArgs e)

{

try

{

if (decimal.TryParse(textBox1.Text, out decimal operand2))

{

calculator.SetOperand2(operand2);

calculator.Calculate();

// Create the equation string

string equation = $"{calculator.operand1} {calculator.operation} {calculator.operand2}";

string equation2 = $"{calculator.operand1} {calculator.operation} {calculator.operand2} = {calculator.CurrentValue}";

// Display results

textBox1.Text = equation; // Full equation

textBox2.Text = equation2; // Result only

}

}

catch (InvalidOperationException ex)

{

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

textBox2.Clear();

}

}

// button 0 to 9

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

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

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

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button10\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button9\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button8\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button7\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button11\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button5\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

private void button4\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

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

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

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

{

Button btn = sender as Button;

textBox1.Text += btn.Text;

}

// operator buttons

private void button12\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

if (btn != null && decimal.TryParse(textBox1.Text, out decimal value))

{

calculator.SetOperand1(value);

calculator.SetOperation(btn.Text);

textBox1.Clear();

}

}

private void button14\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

if (btn != null && decimal.TryParse(textBox1.Text, out decimal value))

{

calculator.SetOperand1(value);

calculator.SetOperation(btn.Text);

textBox1.Clear();

}

}

private void button13\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

if (btn != null && decimal.TryParse(textBox1.Text, out decimal value))

{

calculator.SetOperand1(value);

calculator.SetOperation(btn.Text);

textBox1.Clear();

}

}

private void button15\_Click(object sender, EventArgs e)

{

Button btn = sender as Button;

if (btn != null && decimal.TryParse(textBox1.Text, out decimal value))

{

calculator.SetOperand1(value);

calculator.SetOperation(btn.Text);

textBox1.Clear();

}

}

private void textBox2\_TextChanged(object sender, EventArgs e)

{

}

}

}

**Temperature**

**Heres the code**

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;

// Name Meet Hingu

// date 06/12/2024

// desc : temperature form.

namespace meet\_project

{

public partial class temperature : Form

{

public temperature()

{

InitializeComponent();

}

double convertedTemp = 0;

string conversionType = "";

string message = "";

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

{

if (!double.TryParse(textBox1.Text, out double inputTemp))

{

MessageBox.Show("Please enter a valid numeric temperature.", "Invalid Input", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

if (radioButton1.Checked)

{

convertedTemp = (inputTemp \* 9 / 5) + 32;

conversionType = "Celsius to Fahrenheit";

message = GetMessageForCelsius(inputTemp);

label1.Text = "(°C)"; // Label for Celsius input

label2.Text = "(°F)"; // Label for Fahrenheit output

}

else if (radioButton2.Checked)

{

convertedTemp = (inputTemp - 32) \* 5 / 9;

conversionType = "Fahrenheit to Celsius";

message = GetMessageForFahrenheit(convertedTemp);

label1.Text = "F"; // Label for Celsius input

label2.Text = "C"; // Label for Fahrenheit output

}

else

{

MessageBox.Show("Please select a conversion type.", "Error", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Display results

textBox2.Text = convertedTemp.ToString("F2");

textBox3.Text = message;

// Save the conversion to a text file

// SaveConversionToFile(inputTemp, convertedTemp, conversionType);

}

private string GetMessageForCelsius(double tempCelsius)

{

if (tempCelsius >= 100)

return "Water Boils";

else if (tempCelsius >= 40)

return "Hot Bath";

else if (tempCelsius >= 30)

return "Beach Weather";

else if (tempCelsius >= 21)

return "Room Temperature";

else if (tempCelsius >= 10)

return "Cool Day";

else if (tempCelsius >= 0)

return "Freezing Point";

else

return "Extremely Cold";

}

private string GetMessageForFahrenheit(double tempFahrenheit)

{

if (tempFahrenheit >= 212)

return "Water Boils";

else if (tempFahrenheit >= 104)

return "Hot Bath";

else if (tempFahrenheit >= 86)

return "Beach Weather";

else if (tempFahrenheit >= 69.8)

return "Room Temperature";

else if (tempFahrenheit >= 50)

return "Cool Day";

else if (tempFahrenheit >= 32)

return "Freezing Point";

else

return "Extremely Cold";

}

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

{

string filePath = "./TempConversions.txt";

string logEntry = $"{DateTime.Now}: {conversionType} - Input: {textBox1.Text}, Output: {textBox2.Text}";

try

{

string directoryPath = Path.GetDirectoryName(filePath);

if (!Directory.Exists(directoryPath))

{

Directory.CreateDirectory(directoryPath);

}

// Append the log entry to the file

File.AppendAllText(filePath, logEntry + Environment.NewLine);

MessageBox.Show("Conversion saved successfully.", "Saved", MessageBoxButtons.OK, MessageBoxIcon.Information);

// Display the result in a message box

string resultMessage = $"Conversion completed!\n{conversionType}:\nInput: {textBox1.Text}°\nOutput: {textBox2.Text}°\nMessage: {textBox3.Text}";

MessageBox.Show(resultMessage, "Conversion Result", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Error saving conversion: {ex.Message}", "Error", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

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

{

this.Close();

}

}

}

1. **Present the classes and/or methods that you create or you did use in the project and the class diagram(s) and the algorithm of the method(s) .**

|  |  |
| --- | --- |
| **Class/Method Name** | **Description** |
|  |  |
| 1. Class Calc | Enter a short description of the class or of the method ………………. |
| 1. Method for Calc class |  |
| 1. Method SaveIP() for IP class | This method saves the valid IP address along with the type (IPv4 or IPv6) and the timestamp into a text file. |
| 1. Class IP | Enter a short description of the class or of the method ……………….. |
| 1. void Method(int total) | Enter a short description of the class or of the method ……….. |
|  |  |
| 1. Constructor() | Enter a short description of the class or of the method ………… |
| 1. Class Calc | The Calc class handles basic arithmetic calculations such as addition, subtraction, multiplication, and division. It contains methods that take user input, process the calculation, and return the result. This class is crucial for providing the core functionality of the calculator in the project. |
| 1. Method Calculate() for Calc class | The Calculate method performs the actual computation based on the selected operation (e.g., addition, subtraction). It takes two numbers as input, performs the desired operation, and returns the result. It also handles edge cases such as division by zero, providing appropriate error messages. |
| 1. Class IP | The IP class is responsible for validating and processing IPv4 and IPv6 addresses. It checks whether the provided IP address matches the correct format for IPv4 or IPv6 using regular expressions. The class also includes methods for saving valid IPs into a file and displaying them. |
| 1. Method IsValidIPv4() for IP class | This method validates whether the entered IP address is a valid IPv4 address. It checks the address format (e.g., X.X.X.X) and verifies that each part is a number between 0 and 255. |
| 1. Class MoneyConv | The MoneyConv class is responsible for converting between different currencies. It contains conversion methods for several currencies (CAD, USD, EUR, GBP, INR, BTC) based on exchange rates relative to CAD. The class also handles saving the conversion results to a text file. |
| 1. Method ConvertAmount() for MoneyConv class | This method performs currency conversion based on a given amount in CAD. It multiplies the input amount by predefined exchange rates for each currency and returns the conversion results |
| 1. Class Temperature | The Temperature class handles the conversion between Celsius and Fahrenheit temperatures. It also provides specific messages based on the temperature ranges, like "Room Temperature" or "Water Boils." The class supports both conversions from Celsius to Fahrenheit and vice versa. |

**Class diagram:**

+-----------------------------------+

| IPv4IPv6Conv |

+-----------------------------------+

| - exchangeRates[] |

| - currencyCodes[] |

+-----------------------------------+

| + IsValidIPv4(ip: string) : bool |

| + IsValidIPv6(ip: string) : bool |

| + SaveIP(type: string, ip: string)|

+-----------------------------------+

| uses

v

+-------------------+ +-----------------------+

| Form (ipv4) |--- | MessageBoxHelper |

+-------------------+ +-----------------------+

| + buttonClick() | | + ShowMessage() |

| + ValidateIP() | +-----------------------+

+-------------------+

+----------------------------------+

| MoneyConv |

+----------------------------------+

| - exchangeRates[] |

| - currencyCodes[] |

+----------------------------------+

| + IsValidAmount(amount: string) |

| + ConvertAmount(baseAmount: double) |

| + SaveConversions() |

+----------------------------------+

|

v

+---------------------+

| Form (moneyConv) |

+---------------------+

| + buttonClick() |

| + ValidateInput() |

+---------------------+

+----------------------------+

| Temperature |

+----------------------------+

| + GetMessageForCelsius() |

| + GetMessageForFahrenheit()|

| + SaveConversionToFile() |

+----------------------------+

|

v

+-------------------+ +----------------------+

| Form (temp) |---| MessageBoxHelper |

+-------------------+ +----------------------+

| + buttonClick() | | + ShowMessage() |

| + ValidateInput() |

+-------------------+

+------------------------+

| LottoGenerator |

+------------------------+

| + GenerateNumbers() |

| + SaveLottoResults() |

+------------------------+

|

v

+----------------------+

| Form (lotto) |

+----------------------+

| + buttonClick() |

| + ValidateInput() |

+----------------------+

+-------------------------+

| HelperMethods |

+-------------------------+

| + ValidateAmount() |

| + ShowMessage() |

+-------------------------+

+-------------------------+

| FileHandler |

+-------------------------+

| + SaveToFile() |

| + ReadFromFile() |

+-------------------------+

**Method(s) algorithm:**

Below is an algorithmic breakdown of the key methods used in the various parts of your project (IPV4/IPv6 Converter, Money Conversion, Temperature Conversion, Calculator, Lotto, etc.). These algorithms describe the essential steps for each method to guide the implementation and logic of your application.

### ****1. IPv4/IPv6 Conversion (IPv4IPv6Conv Class)****

#### **IsValidIPv4()**

1. Accept an IP string ip as input.
2. Check if the IP matches the regex pattern ^(\d{1,3}\.){3}\d{1,3}$.
3. Split the IP by the . character.
4. For each part, check if it is an integer and ensure it is between 0 and 255.
5. If all conditions are met, return true. Otherwise, return false.

#### **IsValidIPv6()**

1. Accept an IP string ip as input.
2. Check if the IP matches the regex pattern ^([0-9a-fA-F]{1,4}:){7}[0-9a-fA-F]{1,4}$.
3. If the IP matches the regex pattern, return true. Otherwise, return false.

#### **SaveIP()**

1. Accept parameters: type (either "IPv4" or "IPv6") and ip (the IP address).
2. Open a file stream in append mode for IPRecords.txt.
3. Write the type, ip, and the current date and time to the file.
4. Close the file stream after writing.

### ****2. Money Conversion (MoneyConv Class)****

#### **IsValidAmount()**

1. Accept an amount string amount as input.
2. Validate if it matches the regex pattern ^\d{1,3}(\.\d{1,2})?$, ensuring it is a number with up to two decimal places.
3. If it matches, return true. Otherwise, return false.

#### **ConvertAmount()**

1. Accept a baseAmount in CAD as input.
2. Multiply baseAmount by each value in the exchangeRates[] array (this array contains conversion rates for USD, EUR, GBP, INR, BTC).
3. Store the result of each conversion in an array results[].
4. Return the results[] array containing the converted amounts.

#### **SaveConversions()**

1. Accept the baseCurrency, baseAmount, and conversions[] array.
2. Open a stream writer for MoneyConversions.txt.
3. Write the current date and time, the base amount, and the converted values in the specified format to the file.
4. Close the stream writer after writing.

### ****3. Temperature Conversion (Temperature Class)****

#### **GetMessageForCelsius()**

1. Accept a tempCelsius value.
2. Check the following ranges for the temperature:
   * If tempCelsius >= 100, return "Water Boils".
   * If tempCelsius >= 40, return "Hot Bath".
   * If tempCelsius >= 30, return "Beach Weather".
   * If tempCelsius >= 21, return "Room Temperature".
   * If tempCelsius >= 10, return "Cool Day".
   * If tempCelsius >= 0, return "Freezing Point".
   * Otherwise, return "Extremely Cold".

#### **GetMessageForFahrenheit()**

1. Accept a tempFahrenheit value.
2. Check the following ranges for the temperature:
   * If tempFahrenheit >= 212, return "Water Boils".
   * If tempFahrenheit >= 104, return "Hot Bath".
   * If tempFahrenheit >= 86, return "Beach Weather".
   * If tempFahrenheit >= 69.8, return "Room Temperature".
   * If tempFahrenheit >= 50, return "Cool Day".
   * If tempFahrenheit >= 32, return "Freezing Point".
   * Otherwise, return "Extremely Cold".

#### **SaveConversionToFile()**

1. Accept inputTemp, convertedTemp, and conversionType as input.
2. Open a stream writer for TempConversions.txt.
3. Write the current date and time, conversion type, input temperature, and output temperature in the required format.
4. Close the stream writer after writing.

### ****4. Calculator (CalculatorLogic Class)****

#### **Add()**

1. Accept two input numbers a and b.
2. Return the sum of a + b.

#### **Subtract()**

1. Accept two input numbers a and b.
2. Return the result of a - b.

#### **Multiply()**

1. Accept two input numbers a and b.
2. Return the result of a \* b.

#### **Divide()**

1. Accept two input numbers a and b.
2. If b == 0, display an error (division by zero).
3. Otherwise, return the result of a / b.

#### **Clear()**

1. Clear the display/input fields of the calculator.

### ****5. Lotto Max / Lotto 649 (LottoGenerator Class)****

#### **GenerateNumbers()**

1. Define the number of picks (e.g., 6 numbers for Lotto 649).
2. Randomly generate the required number of unique numbers within a defined range (e.g., 1-49).
3. Store the generated numbers in an array or list.
4. Return the generated list/array of numbers.

#### **SaveLottoResults()**

1. Accept the generated lotto numbers and the current date and time.
2. Open a stream writer for LottoResults.txt.
3. Write the current date and time, followed by the generated numbers.
4. Close the stream writer after writing.

### ****6. Helper Methods (HelperMethods Class)****

#### **ValidateAmount()**

1. Accept an amount string as input.
2. Check if the amount is a valid numeric value.
3. If valid, return true. Otherwise, return false.

#### **ShowMessage()**

1. Accept a message string as input.
2. Display the message using MessageBox.Show().
3. Return the result (if necessary).

#### **SaveToFile()**

1. Accept data (e.g., conversion results or logs) and a filePath.
2. Open a file stream in append mode for the specified file.
3. Write the data to the file.
4. Close the file stream after writing.

#### **ReadFromFile()**

1. Accept a filePath as input.
2. Open a stream reader for the specified file.
3. Read and return the contents of the file.
4. Close the stream reader after reading.
5. **Present the difficulties that you have, what was the hardest and the easiest part of your project application development.**

**Hardest Part of the Project:**

**Problem:** Validating and Converting Data  
For the money conversion and temperature conversion features, I struggled with ensuring that the data input by the user was valid and properly converted. Handling edge cases such as invalid inputs (e.g., letters or symbols instead of numbers) took some time.

**Solution:**  
I implemented validation using regular expressions to check for numeric inputs in specific formats, ensuring that only valid data was accepted. I also used try-catch blocks for better error handling to notify the user if something went wrong.

**Reflection:**  
Data validation and conversion were tricky, as they involved multiple steps to ensure proper formatting and prevent crashes. However, this made me better at using regex for validation and handling exceptions in C#.

**Problem:** Saving Data to a Text File  
Saving the conversion data (like the results of money or temperature conversions) to a text file was a bit confusing initially, especially with formatting the output and ensuring the file was written correctly.

**Solution:**  
I used the StreamWriter class to handle file output and ensured the data was appended to the file instead of overwriting it. Additionally, I formatted the text output with the current timestamp to keep track of each conversion.

**Reflection:**  
This task was challenging because I had to ensure that file handling was done properly to avoid data loss and that the user experience was smooth. However, it also taught me about efficient file I/O operations in C#.

### ****Easiest Part of the Project:****

The easiest part of the project was **creating the basic structure of the application** (such as setting up forms, buttons, text boxes, etc.). These were straightforward tasks and involved primarily layout and event handling, which Visual Studio simplifies with its drag-and-drop interface.