This code demonstrates a simple Windows Forms application in C#. The application receives a name from a text box, validates it, formats it, and then writes it to a file named "Friends.txt.” If an exception occurs during this process, it shows an error message.

**Let's walk through the code step by step:**

**A screenshot of a computer

Description automatically generated**

**A screenshot of a computer code

Description automatically generated with low confidence**

**Namespace and Using Statements**

**A picture containing text, font, line, number

Description automatically generated**

using System;

using System.IO;

using System.Linq;

using System.Windows.Forms;

These lines import namespaces that contain classes and methods which our program needs. System is the root namespace for fundamental types. System.IO has types that allow reading and writing to files and data streams. System.Linq provides LINQ (Language Integrated Query) functionalities. System.Windows.Forms contain classes for creating Windows-based applications that take input from a mouse, keyboard, pen, or touch.

**Namespace Definition**

****

namespace Friend\_File

This line declares a namespace named Friend\_File. A namespace organizes your code and is a collection of classes, interfaces, structs, enums, and delegates.

**Form1 Class and Constructor**

**A picture containing text, font, line, plot

Description automatically generated**

public partial class Form1: Form

{

public Form1()

{

InitializeComponent();

}

Here Form1 is a class which is a partial class of type Form. This indicates that class Form1 is a Windows Form. Form1 is the default name given to the first form in a Windows Forms Application. The InitializeComponent() method in the constructor is auto-generated by the Windows Forms designer, and it initializes the form and its components.

**writeNameButton\_Click Event Handler**

****

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

This event handler gets called when the 'writeNameButton' button is clicked. The sender parameter is the control that triggered the event—in this case, the writeNameButton. EventArgs e is the data for the event.

**Name Processing and Writing**

**A screenshot of a computer program

Description automatically generated with low confidence**

The first section of the click event handler fetches the name from the text box, checks if it's valid using the IsNameFormatValid method, and if it is, it writes it to the file "Friends.txt" using a StreamWriter object.

string name = nameTextBox.Text.Trim();

if (IsNameFormatValid(name))

{

using (StreamWriter outputFile = File.AppendText("Friends.txt"))

{

string formattedName = FormatName(name);

outputFile.WriteLine(formattedName);

MessageBox.Show("The name was written.");

}

}

else

{

MessageBox.Show("Invalid name format. Please enter a name with only letters and follow the specified format.");

}

**Exception Handling**

The entire name processing and writing section is wrapped in a try-catch block, which handles any exceptions that might occur during the process.

A picture containing text, font, line, screenshot

Description automatically generated

catch (Exception ex)

{

MessageBox.Show("An error occurred: " + ex.Message);

}

IsNameFormatValid Method

This method checks if the name is valid. A valid name starts with an uppercase letter and follows with lowercase letters. The name can have multiple parts separated by spaces, and each part should follow this rule.

A screen shot of a computer code

Description automatically generated with medium confidence

private bool IsNameFormatValid(string name)

{

string[] nameParts = name.Split(' ');

foreach (string part in nameParts)

{

if (part.Length < 1 || !char.IsUpper(part[0]) || !part.Substring(1).All(char.IsLower))

{

return false;

}

}

return true;

}

This method splits the name into parts by spaces. It then checks each piece to see if it starts with an uppercase letter and if the other characters are lowercase. Any function that does not meet these conditions returns false to indicate an invalid name. If all parts are valid, it returns true.

**FormatName Method**

**A picture containing text, screenshot, line, font

Description automatically generated**

private string FormatName(string name)

{

string[] nameParts = name.Split(' ');

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

{

nameParts[i] = char.ToUpper(nameParts[i][0]) + nameParts[i].Substring(1).ToLower();

}

return string.Join(" ", nameParts);

}

This method also splits the name into parts by spaces. It then capitalizes the first letter of each piece and converts the rest of the region to lowercase. It finally joins all the details with spaces in between and returns the formatted name.

**exitButton\_Click Event Handler**

**A picture containing text, line, screenshot, plot

Description automatically generated**

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

{

this.Close();

}

This is the event handler for the 'exitButton' button click event. It closes the form when the button is clicked. The Close() method is a built-in method of the Form class used to close the current form.

This walk-through should provide a comprehensive understanding of the code. Remember, best practices in C# and .NET often involve cleanly handling exceptions, ensuring resources like file handles are correctly disposed of, and providing clear feedback to the user, all of which this code does.