Currency Converter Program in CSharp

Are you ready to start a journey into the world of currency conversion using C# programming? Welcome to the Currency Converter Program in C# Tutorial, where you'll make the process of creating a versatile and handy tool for currency exchange calculations easy to understand. In this tutorial, we'll guide you through the steps of building your very own peso-to-dollar and dollar-to-peso converter program. Whether you're a new programmer eager to learn or an experienced developer looking for a practical project, this tutorial has something valuable to offer. Plus, stay tuned for a bonus – you'll have the opportunity to download the complete source code for free, giving you a head start on your currency converter project. Let's get started!

Introduction

A currency converter app is a valuable tool that allows users to convert the value of one currency into another. Whether you're a traveler planning your expenses, an online shopper dealing with international prices, or simply curious about exchange rates, understanding how to create a currency converter program in C# can be a practical and educational endeavor.

In this lesson, we will get on a trip to explain the process of building a simple currency converter using C# in a console application. By the end of this tutorial, you will have the skills to create your own currency converter, providing a useful tool for various real-world scenarios. Let's dive into the world of C# programming and currency conversion!

Objectives

1. Learn: Understand the fundamental principles of currency conversion, including exchange rates and currency pairs. Gain knowledge of the C# programming language, its syntax, and essential concepts related to console applications.

2. Apply: Apply your learning to design and implement a currency converter program in C#. Use variables, user input, and decision-making structures to create a functional application capable of converting between pesos and dollars.

3. Create: Develop a complete currency converter application from scratch, taking into account user-friendly interfaces and efficient currency conversion logic. Create a program that can handle both peso-to-dollar and dollar-to-peso conversions seamlessly.

4. Upgrade: Enhance your currency converter application by implementing additional features or improvements. Consider adding support for more currencies, implementing error handling mechanisms, or designing a graphical user interface (GUI) for a more user-friendly experience.

These objectives will guide your learning journey, helping you progressively build and improve your currency converter program while gaining valuable skills in C# programming and application development.

Example Source code

```
1.  using System;
2.
3.  namespace CurrencyConverter
4.  {
5.      class Program
6.      {
7.          static void Main(string[] args)
8.          {
9.              Console.WriteLine("Currency Converter");
10.             Console.WriteLine("1. Peso to Dollar");
11.             Console.WriteLine("2. Dollar to Peso");
12.             Console.WriteLine("Enter your choice (1 or 2):");
13.
14.             int choice;
15.             bool validChoice = int.TryParse(Console.ReadLine(), out choice);
16.
17.             if (validChoice)
18.             {
19.                 Console.Write("Enter the amount: ");
20.                 double amount;
21.                 bool validAmount = double.TryParse(Console.ReadLine(), out amount);
22.
23.                 if (validAmount)
24.                 {
25.                     double convertedAmount = 0;
26.
27.                     if (choice == 1)
28.                     {
29.                         // Peso to Dollar conversion (1 Peso = 0.020 USD)
30.                         convertedAmount = amount * 0.020;
31.                         Console.WriteLine($"{amount} Pesos = {convertedAmount} USD");
32.                     }
33.                     else if (choice == 2)
34.                     {
35.                         // Dollar to Peso conversion (1 USD = 50 Pesos)
36.                         convertedAmount = amount * 50;
37.                         Console.WriteLine($"{amount} USD = {convertedAmount} Pesos");
38.                     }
39.                     else
40.                     {
41.                         Console.WriteLine("Invalid choice. Please enter either 1 or 2.");
42.                     }
43.                 }
44.                 else
45.                 {
46.                     Console.WriteLine("Invalid amount. Please enter a valid numeric value.");
47.                 }
48.             }
49.             else
50.             {
51.                 Console.WriteLine("Invalid choice. Please enter either 1 or 2.");
52.             }
53.
54.             Console.ReadKey();
```

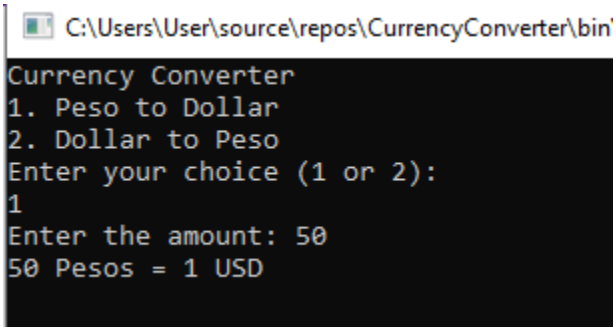
```
        }
    }
}
```

Explanation

1. ExplanWe start by including the System namespace, which provides essential classes and functionality for console applications.
2. We define a namespace called CurrencyConverter to encapsulate our program.
3. Within the CurrencyConverter namespace, we have a class named Program, which serves as the entry point for our program.
4. Inside the Main method, we display a welcome message to the user.
5. We then provide the user with two options: converting from Peso to Dollar (Option 1) or from Dollar to Peso (Option 2). The user's choice is stored in the choice variable.
6. Next, we prompt the user to enter the amount they want to convert, and we store this value in the amount variable.
7. We initialize convertedAmount to 0.0; this variable will hold the result of the currency conversion.
8. Depending on the user's choice, we perform the currency conversion logic. If the user selects Option 1, we divide the amount by 50 (assuming an exchange rate of 1 USD = 50 PHP). If the user selects Option 2, we multiply the amount by 50. The converted amount is stored in the convertedAmount variable.
9. We display the converted amount along with the original currency symbol (either $ or ₱) based on the user's choice.
10. If the user enters an invalid choice (neither 1 nor 2), we display an error message.

This code demonstrates a simple currency converter program in C# that allows users to convert between Peso and Dollar. Users can select the conversion type, enter an amount, and get the converted result. It also includes basic error handling for invalid choices.

Output



Summary

In this lesson, we explored the process of creating a Currency Converter program in C#. We unraveled the steps involved in building a practical application that allows users to easily convert between Peso and Dollar. The program's functionality was explained step by step, from user input to currency conversion logic. Additionally, we highlighted the importance of error handling to ensure a smooth user experience. By the end of this lesson, you should have a clear understanding of how to develop a basic currency converter and a foundation for more advanced C# projects. As you continue to practice and experiment with different features, you'll unlock the full potential of C# programming. Happy coding!

Assessment

To enhance the Currency Converter program in C#, you can consider the following assessment and assignments:

- **Additional Currency Support:** Expand the program to support multiple currencies beyond just Peso and Dollar. Allow users to choose from a list of currencies they want to convert between. You can use exchange rate APIs to fetch real-time conversion rates.
- **Currency Symbols:** Add currency symbols to the converted amounts to make the results more informative. For example, display "$100" instead of just "100" for dollars.
- **Decimal Precision:** Allow users to specify the number of decimal places for the converted amount, as different currencies have different precision requirements.
- **Error Handling:** Enhance error handling by providing clear error messages and solutions for common issues, such as invalid input.

Quiz

Question 1: What is the purpose of a constructor in C#?

A) To create an object of a class.
B) To destroy an object of a class.
C) To define the properties of a class.
D) To declare a variable.

Question 2: Which of the following data types is used to store whole numbers in C#?

A) float
B) double
C) int
D) char

Question 3: What does the "Console.WriteLine()" method do in C#?

A) Reads input from the console.
B) Writes output to the console.
C) Declares a new variable.
D) Performs mathematical calculations.

Question 4: Which keyword is used to define a constant variable in C#?

A) static
B) final
C) const
D) readonly

Question 5: What is the purpose of the "if" statement in C#?

A) To declare a loop.
B) To define a method.
C) To conditionally execute code.
D) To perform type casting.

```
using System;

namespace CurrencyConverter
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Currency Converter");
            Console.WriteLine("1. Peso to Dollar");
            Console.WriteLine("2. Dollar to Peso");
            Console.WriteLine("Enter your choice (1 or 2):");

            int choice;
            bool validChoice = int.TryParse(Console.ReadLine(), out choice);

            if (validChoice)
            {
                Console.Write("Enter the amount: ");
                double amount;
                bool validAmount = double.TryParse(Console.ReadLine(), out amount);

                if (validAmount)
                {
                    double convertedAmount = 0;

                    if (choice == 1)
                    {
                        // Peso to Dollar conversion (1 Peso = 0.020 USD)
                        convertedAmount = amount * 0.020;
                        Console.WriteLine($"{amount} Pesos = {convertedAmount} USD");
                    }
                    else if (choice == 2)
                    {
                        // Dollar to Peso conversion (1 USD = 50 Pesos)
                        convertedAmount = amount * 50;
                        Console.WriteLine($"{amount} USD = {convertedAmount} Pesos");
                    }
                    else
                    {
                        Console.WriteLine("Invalid choice. Please enter either 1 or 2.");
                    }
                }
                else
                {
                    Console.WriteLine("Invalid amount. Please enter a valid numeric value.");
                }
            }
            else
            {
                Console.WriteLine("Invalid choice. Please enter either 1 or 2.");
            }

            Console.ReadKey();
        }
    }
}
```