Odd, Even Number Checker in C-Sharp

Introduction

Welcome to this tutorial on building an Odd-Even Number Checker in C#! In the world of mathematics, numbers can be categorized into two main groups: odd and even. Understanding the concept of odd and even numbers is essential in various programming tasks and problem-solving scenarios.

An odd number is any integer that is not divisible evenly by 2, leaving a remainder of 1 when divided by 2. On the other hand, an even number is an integer that can be divided evenly by 2, resulting in no remainder.

In this tutorial, we will dive into the world of C# programming and explore how to create a simple yet powerful Odd-Even Number Checker. This program will allow us to input a number and determine whether it is odd or even. We will cover the step-by-step process of setting up a C# console application, receiving user input, performing the necessary calculations, and displaying the result.

By the end of this tutorial, you will have a solid understanding of how to build an Odd-Even Number Checker in C#, empowering you to solve similar problems and apply this knowledge to more advanced programming tasks. So, let's get started on this exciting journey of exploring odd and even numbers with C#!

Objectives

By the end of the lesson or tutorial, the students should be able to:

1. Identify and understand the concept of odd and even numbers.
2. Students will be able to explain the distinction between odd and even numbers and recognize their properties in mathematical terms.
3. Develop proficiency in using conditional statements and operators in C# to check for odd and even numbers.
4. Students will be able to write C# code that utilizes conditional statements and the modulo operator to determine whether a given number is odd or even.

Improvements and recommendations for the enhancement of the project:

- Implement user input handling and error validation to ensure the program handles various scenarios.
- Students will be able to enhance their program by incorporating error handling techniques, such as checking for non-numeric input and providing appropriate feedback to the user.
- Extend the functionality of the Odd-Even Number Checker program by adding additional features or enhancements.

- Students will be encouraged to experiment and explore the code to add new functionalities, such as checking multiple numbers, incorporating a loop, or creating a graphical user interface.

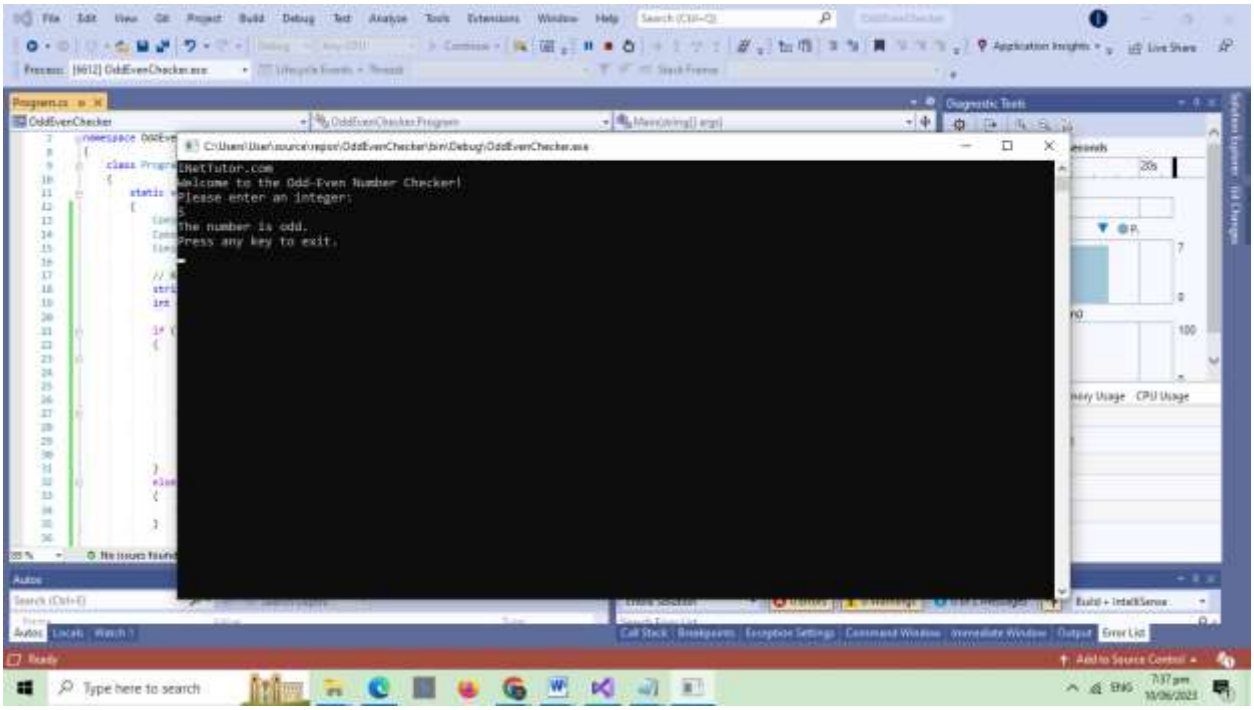
These objectives aim to provide students with a comprehensive understanding of odd and even numbers, proficiency in coding the Odd-Even Number Checker program in C#, and the ability to extend the program's functionality based on their creativity and problem-solving skills.

Code Example

```
1.  using System;
2.  using System.Collections.Generic;
3.  using System.Linq;
4.  using System.Text;
5.  using System.Threading.Tasks;
6.
7.  namespace OddEvenChecker
8.  {
9.      class Program
10.     {
11.         static void Main(string[] args)
12.         {
13.             Console.WriteLine("iNetTutor.com");
14.             Console.WriteLine("Welcome to the Odd-Even Number Checker!");
15.             Console.WriteLine("Please enter an integer:");
16.
17.             // Read user input as string and convert it to an integer
18.             string input = Console.ReadLine();
19.             int number;
20.
21.             if (int.TryParse(input, out number))
22.             {
23.                 if (number % 2 == 0)
24.                 {
25.                     Console.WriteLine("The number is even.");
26.                 }
27.                 else
28.                 {
29.                     Console.WriteLine("The number is odd.");
30.                 }
31.             }
32.             else
33.             {
34.                 Console.WriteLine("Invalid input. Please enter a valid integer.");
35.             }
36.
37.             Console.WriteLine("Press any key to exit.");
38.             Console.ReadKey();
39.         }
40.     }
41. }
```

```
7   namespace OddEvenChecker
8   {
9       class Program
10      {
11          static void Main(string[] args)
12          {
13              Console.WriteLine("iNetTutor.com");
14              Console.WriteLine("Welcome to the Odd-Even Number Checker!");
15              Console.WriteLine("Please enter an integer:");
16
17              // Read user input as string and convert it to an integer
18              string input = Console.ReadLine();
19              int number;
20
21              if (int.TryParse(input, out number))
22              {
23                  if (number % 2 == 0)
24                  {
25                      Console.WriteLine("The number is even.");
26                  }
27                  else
28                  {
29                      Console.WriteLine("The number is odd.");
30                  }
31              }
32              else
33              {
34                  Console.WriteLine("Invalid input. Please enter a valid integer.");
35              }
36
37              Console.WriteLine("Press any key to exit.");
38              Console.ReadKey();
39          }
40      }
41  }
```

Output



Code Explanation

Line 13-15

This code snippet is responsible for displaying messages on the console for the Odd-Even Number Checker program.

Line 13, **Console.WriteLine("iNetTutor.com");**, prints the string "iNetTutor.com" on the console. This line serves as a header or branding for the program.

Line 14, **Console.WriteLine("Welcome to the Odd-Even Number Checker!");**, displays the message "Welcome to the Odd-Even Number Checker!" on the console. This line serves as a welcoming message for the user, indicating the purpose of the program.

Line 15, **Console.WriteLine("Please enter an integer:");**, prompts the user to enter an integer. This message guides the user on what type of input is expected from them.

Line 18-19

The line **string input = Console.ReadLine();** is responsible for reading user input from the console. The **Console.ReadLine()** method reads a line of text entered by the user and returns it as a string. In this case, the user is expected to enter a number.

The next line, **int number;**, declares an integer variable named number without initializing it. This variable will be used to store the converted value of the user's input.

The purpose of these lines is to capture the user's input as a string and prepare a variable to store the converted integer value of that input.

Line 21-35

The code snippet provided is responsible for determining whether the entered number is odd or even, and displaying the appropriate message based on the result.

Line 21, **if (int.TryParse(input, out number))**, checks if the user's input can be successfully converted to an integer. It uses the **int.TryParse()** method, which attempts to convert the **string input** to an integer. If the conversion is successful, the converted value is stored in the **number** variable, and the condition evaluates to true.

If the input is successfully converted to an integer, the program enters the if statement's block. Inside this block, the next lines of code are executed.

Line 23, **if (number % 2 == 0)** checks if the number stored in the **number** variable is divisible evenly by 2. If the remainder of the division **(number % 2)** is equal to 0, it means the number is even. In that case, the program displays the message "The number is even." using the **Console.WriteLine()** method.

If the remainder of the division is not equal to 0, the program enters the else block. This means the number is odd, and the program displays the message "The number is odd." using **Console.WriteLine()**.

If the input cannot be converted to an integer, the program skips the first if statement's block and enters the else block. In this case, it means the input was invalid or not a valid integer. The program displays the message "Invalid input. Please enter a valid integer." using **Console.WriteLine().**

These if-else statements help determine whether the entered number is odd or even and provide the appropriate output message to the user based on the result.

Line 37-38

Line 37, **Console.WriteLine("Press any key to exit.");** displays the message "Press any key to exit." on the console. This message informs the user that they can exit the program by pressing any key on their keyboard.

Line 38, **Console.ReadKey();**, waits for the user to press a key. It reads a single key input from the user without displaying it on the console. This line essentially acts as a pause in the program, allowing the user to view the program's output before it terminates.

By including these lines in the code, the program gives the user an opportunity to review the results or keep the console window open before it automatically closes. Once the user presses any key, the program proceeds to exit.

Summary

In this blog post, we explored the concept of odd and even numbers and learned how to create an Odd-Even Number Checker in C#. We began by understanding the properties and characteristics of odd and even numbers.

We then delved into the step-by-step process of building the Odd-Even Number Checker program. We covered topics such as setting up the C# console application, prompting the user for input, performing the necessary calculations using conditional statements and the modulo operator, and displaying the result. Additionally, we incorporated error handling techniques to handle invalid input.

Throughout the tutorial, we emphasized the importance of understanding the basics of conditional statements, user input handling, and programming logic. We provided a complete source code example for the Odd-Even Number Checker and encouraged readers to run and experiment with it.