Print Day in Word in CSharp

Welcome back to our journey of mastering C# console applications! In this lesson, we'll dive into the world of control flow with a focus on the versatile switch-case statement. Our mission is to create a program, aptly named PrintDayInWord, that takes a numeric input representing a day of the week and elegantly prints the corresponding day, from "Sunday" to "Saturday." Additionally, the program will handle invalid inputs gracefully, providing a robust user experience.

The Power of Switch-Case:

The switch-case statement is a powerful tool in C# that simplifies decision-making based on the value of a variable. In this lesson, we'll leverage its clarity and efficiency to enhance our console application. By the end, you'll have a solid understanding of how to employ this construct to streamline your code and make it more readable.

What to Expect:

1. User Input Handling:
    - The program will prompt users to enter a number (0-6) to represent a day of the week.
    - Robust input validation will ensure that the program gracefully handles invalid entries.
2. Switch-Case Magic:
    - We will explore the elegance of the switch-case statement to map numeric inputs to their corresponding days.
    - Each case will represent a day of the week, making the code clear and concise.
3. Graceful Handling of Invalid Input:
    - The default case within the switch block will ensure that the program responds gracefully to inputs outside the expected range.
    - Users will receive a friendly message indicating that the input is not a valid representation of a day.

Why Console Applications Matter:

Console applications are the backbone of many software systems. Understanding how to build robust, user-friendly console programs is a valuable skill that can be applied in various domains, from system administration tools to interactive command-line utilities.

Objectives

Objectives of the Lesson:

1. Understanding User Input Handling:
    - Objective: Introduce users to the concept of receiving input in a C# console application.
    - Discussion: The program prompts users to enter a number representing a day of the week. Handling user input is a fundamental aspect of console applications, and we will ensure users can interact with the program seamlessly.
2. Mastering the Switch-Case Statement:
    - Objective: Explore and demonstrate the functionality of the switch-case statement in C#.
    - Discussion: The switch-case statement is a powerful decision-making tool. By using it to map numeric inputs to corresponding days, learners will gain a solid understanding of how to efficiently manage multiple conditions in their code.
3. Ensuring Robust Input Validation:
    - Objective: Implement robust input validation to enhance the user experience.
    - Discussion: Handling invalid user inputs is crucial for maintaining the integrity of the program. We will employ techniques to validate user entries and gracefully guide them in the event of errors, fostering a positive user experience.
4. Enhancing Code Readability with Switch-Case:
    - Objective: Showcase the clarity and readability that the switch-case statement brings to code.
    - Discussion: One of the main advantages of using switch-case is its ability to make code more readable. By assigning each case to a day of the week, learners will observe how this construct improves code organization and comprehensibility.
5. Grasping the Default Case Handling:
    - Objective: Highlight the significance of the default case in handling unexpected inputs.
    - Discussion: The default case within the switch statement serves as a safety net. We will discuss its role in gracefully handling inputs that do not match any predefined cases, preventing unexpected behavior and ensuring a more robust program.

6. Applying Concepts to Real-World Scenarios:
   - Objective: Encourage learners to think about practical applications of switch-case statements in their projects.
   - Discussion: Understanding how to use the switch-case statement in the context of a day-of-week mapping is a stepping stone. We will discuss how learners can apply similar principles to other scenarios, fostering a deeper comprehension of the concept.
7. Emphasizing the Importance of Console Applications:
   - Objective: Convey the significance of console applications and their widespread use.
   - Discussion: Console applications are the building blocks of many software systems. By mastering the creation of robust and user-friendly console programs, learners will gain a skill set applicable to various domains, contributing to their versatility as developers.

By the end of this lesson, participants should feel confident in their ability to handle user input, leverage the switch-case statement effectively, and create console applications that are not only functional but also elegant and user-friendly. These objectives lay the foundation for mastering C# console applications and can be applied to a broad range of programming scenarios.

Source code example

```csharp
using System;

class PrintDayInWord
{
    static void Main()
    {
        Console.WriteLine("iNetTutor.com: ");
        Console.Write("Enter a number (0-6) to represent a day: ");

        if (int.TryParse(Console.ReadLine(), out int day))
        {
            PrintDay(day);
        }
        else
        {
            Console.WriteLine("Invalid input. Please enter a valid number (0-6).");
        }
    }

    static void PrintDay(int day)
    {
        switch (day)
        {
            case 0:
                Console.WriteLine("Sunday");
                break;
27.             case 1:
28.                 Console.WriteLine("Monday");
29.                 break;
30.             case 2:
31.                 Console.WriteLine("Tuesday");
32.                 break;
33.             case 3:
34.                 Console.WriteLine("Wednesday");
35.                 break;
36.             case 4:
37.                 Console.WriteLine("Thursday");
38.                 break;
39.             case 5:
40.                 Console.WriteLine("Friday");
41.                 break;
42.             case 6:
43.                 Console.WriteLine("Saturday");
44.                 break;
45.             default:
46.                 Console.WriteLine("Not a valid day");
47.                 break;
48.         }
49.         Console.ReadKey();
50.     }
51. }
```

Explanation

1. User Input (line 8):

This line prompts the user to enter a number between 0 and 6 to represent a day of the week.

2. Input Validation (line 10-17):

- The Console.ReadLine() method reads the user input as a string.
- int.TryParse attempts to convert the string input to an integer (day).
- If the conversion is successful, the code inside the if block is executed.

3. Calling the PrintDay Method (line 12):

If the input is a valid integer, the program calls the PrintDay method, passing the user's input (day) as an argument.

4. PrintDay Method (line 20-48):

- The PrintDay method takes the user's input (day) and uses a switch statement to match it to a case.
- For each valid case (0 to 6), it prints the corresponding day of the week using Console.WriteLine.

- If the input doesn't match any case, it enters the default case and prints "Not a valid day."
- Console.ReadKey() is used to keep the console window open until a key is pressed.

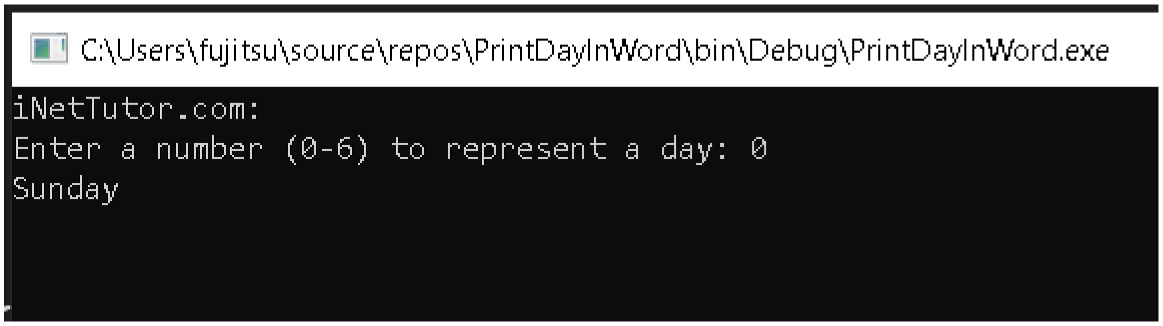
5. Closing Remarks:

The program ends, and the console window remains open for the user to view the output until they press a key.

Purpose:

- The purpose of this program is to demonstrate the use of the **switch** statement in C# for decision-making based on a numeric input. It provides a simple and interactive way for users to input a number and receive the corresponding day of the week as output. Additionally, it incorporates input validation to handle cases where the user enters invalid data, enhancing the program's robustness.

Output



```
C:\Users\fujitsu\source\repos\PrintDayInWord\bin\Debug\PrintDayInWord.exe
iNetTutor.com:
Enter a number (0-6) to represent a day: 0
Sunday
```

Summary

In this lesson, we delved into the essentials of C# console applications by creating the "PrintDayInWord" program. The primary focus was on mastering the switch-case statement, enhancing user input handling, and improving code readability. Users were prompted to input a number representing a day of the week, with robust validation ensuring a seamless experience. The switch-case statement demonstrated its power in efficiently handling multiple conditions, with each case representing a day of the week for clear and organized code. The inclusion of a default case served as a safety net for unexpected inputs, printing a message for invalid days. Beyond the technical aspects, the lesson encouraged learners to consider real-world applications of switch-case statements and highlighted the significance of console applications as foundational elements in software development. This lesson lays a solid foundation for further exploration into C# programming and practical application development.

Exercises and Assessment

Exercise: Objective: Reinforce understanding of the switch-case statement and user input handling in C# console applications.

Instructions:

1. Extend the "PrintDayInWord" program to include additional cases for handling different scenarios. For example, you might add cases for printing a message when the user enters the weekend (Saturday or Sunday) or for handling special days like holidays.
2. Implement a loop structure that allows the user to continuously input numbers and see the corresponding day until they choose to exit.
3. Enhance the program by incorporating more user-friendly messages and improving the overall user experience.

Assessment: Objective: Evaluate learners' proficiency in utilizing switch-case statements, handling user input, and enhancing code structure.

Criteria:

1. Correct Implementation: Assess whether the extended program correctly uses the switch-case statement to handle new cases and scenarios.
2. Input Handling: Check if the program effectively handles user input, providing clear messages for invalid inputs and maintaining robustness.
3. Loop Structure: Evaluate the implementation of the loop structure, ensuring it allows continuous input until the user chooses to exit.
4. User Experience: Consider the clarity and friendliness of the messages displayed to the user, providing helpful information and guidance.
5. Code Readability: Assess the overall code structure for readability, making sure it adheres to best practices and principles discussed in the lesson.

Additional Challenge (Optional): For an additional challenge, encourage learners to create a more dynamic program that allows users to define their own custom cases for specific numeric inputs. This could involve incorporating user-defined messages or actions for a given input. This challenge will test their ability to generalize concepts learned in the lesson to handle more complex scenarios.

Meta Description

"Dive into C# console apps with 'PrintDayInWord.' Master switch-case, handle input, and enhance code. Elevate coding skills in this practical lesson."