International University of Sarajevo

CS305 Programming Languages

CLI Habit Tracker

Project Report

Professor: Mirza Selimović

Students:

Ethem Kesim 210302205

Emre Çubuk 220302303

Table Of Contents

1. Introduction

This project is a C# Console Application (CLI) designed to help users track their daily habits.
The system allows adding new habits, marking them as completed, tracking streak progression, viewing historical activity, and persisting data using JSON serialization.

The goal of the project is to demonstrate:

- Object-oriented programming (OOP) principles
- File I/O operations
- Date and time manipulation
- Control structures and functions
- Data persistence with JSON
- Basic application design using the command-line interface

2. Project Objectives

The primary objectives of the Habit Tracker are:

1. Allow users to create and manage habit entries
2. Track daily completion and maintain streaks
3. Store data persistently so it is not lost between sessions
4. Display all habit details in an organized format
5. Provide a lightweight but functional productivity tool

3. System Design Overview

The project consists of **two main source files**:

- Habit.cs → Represents the habit model (class with properties)
- Program.cs → Contains the application logic, menu, and functions

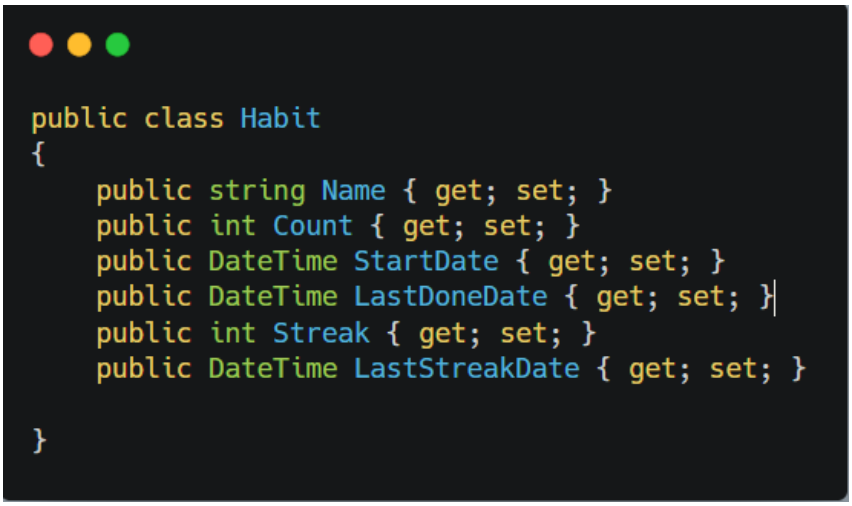
The application uses the following core concepts:

- Class design (Habit class)

- Lists (List<Habit>) for storing habits in memory
- Serialization/Deserialization for saving/loading
- DateTime operations for streak logic
- A continuous while loop for CLI interaction

4. Class Structure: Habit.cs

The Habit class is the fundamental building block of the program. Each habit created by the user is represented as an instance of this class.



```
public class Habit
{
    public string Name { get; set; }
    public int Count { get; set; }
    public DateTime StartDate { get; set; }
    public DateTime LastDoneDate { get; set; }
    public int Streak { get; set; }
    public DateTime LastStreakDate { get; set; }

}
```

Property	Description
Name	The name/title of the habit (e.g., “Study”, “Gym”).
Count	Total number of times the habit has been completed.
StartDate	The date when the user created the habit.
LastDoneDate	The timestamp of the most recent completion.
Streak	The number of consecutive days the habit has been completed.
LastStreakDate	Tracks the last date the streak was updated.

This structure enables detailed tracking and supports streak calculations based on user activity

5. Application Logic (Program.cs)

The main program handles:

- User interaction
- Habit creation
- Completion tracking
- Deleting habits
- Saving/loading JSON data
- Displaying habit information

Below are the major functions with explanations.

5.1 Menu Loop

```
while (true)
{
    Console.WriteLine("\n--- Habit Tracker ---");
    Console.WriteLine("1. Add Habit");
    Console.WriteLine("2. List Habits");
    Console.WriteLine("3. Mark Habit as Done");
    Console.WriteLine("4. Delete Habit");
    Console.WriteLine("5. Exit");
    Console.Write("Select: ");

    string choice = Console.ReadLine();
```

```
    switch (choice)
    {
        case "1":
            AddHabit();
            break;

        case "2":
            ListHabits();
            break;

        case "3":
            MarkDone();
            break;

        case "4":
            DeleteHabit();
            break;

        case "5":
            Save();
            Console.WriteLine("Saved, goodbye!");
            return;

        default:
            Console.WriteLine("Invalid selection");
            break;
    }
}
```

Function:

The loop runs continuously until the user chooses option 5 (Exit).
This acts as the central navigation system of the application.

5.2 AddHabit() – Creating a New Habit

```
void AddHabit()
{
    Console.Write("Enter habit name: ");
    string habitName = Console.ReadLine();

    habits.Add(new Habit
{
    Name = habitName,
    Count = 0,
    StartDate = DateTime.Now,
    LastDoneDate = DateTime.MinValue,
    Streak = 0,
    LastStreakDate = DateTime.MinValue
});
```

Explanation:

- Creates a new Habit object
- Initializes tracking variables
- Sets the start date
- Ensures streak starts from 0
- Adds the habit to the in-memory list

5.3 ListHabits() – Display Habit Details

```
void ListHabits()
{
    if (habits.Count == 0)
    {
        Console.WriteLine("No habits added yet.");
        return;
    }

    Console.WriteLine("\n--- Habit List ---");

    for (int i = 0; i < habits.Count; i++)
    {
        var h = habits[i];

        Console.WriteLine($"\n{i + 1}. {h.Name}");
        Console.WriteLine($"   Started      : {h.StartDate}");
        Console.WriteLine($"   Last Done    : {h.LastDoneDate}");
        Console.WriteLine($"   Done Count   : {h.Count}");

        Console.WriteLine($"   Streak       : {h.Streak} days");
    }

    Console.WriteLine("\n---------------------------------\n");
}
```

Displays all important habit data in a clean, readable format.

5.4 MarkDone() – Updating Streak and Count

```
void MarkDone()
{
    ListHabits();

    Console.Write("Which habit number? ");
    int index = int.Parse(Console.ReadLine()) - 1;

    if (index < 0 || index >= habits.Count)
    {
        Console.WriteLine("Invalid selection.");
        return;
    }

    Habit h = habits[index];
    DateTime today = DateTime.Today;
```

```
    if (h.LastStreakDate == today)
    {
        Console.WriteLine("You already completed this habit today.");
        return;
    }

    if (h.LastStreakDate == today.AddDays(-1))
    {
        h.Streak++;
    }
    else
    {

        h.Streak = 1;
    }

    h.LastStreakDate = today;
    h.LastDoneDate = DateTime.Now;
    h.Count++;

    Console.WriteLine("Marked as done, streak updated.");
}
```

Explanation:

- If the user completed the habit yesterday → streak continues
- If more than 1 day passed → streak resets to 1
- If already completed today → no further increment
- Count and LastDoneDate are updated

This function implements the core behavior of habit tracking systems.

5.5 DeleteHabit() – Removing an Entry

```
void DeleteHabit()
{
    ListHabits();

    Console.Write("Which habit to delete? ");
    int index = int.Parse(Console.ReadLine()) - 1;

    if (index < 0 || index >= habits.Count)
    {
        Console.WriteLine("Invalid selection.");
        return;
    }

    habits.RemoveAt(index);
    Console.WriteLine("Habit deleted.");
}
```

Removes the selected habit from the list.

5.6 Save() – JSON Data Persistence

```
void Save()
{
    string json = JsonSerializer.Serialize(habits, new JsonSerializerOptions { WriteIndented = true });
    File.WriteAllText("habits.json", json);
}
```

Explanation:

- Converts the habit list to JSON text
- Saves it to a file
- Ensures user progress remains safe after exit

5.7 Loading Data at Startup

```
if (File.Exists("habits.json"))
{
    string json = File.ReadAllText("habits.json");
    habits = JsonSerializer.Deserialize<List<Habit>>(json);
}
```

This restores the saved state so the user continues where they left off.

6. File Handling and Data Persistence

The use of System.Text.Json ensures:

- Lightweight storage
- Human-readable habits.json file
- Automatic serialization/deserialization

Compared to databases, JSON is simpler and perfect for small applications.

7. User Interface Design (CLI)

The interface is text-based, using:

- Clear menu options
- Formatted output
- Structured blocks for each habit

Advantages of CLI:

- Fast
- Lightweight
- Easy to maintain
- No complex UI libraries needed

8. Possible Improvements

Future enhancements may include:

- Weekly/Monthly goal tracking
- Notifications or reminders
- Exporting progress to CSV
- Color-coded terminal UI
- SQLite or SQL database integration
- Graphical user interface (GUI) with WPF or WinForms

9. Conclusion

This project demonstrates a complete, functional habit tracking system using core programming concepts such as:

- OOP
- File persistence
- Date/time operations
- Loop-based UIs
- Data structures