Lab Assignment 10

Course: CS202 Software Tools and Techniques for CSE
Lab Topic: Development of C# Console Applications
Date: 27th March 2025

Objective

This lab introduces students to **.NET** development using **C#** in **Visual Studio**. The focus is on creating simple console applications to understand basic syntax, control structures, and object-oriented programming principles in C#.

Learning Outcomes

By the end of this lab, students will be able to:

- Set up and use Visual Studio for .NET development.
- Write and execute basic C# console applications.
- Implement fundamental programming constructs: loops, conditionals, and functions.
- Apply object-oriented programming concepts in C#.
- Understand the benefit of the Visual Studio Debugger.

Pre-Lab Requirements

- Operating System: Windows
- Software: Visual Studio 2022 (Community Edition), Visual Studio with .NET SDK
- Programming Language: C# (latest stable version)

Lab Activities:

(1) Setting Up .NET Development Environment.
   - Open Visual Studio and create a new **C# Console Application** project.
   - Ensure the target framework is .NET 6 or later.
   - Write a simple program (any) and run it.

(2) Understanding Basic Syntax and Control Structures
   - Write an object-oriented C# program that:
     - Accepts user input[1] for two numbers.
     - Performs addition, subtraction, multiplication, and division.
     - Uses `if-else` conditions to determine if the sum is even or odd.
     - Displays the results using **Console.WriteLine()**.

(3) Implementing Loops and Functions
   - Design an object-oriented C# program that:

[1] Use Visual Studio IDE suggestions to identify the method that should be called to achieve this.

- ❖ Uses a `for` loop to print numbers from 1 to 10.
- ❖ Uses a `while` loop to keep asking the user for input until they enter `exit`.
- ❖ Defines and calls a function that calculates the factorial of a number provided by the user.

(4) Perform Object-Oriented Programming in C#

- Create a class `Student` with:
  - ❖ Properties[2]: `Name`, `ID`, `Marks`.
  - ❖ A constructor[3] to initialize these values.
  - ❖ A method `getGrade()` that returns the grade based on marks (A, B, C, etc.).
  - ❖ A `Main()`[4] method to create and display student details.
- Create a sub-class `StudentIITGN` from `Student` with:
  - ❖ A new property: `Hostel_Name_IITGN`
  - ❖ A `Main()`[5] method to create and display IITGN student details.

(5) Exception Handling

- Modify the program from Activity 2 to handle exceptions:
- Use `try-catch` to handle **division-by-zero** errors.
- Ensure the program does not crash on invalid input.

(6) Debugging using Visual Studio Debugger

- For Activities (2) to (5), visually illustrate[6] program execution in `Debug` mode by inserting `breakpoint`s at appropriate places.
- Your illustration should cover operations: `step-in`, `step-over`, `step-out`.

Resources

- Lecture 9 Slides
- Lecture 10 Slides
- https://learn.microsoft.com/en-us/dotnet/csharp
- https://github.com/dotnet/dotnet-console-games
- https://visualstudio.microsoft.com
- https://dotnet.microsoft.com/en-us/download
- https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions
- https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/exceptions/exception-handling
- https://learn.microsoft.com/en-us/visualstudio/debugger
- https://learn.microsoft.com/en-us/dotnet/csharp/fundamentals/tutorials/inheritance

---

[2] Identify appropriate datatype in C# for these fields.
[3] Can you also define a copy-constructor (what does it mean!) for this class? Additionally, can you also overload constructors and call them as needed?

[4] What does the **Main()** method need to be static? See what happens if you try to call **Main()** from itself. Additionally, observe what happens if we spell **Main()** as **main()**?
[5] What happens when both base-class and derived-class have their corresponding `Main()` methods?
[6] Screenshots are acceptable.