

Lab 9

Introduction:

In this lab we will analyze Python project dependencies using **pydeps** for python projects to compute the **fan-in** and **fan-out** for each module. We also **detect cycles** in the dependency graph. In this lab we also identify **isolated modules**, which are defined but not used. This analysis aids in improving code organization, reducing coupling, and enhancing modular design.

We will also work with Java to understand how modularity affects object-oriented programs. As part of this, we will use the **LCOM** (Lack of Cohesion in Methods) metric—a measure of how related the methods in a class are to each other. Using a **Java-based LCOM tool**, we will calculate the **LCOM1**, **LCOM2**, **LCOM3**, **LCOM4**, **LCOM5** and **YALCOM** scores of an open-source repository.

Setup and Tools:

1. Setting up a virtual environment.

```
sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ python3 -m venv Lab9
sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ source Lab9/bin/activate
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$
```

2. Installing “pydeps”.

```
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ pip install pydeps
Collecting pydeps
  Downloading pydeps-3.0.1-py3-none-any.whl.metadata (22 kB)
Collecting stdlib_list (from pydeps)
  Downloading stdlib_list-0.11.1-py3-none-any.whl.metadata (3.3 kB)
Downloading pydeps-3.0.1-py3-none-any.whl (47 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 47.6/47.6 kB 7.6 MB/s eta 0:00:00
Downloading stdlib_list-0.11.1-py3-none-any.whl (83 kB)
  ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 83.6/83.6 kB 2.9 MB/s eta 0:00:00
Installing collected packages: stdlib_list, pydeps
Successfully installed pydeps-3.0.1 stdlib_list-0.11.1
```

3. Installing “java”

```
sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/LCOM/src$ sudo apt install default-jre
[sudo] password for sumeet:
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
The following packages were automatically installed and are no longer required:
  acpid blt libns12 libtk8.6 mailcap tk8.6-blt2.5
Use 'sudo apt autoremove' to remove them.
The following additional packages will be installed:
  ca-certificates-java default-jre-headless fonts-dejavu-extra java-common
  libatk-wrapper-java libatk-wrapper-java-jni openjdk-21-jre
  openjdk-21-jre-headless
```

Methodology:

1. Finding a real-world repository using GitHub SEART.

General

Search by keyword in name Contains ▾ Python

License Has topic Uses Label

History and Activity

Number of Commits min max Number of Contributors 20 max

Number of Issues min max Number of Pull Requests min max

Number of Branches min max Number of Releases min max

Popularity Filters

Number of Stars min max Size of codebase ⓘ Non Blank Lines min max

Number of Watchers min max Code Lines 500 max

Number of Forks min max Comment Lines min max

Date-based Filters

Created Between dd/mm/yyyy dd/mm/yyyy

Last Commit Between dd/mm/yyyy dd/mm/yyyy

Additional Filters

Sorting Name ▾ Ascending ▾

Repository Characteristics

☐ Exclude Forks ☐ Has License

☐ Only Forks ☐ Has Open Issues

☐ Has Wiki ☐ Has Pull Requests

Search

- Language = Python.
- Minimum 20 contributors.
- Minimum 500 lines of code.

2. Selected repository [Link](#)

 [@b01001001/spectree](#) 🔗

Commits: 500 👁️ Watchers: 7 ☆ Stars: 330 🍴 Forks: 79

🕒 Total Issues: 115 🔧 Total Pull Req.s: 274 🌿 Branches: 5 👤 Contributors: 40

🔴 Open Issues: 9 🔖 Open Pull Req.s: 4 📦 Releases: 86 📦 Size: 990 B

+ Created: 2019-12-01 📅 Updated: 2025-03-04 📅 Last Push: 2025-03-04 📅 Last Commit: 2025-03-04

<> Code Lines: 12,222 📄 Comment Lines: 954 📄 Blank Lines: 1,790

Last Commit SHA: [f4b8306ae17e1bd8139fff8cf207b12525a4317](#)

Show More

3. Running pydeps and storing output in “out.json”

```
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ pydeps --show-deps spectree/ > out.json
```

4. Python script for computing fan-in fan-out

```
analysis.py U X
analysis.py > ...
1  import json
2
3  with open('out.json') as f:
4      data = json.load(f)
5
6  fan_in = {}
7  fan_out = {}
8
9  for module, details in data.items():
10     deps = details.get('imports', [])
11     fan_out[module] = len(deps)
12     for dep in deps:
13         fan_in[dep] = fan_in.get(dep, 0) + 1
14
15  all_modules = set(fan_in.keys()) | set(fan_out.keys())
16
17  print(f"{'Module':<35} | {'Fan-in':>7} | {'Fan-out':>8}")
18  print("-" * 60)
19  for module in sorted(all_modules):
20      fi = fan_in.get(module, 0)
21      fo = fan_out.get(module, 0)
22      print(f"{module:<35} | {fi:>7} | {fo:>8}")
23
24
25  with open("fan_analysis.json", "w") as f:
26      json.dump({"fan_in": fan_in, "fan_out": fan_out}, f, indent=4)
27
```

5. Python script for cycle detection:

```
cycle_detection.py X
spectree > cycle_detection.py > cycle_detection
1  import json
2
3  with open('out.json') as f:
4      data = json.load(f)
5
6  cycle = None
7
8  def cycle_detection(path, cur_node):
9
10     global cycle
11
12     if cur_node in path:
13         cycle = path + [cur_node]
14         return True
15
16     if 'imports' not in data[cur_node].keys():
17         return False
18
19     for node in data[cur_node]['imports']:
20         if cycle_detection(path + [cur_node], node):
21             return True
22
23     return False
24
25 if cycle_detection([], '__main__'):
26     print("Found Cycle")
27     print(cycle)
28 else:
29     print("No Cycle found")
30
```

6. Python script for isolated modules

```
isolated.py U X
~/Sumeet/Study/STT/spectree/isolated.py • Untracked
1  import json
2  import os
3
4  with open('out.json') as f:
5      data = json.load(f)
6
7  project_root = '/home/sumeet/Sumeet/Study/STT/spectree/spectree'
8  python_files = []
9  for root, dirs, files in os.walk(project_root):
10     for file in files:
11         if file.endswith('.py'):
12             python_files.append(os.path.join(root, file))
13
14
15  paths = []
16  info = data.values()
17  for i in info:
18      paths.append(i['path'])
19
20
21  num_isolated = 0
22  for file in python_files:
23      if file not in paths :
24          print(file)
25          num_isolated += 1
26
27  print("Number of isolated files: ", num_isolated)
```

7. Selecting a JAVA project

General

Search by keyword in name Contains ▾

License Has topic Uses Label

History and Activity

Number of Commits Number of Contributors

Number of Issues Number of Pull Requests

Number of Branches Number of Releases

Popularity Filters

Number of Stars Size of codebase ⓘ

Number of Watchers Non Blank Lines

Number of Forks Code Lines

Comment Lines

Date-based Filters

Created Between

Last Commit Between

Additional Filters

Sorting

Repository Characteristics

☐ Exclude Forks ☐ Has License

☐ Only Forks ☐ Has Open Issues

☐ Has Wiki ☐ Has Pull Requests

- Language = Java
- Min 1000 lines of code

8. Selected repository

 [00aj99/nocturnespy-client](#) 🔖

Commits: 2 👁 Watchers: 2 ☆ Stars: 14 🍴 Forks: 16

🕒 Total Issues: N/A 🔗 Total Pull Req: N/A 🌿 Branches: 1 👤 Contributors: N/A

🔴 Open Issues: N/A 🔗 Open Pull Req: N/A 📦 Releases: 0 📦 Size: 126 B

+ Created: 2018-06-09 📅 Updated: 2020-12-04 📅 Last Push: 2018-04-20 📅 Last Commit: 2018-04-20

<> Code Lines: 2,241 🗨 Comment Lines: 55 Blank Lines: 496

Last Commit SHA: 475f4f9341908f598d5d275c5b84021bf13d6449

[Show More](#)

9. Running LCOM.jar

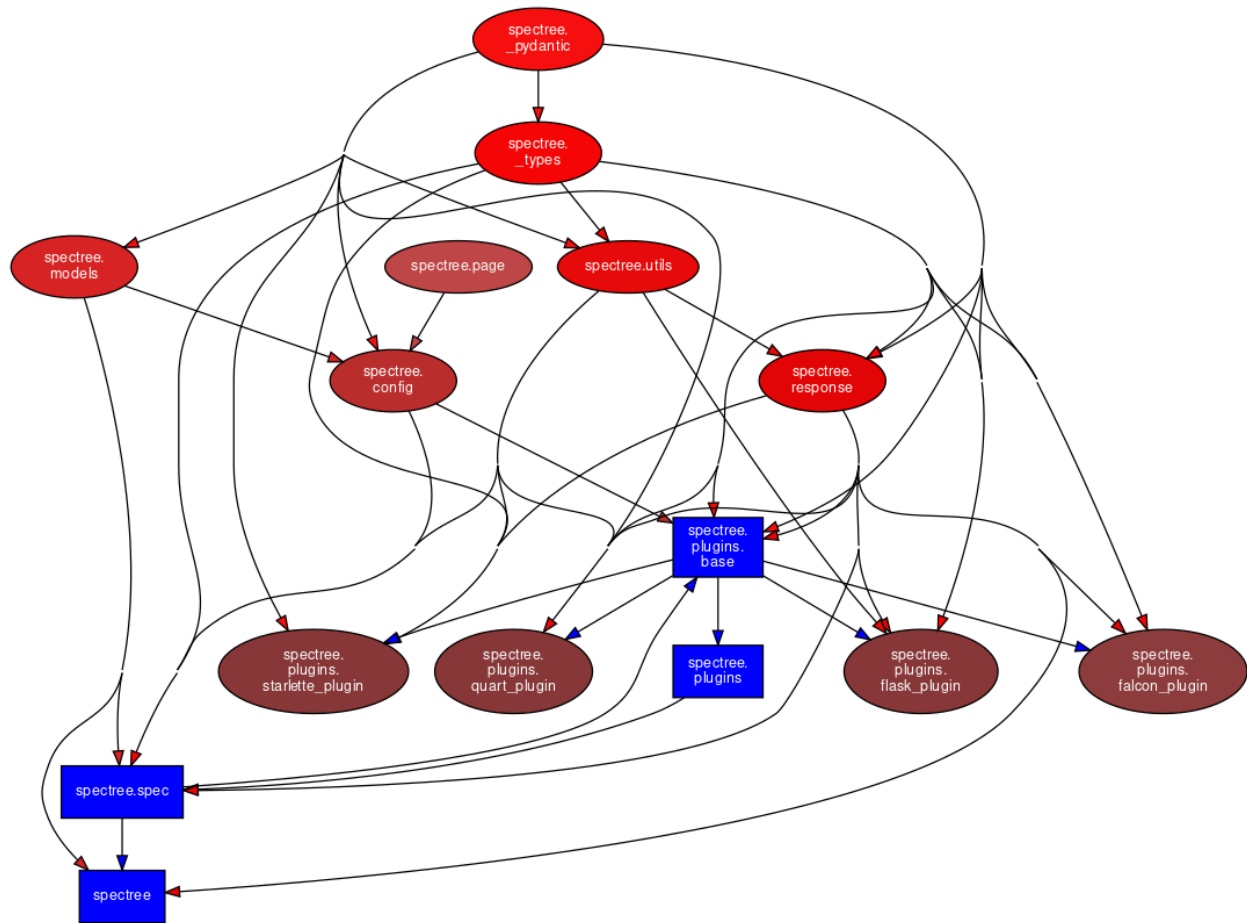
```
suneet@suneet-G5-5505:~/Suneet/Study/STT$ java -jar LCOM.jar -i NocturneSpy-Client/ -o .
Parsing the source code ...
Resolving symbols...
Computing metrics...
Done.
```

10. Python script to arrange the packages according to average LCOM value and take the top 5.

```
analysis.py ×
analysis.py > ...
1  import pandas as pd
2
3  data = pd.read_csv('TypeMetrics.csv')
4
5  data['average'] = data[['LCOM1', 'LCOM2', 'LCOM3', 'LCOM4', 'LCOM5', 'YALCOM']].mean(axis=1)
6  data = data.sort_values(by='average', ascending=False, ignore_index=True).head(5)
7
8  print(data)
9
10 data.to_csv('TypeMetrics_sorted.csv', index=False)
```


Result and Analysis:

1. Pydeps dependency graph



2. Fan-in Fan-out

```
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ python3 analysis.py
Module | Fan-in | Fan-out
-----|-----|-----
__main__ | 0 | 16
spectree | 6 | 3
spectree._pydantic | 11 | 0
spectree._types | 9 | 1
spectree.analysis | 1 | 0
spectree.config | 3 | 3
spectree.models | 4 | 1
spectree.page | 2 | 0
spectree.plugins | 6 | 1
spectree.plugins.base | 6 | 6
spectree.plugins.falcon_plugin | 1 | 6
spectree.plugins.flask_plugin | 1 | 7
spectree.plugins.quart_plugin | 1 | 7
spectree.plugins.starlette_plugin | 1 | 7
spectree.response | 8 | 3
spectree.spec | 3 | 6
spectree.utils | 6 | 2
```

3. Analyzing the generated dependency graph

- **Highly coupled modules.**

“spectree._pydantic” has highest fan-in of 11

```
Module | Fan-in | Fan-out
-----|-----|-----
spectree._pydantic | 11 | 0
```

- **Cyclic dependencies.**

```
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ python3 cycle_detection.py
Found Cycle
['__main__', 'spectree', 'spectree.spec', 'spectree.plugins', 'spectree.plugins.base',
'spectree']
```

- **Isolated modules**

```
(Lab9) sumeet@sumeet-G5-5505:~/Sumeet/Study/STT/spectree$ python3 isolated.py
Number of isolated files: 0
```

4. Modules with high fan-in are at high risk of breaking of system since many modules depend on them.

- **spectree._pyndatic**
- **spectree._types**
- **spectree.response**

5. Output of LCOM.jar

| Project Name | Package Name | Type Name | LCOM1 | LCOM2 | LCOM3 | LCOM4 | LCOM5 | YALCOM |
|--------------------|-------------------------------|---------------------|-------|-------|-------|-------|-------------------|-------------------|
| NocturneSpy-Client | com.android.adobot.utils | CalendarUtils | 1 | 1 | 2 | 1 | 1 | 0 |
| NocturneSpy-Client | com.android.adobot.http | Http | 1 | 0 | 1 | 1 | 0.5 | 0 |
| NocturneSpy-Client | com.android.adobot.http | HttpRequest | 6 | 2 | 1 | 1 | 0.9375 | 0 |
| NocturneSpy-Client | com.android.adobot.http | HttpCallback | 0 | 0 | 1 | 1 | 0 | -1 |
| NocturneSpy-Client | com.android.adobot | CommonParams | 28 | 20 | 1 | 1 | 0.888888888888889 | 0 |
| NocturneSpy-Client | com.android.adobot | Constants | 0 | 0 | 0 | 0 | 0 | -1 |
| NocturneSpy-Client | com.android.adobot | SMSSwatcher | 0 | 0 | 1 | 1 | 0 | 0 |
| NocturneSpy-Client | com.android.adobot | NetworkWatcher | 0 | 0 | 1 | 1 | 0 | 0 |
| NocturneSpy-Client | com.android.adobot | CommandService | 46 | 26 | 5 | 4 | 0.681818181818182 | 0.166666666666667 |
| NocturneSpy-Client | com.android.adobot.activities | BaseActivity | 10 | 0 | 5 | 5 | 0 | 1 |
| NocturneSpy-Client | com.android.adobot.activities | SetupActivity | 2 | 1 | 2 | 1 | 1 | 0 |
| NocturneSpy-Client | com.android.adobot.activities | PermissionsActivity | 20 | 19 | 6 | 1 | 0.958333333333333 | 0 |
| NocturneSpy-Client | com.android.adobot.activities | UpdateActivity | 3 | 0 | 2 | 1 | 0.666666666666667 | 0 |
| NocturneSpy-Client | com.android.adobot.job | CheckJobCreator | 0 | 0 | 1 | 1 | 0 | 0 |
| NocturneSpy-Client | com.android.adobot.job | CheckJob | 1 | 0 | 2 | 2 | 1 | 1 |
| NocturneSpy-Client | com.android.adobot.tasks | GetSmsTask | 2 | 1 | 2 | 1 | 0.5 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | GetCallLogsTask | 2 | 1 | 2 | 1 | 0.75 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | GetCalendarTask | 0 | 0 | 2 | 2 | 0.833333333333333 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | LocationMonitor | 22 | 16 | 4 | 1 | 0.821428571428571 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | SendSmsTask | 10 | 5 | 3 | 2 | 0.85 | 0.333333333333333 |
| NocturneSpy-Client | com.android.adobot.tasks | GetContactsTask | 0 | 0 | 2 | 2 | 0.833333333333333 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | TransferBotTask | 3 | 0 | 2 | 1 | 0.8 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | BaseTask | 3 | 0 | 1 | 1 | 0.5 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | UpdateAppTask | 2 | 1 | 2 | 1 | 0.5 | 0 |
| NocturneSpy-Client | com.android.adobot.tasks | SmsForwarder | 43 | 31 | 4 | 3 | 0.92 | 0.272727272727273 |
| NocturneSpy-Client | com.android.adobot.tasks | SmsObserver | 9 | 8 | 4 | 3 | 0.9 | 0.6 |
| NocturneSpy-Client | com.android.adobot.tasks | SendSmsThread | 0 | 0 | 1 | 1 | 0.5 | 0 |

6. Packages with high LCOM values:

| Project Name | Package Name | Type Name | LCOM1 | LCOM2 | LCOM3 | LCOM4 | LCOM5 | YALCOM |
|--------------|-------------------------------|---------------------|-------|-------|-------|-------|-------------------|-------------------|
| java | com.android.adobot.tasks | SmsForwarder | 43 | 31 | 4 | 3 | 0.92 | 0.272727272727273 |
| java | com.android.adobot | CommandService | 46 | 26 | 5 | 4 | 0.681818181818182 | 0.166666666666667 |
| java | com.android.adobot | CommonParams | 28 | 20 | 1 | 1 | 0.888888888888889 | 0 |
| java | com.android.adobot.activities | PermissionsActivity | 20 | 19 | 6 | 1 | 0.958333333333333 | 0 |
| java | com.android.adobot.tasks | LocationMonitor | 22 | 16 | 4 | 1 | 0.821428571428571 | 0 |

Conclusion:

Through this lab, we have explored **fan-in** and **fan-out** analysis, detected **cyclic dependencies**, and identified **isolated or untracked** Python modules. The insights derived from this analysis can guide refactoring efforts, highlight central utility components, and flag tightly coupled or unused code. Overall, this exercise deepens our understanding of software architecture and dependency management, which are key to scalable and maintainable codebases.

By using **LCOM** analysis in Java, we deepened our understanding of internal class design and the significance of maintaining strong cohesion within classes. Overall, the lab highlighted the importance of both **inter-module** and **intra-module** analysis in software engineering, emphasizing clean architecture and modular design principles for building robust applications.

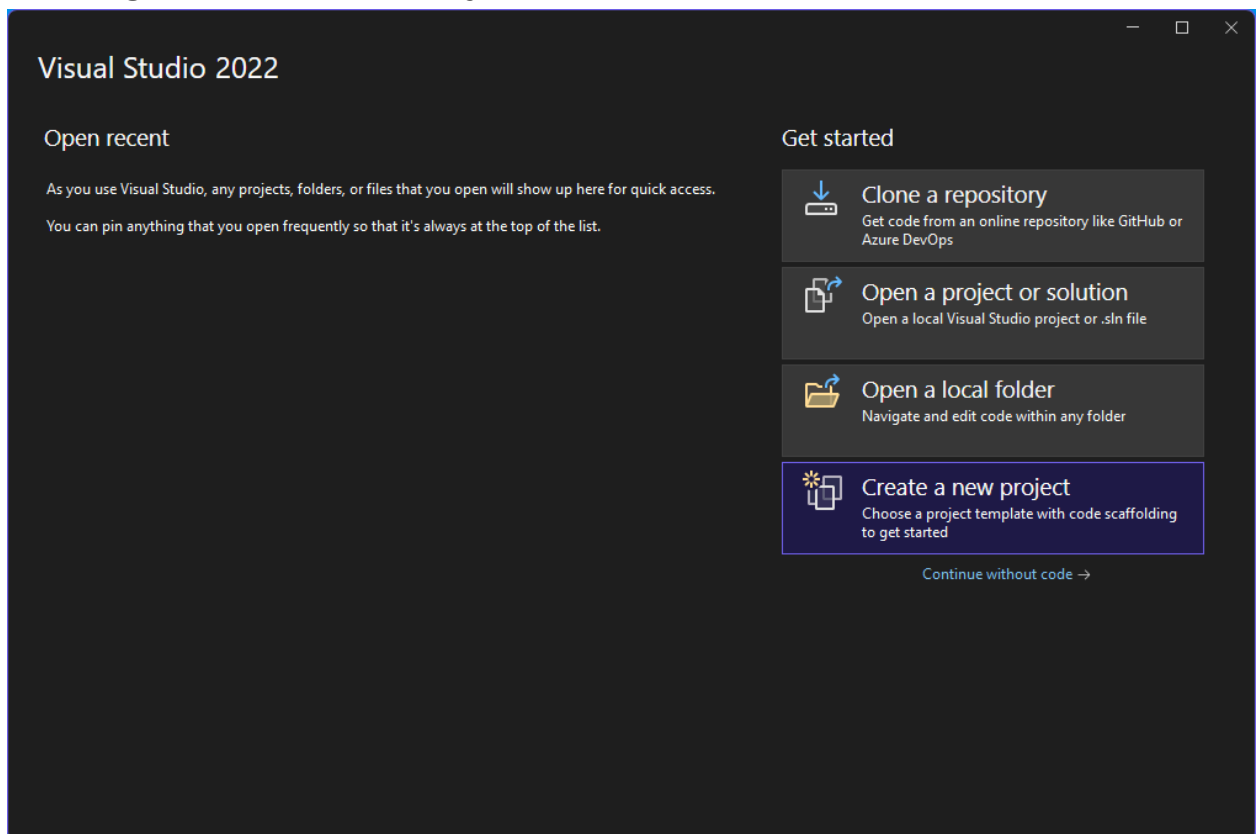
Lab 10

Introduction:

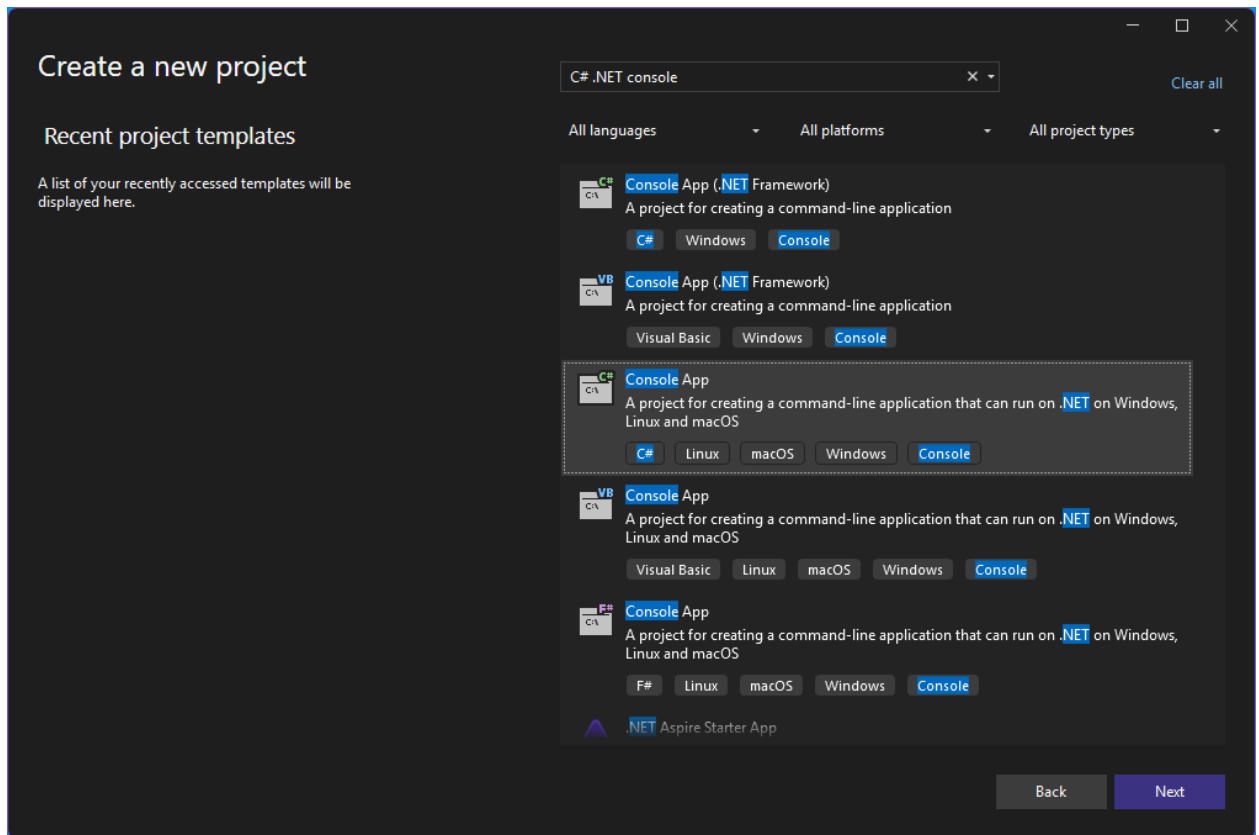
In this lab we will use **.NET** framework and **C#** language. The primary aim is to learn how to use **Visual Studio** and its **debugger**. By creating and executing console-based applications, we will learn about **loops**, **functions**, **object-oriented programming**, and **exception handling**.

Tools and Setup:

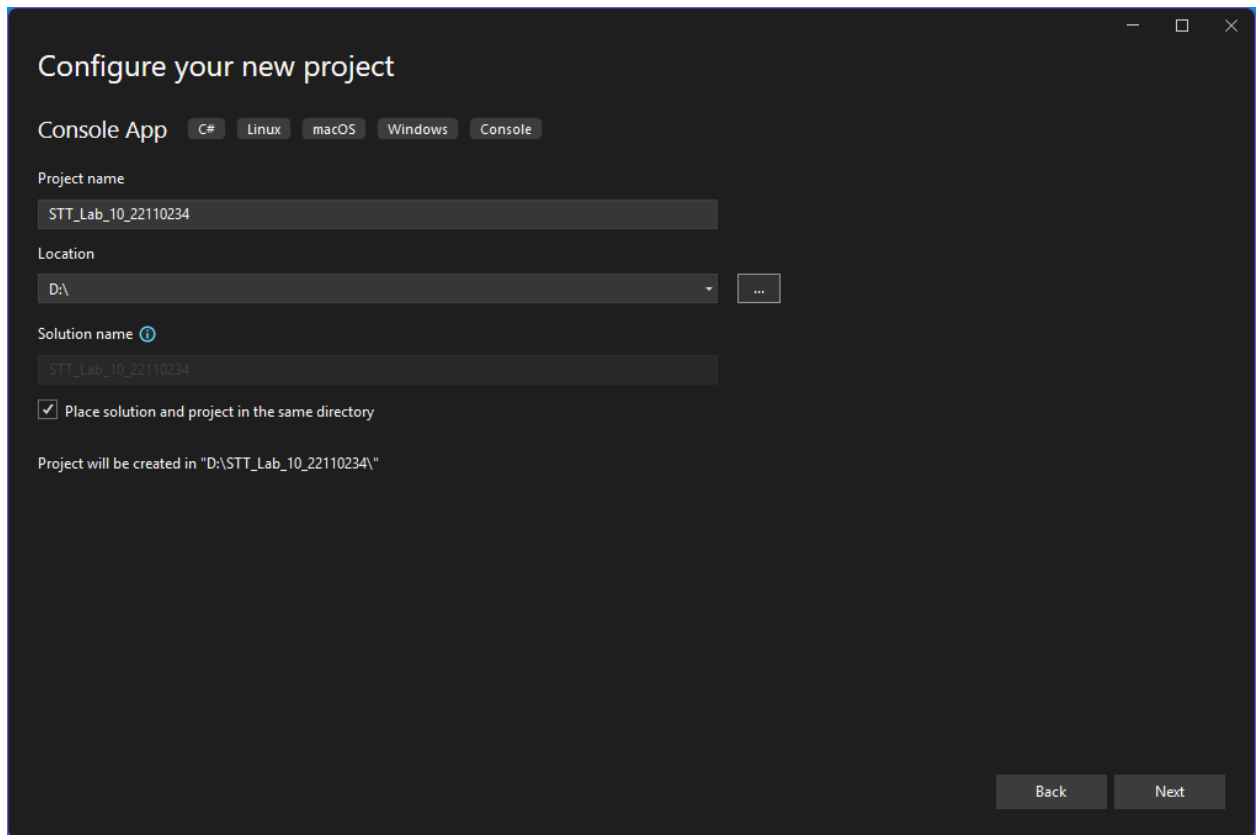
1. Creating a Visual Studio Project



2. Searching for .NET console app in C#



3. Creating the project



The screenshot shows the 'Configure your new project' dialog box in Visual Studio. The title bar includes standard window controls (minimize, maximize, close). The main title is 'Configure your new project'. Below it, the project type is 'Console App', and the language is 'C#'. Other available languages are 'Linux', 'macOS', 'Windows', and 'Console'. The 'Project name' field contains 'STT_Lab_10_22110234'. The 'Location' field shows 'D:\' with a dropdown arrow and a browse button ('...'). The 'Solution name' field, which has a help icon, also contains 'STT_Lab_10_22110234'. A checkbox labeled 'Place solution and project in the same directory' is checked. At the bottom, a summary line states 'Project will be created in "D:\STT_Lab_10_22110234\"'. The bottom right corner features 'Back' and 'Next' buttons.

Configure your new project

Console App C# Linux macOS Windows Console

Project name

STT_Lab_10_22110234

Location

D:\ ...

Solution name ⓘ

STT_Lab_10_22110234

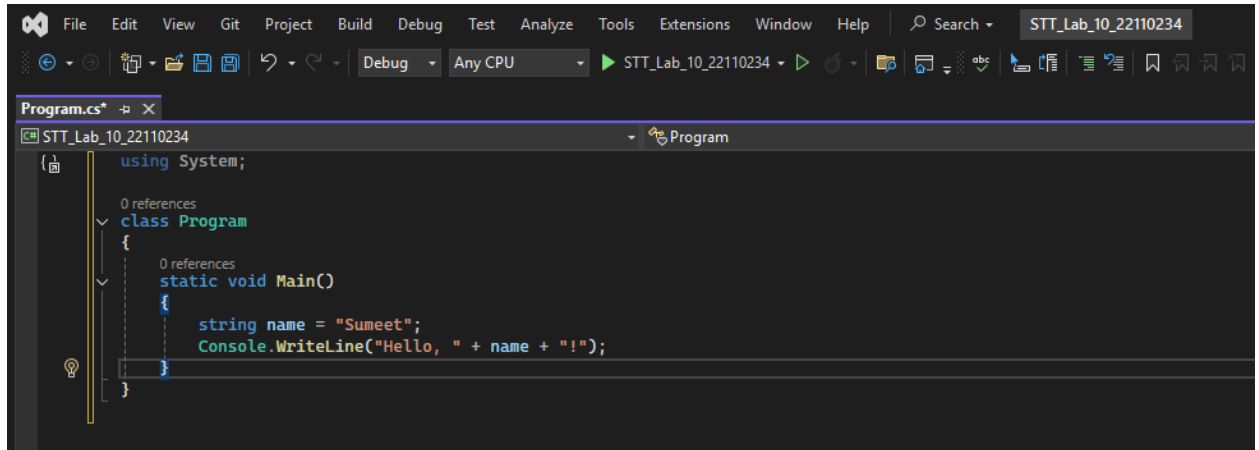
☒ Place solution and project in the same directory

Project will be created in "D:\STT_Lab_10_22110234\"

Back Next

Methodology:

1. Writing a simple C# program



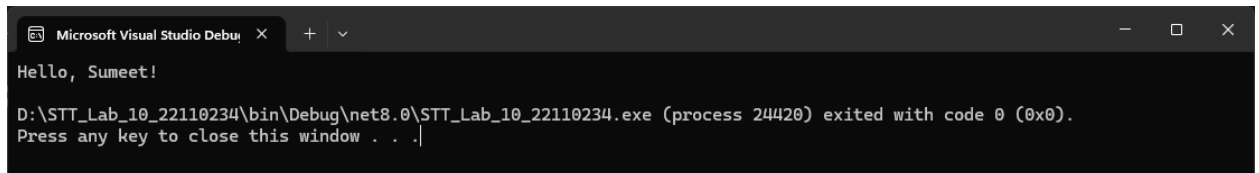
The screenshot shows the Visual Studio IDE with a C# program in a file named Program.cs. The code is as follows:

```
using System;

class Program
{
    static void Main()
    {
        string name = "Sumeet";
        Console.WriteLine("Hello, " + name + "!");
    }
}
```

The interface includes a menu bar (File, Edit, View, Git, Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help), a search bar, and a toolbar with icons for running, debugging, and other actions. The project name STT_Lab_10_22110234 is visible in the top right.

2. Executing the program



The screenshot shows the Microsoft Visual Studio Debug Console window. It displays the output of the program:

```
Hello, Sumeet!
```

Below the output, the console shows the program's exit status:

```
D:\STT_Lab_10_22110234\bin\Debug\net8.0\STT_Lab_10_22110234.exe (process 24420) exited with code 0 (0x0).  
Press any key to close this window . . .
```


3. Writing C# program for basic calculator

```
using System;

0 references
class Program
{
    1 reference
    static void PerformOperations(double num1, double num2)
    {
        double sum = num1 + num2;
        double difference = num1 - num2;
        double product = num1 * num2;
        double quotient = num1 / num2;

        Console.WriteLine("Addition: " + sum);
        Console.WriteLine("Subtraction: " + difference);
        Console.WriteLine("Multiplication: " + product);
        Console.WriteLine("Division: " + quotient);

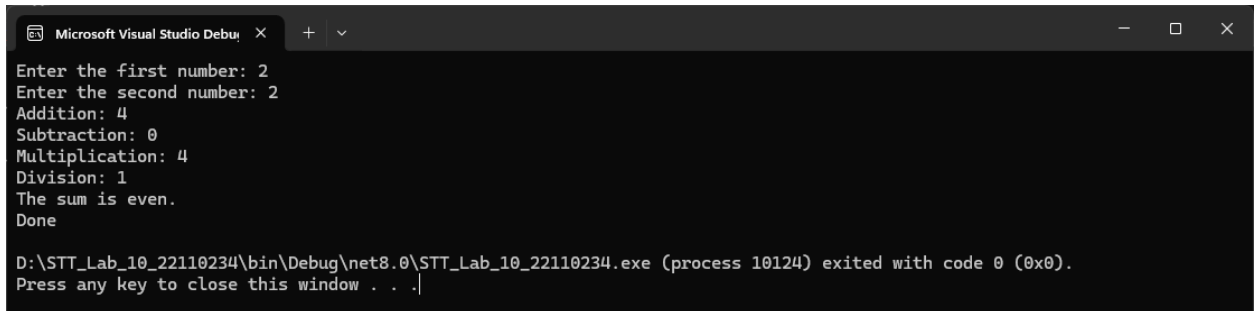
        if (sum % 2 == 0)
        {
            Console.WriteLine("The sum is even.");
        }
        else
        {
            Console.WriteLine("The sum is odd.");
        }
    }

    0 references
    static void Main()
    {
        Console.Write("Enter the first number: ");
        double number1 = Convert.ToDouble(Console.ReadLine());

        Console.Write("Enter the second number: ");
        double number2 = Convert.ToDouble(Console.ReadLine());

        PerformOperations(number1, number2);
        Console.WriteLine("Done");
    }
}
```

4. Executing the calculator program

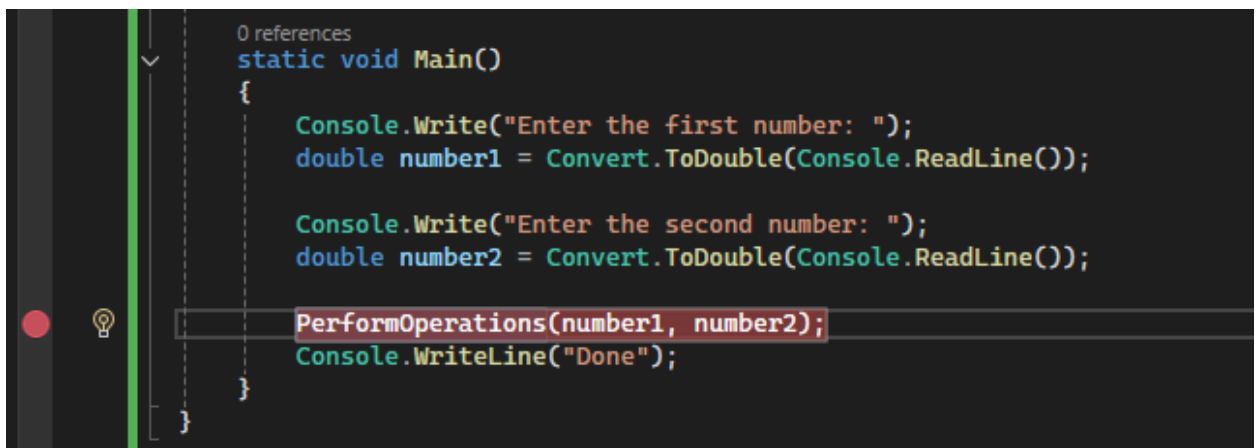


The screenshot shows the Microsoft Visual Studio Debug Console window. The output text is as follows:

```
Enter the first number: 2
Enter the second number: 2
Addition: 4
Subtraction: 0
Multiplication: 4
Division: 1
The sum is even.
Done

D:\STT_Lab_10_22110234\bin\Debug\net8.0\STT_Lab_10_22110234.exe (process 10124) exited with code 0 (0x0).
Press any key to close this window . . .|
```

5. Adding breakpoints



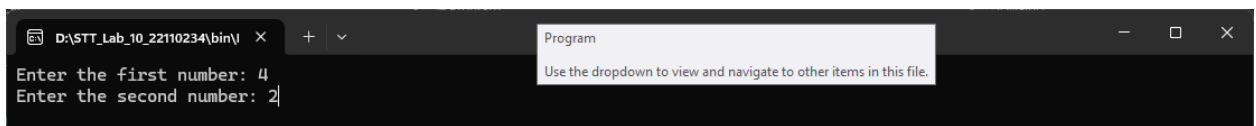
The screenshot shows the Visual Studio code editor with a C# file. A breakpoint (red dot) is set on the line `PerformOperations(number1, number2);`. The code is as follows:

```
0 references
static void Main()
{
    Console.Write("Enter the first number: ");
    double number1 = Convert.ToDouble(Console.ReadLine());

    Console.Write("Enter the second number: ");
    double number2 = Convert.ToDouble(Console.ReadLine());

    PerformOperations(number1, number2);
    Console.WriteLine("Done");
}
```

6. Running in debug mode



The screenshot shows the Visual Studio interface with the program running in debug mode. The output window shows the input and output of the program:

```
Enter the first number: 4
Enter the second number: 2|
```

The top of the window shows the file path `D:\STT_Lab_10_22110234\bin\` and a dropdown menu for the 'Program' window.

7. Implementing loops and functions

```
using System;

0 references
class Program
{
    1 reference
    public static void PrintNumbers()
    {
        for (int i = 1; i <= 10; i++)
        {
            Console.WriteLine(i);
        }
    }

    1 reference
    public static long CalculateFactorial(int number)
    {
        long factorial = 1;
        for (int i = 1; i <= number; i++)
        {
            factorial *= i;
        }
        return factorial;
    }

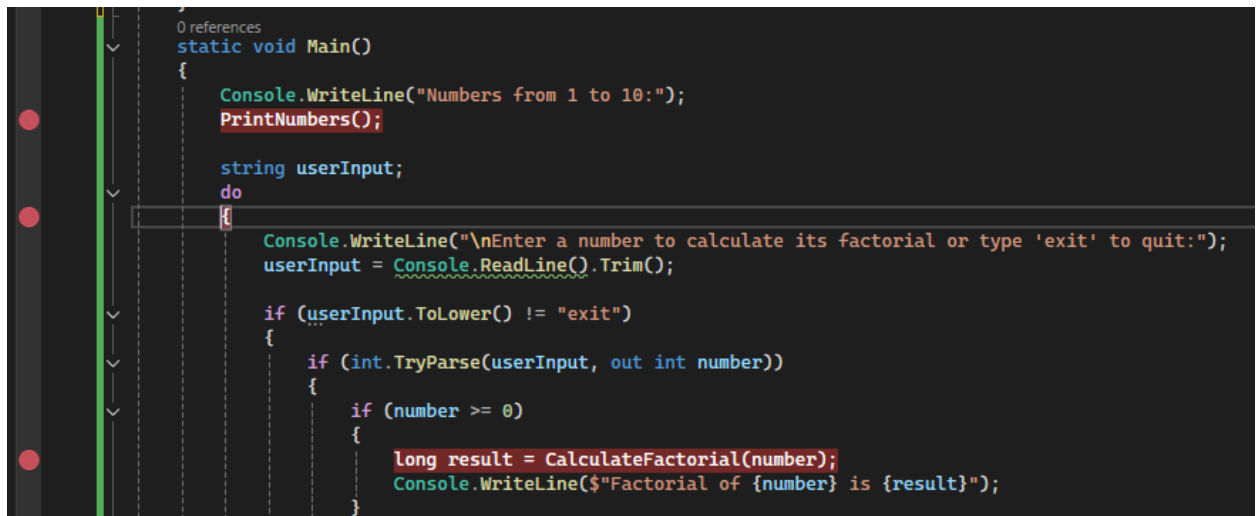
    0 references
    static void Main()
    {
        Console.WriteLine("Numbers from 1 to 10:");
        PrintNumbers();

        string userInput;
        do
        {
            Console.WriteLine("\nEnter a number to calculate its factorial or type 'exit' to quit:");
            userInput = Console.ReadLine().Trim();

            if (userInput.ToLower() != "exit")
            {
                if (int.TryParse(userInput, out int number))
                {
                    if (number >= 0)
                    {
                        long result = CalculateFactorial(number);
                        Console.WriteLine($"Factorial of {number} is {result}");
                    }
                    else
                    {
                        Console.WriteLine("Please enter a non-negative integer.");
                    }
                }
                else
                {
                    Console.WriteLine("Invalid input. Please enter a valid number.");
                }
            }
        } while (userInput.ToLower() != "exit");

        Console.WriteLine("Goodbye!");
    }
}
```

8. Adding breakpoints



The screenshot shows a code editor with a dark theme. On the left, a vertical toolbar contains several icons, including a red circle with a white dot, which is the breakpoint icon. A vertical green line is positioned at the start of the code, and a red dot (breakpoint) is placed on the line containing `PrintNumbers();`. The code is as follows:

```
0 references
static void Main()
{
    Console.WriteLine("Numbers from 1 to 10:");
    PrintNumbers();

    string userInput;
    do
    {
        Console.WriteLine("\nEnter a number to calculate its factorial or type 'exit' to quit:");
        userInput = Console.ReadLine().Trim();

        if (userInput.ToLower() != "exit")
        {
            if (int.TryParse(userInput, out int number))
            {
                if (number >= 0)
                {
                    long result = CalculateFactorial(number);
                    Console.WriteLine($"Factorial of {number} is {result}");
                }
            }
        }
    } while (userInput != "exit");
}
```

9. Object-oriented programming

```
using System;

6 references
class Student
{
    4 references
    public string Name { get; set; }
    4 references
    public int ID { get; set; }
    8 references
    public double Marks { get; set; }

    2 references
    public Student(string name, int id, double marks)
    {
        Name = name;
        ID = id;
        Marks = marks;
    }

    2 references
    public string GetGrade()
    {
        if (Marks >= 90)
        {
            return "A";
        }
        else if (Marks >= 80)
        {
            return "B";
        }
        else if (Marks >= 70)
        {
            return "C";
        }
        else if (Marks >= 60)
        {
            return "D";
        }
        else
        {
            return "F";
        }
    }
}
```

```

1 reference
public void DisplayDetails()
{
    Console.WriteLine("Student Name: " + Name);
    Console.WriteLine("Student ID: " + ID);
    Console.WriteLine("Student Marks: " + Marks);
    Console.WriteLine("Student Grade: " + GetGrade());
}

```

```

3 references
class StudentIITGN : Student
{
    2 references
    public string Hostel_Name_IITGN { get; set; }

    1 reference
    public StudentIITGN(Student student, string hostelName)
        : base(student.Name, student.ID, student.Marks)
    {
        Hostel_Name_IITGN = hostelName;
    }

    1 reference
    public void DisplayIITGNDetails()
    {
        Console.WriteLine("Student Name: " + Name);
        Console.WriteLine("Student ID: " + ID);
        Console.WriteLine("Student Marks: " + Marks);
        Console.WriteLine("Student Grade: " + GetGrade());
        Console.WriteLine("Hostel Name: " + Hostel_Name_IITGN);
    }
}

```

```

0 references
class Program
{
    0 references
    static void Main()
    {
        Student student = new Student("Sumeet", 22110234, 95);

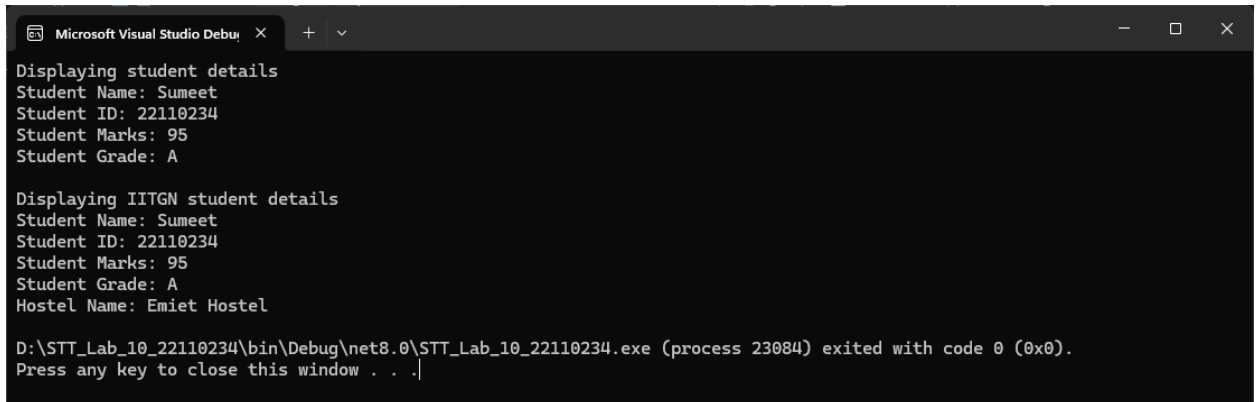
        StudentIITGN studentIITGN = new StudentIITGN(student, "Emiet Hostel");

        Console.WriteLine("Displaying student details");
        student.DisplayDetails();

        Console.WriteLine("\nDisplaying IITGN student details");
        studentIITGN.DisplayIITGNDetails();
    }
}

```

10. Executing the program



The screenshot shows the Microsoft Visual Studio Debug Console window. The output text is as follows:

```
Microsoft Visual Studio Debug Console
+ -
Displaying student details
Student Name: Sumeet
Student ID: 22110234
Student Marks: 95
Student Grade: A

Displaying IITGN student details
Student Name: Sumeet
Student ID: 22110234
Student Marks: 95
Student Grade: A
Hostel Name: Emiet Hostel

D:\STT_Lab_10_22110234\bin\Debug\net8.0\STT_Lab_10_22110234.exe (process 23084) exited with code 0 (0x0).
Press any key to close this window . . .
```

11. Adding breakpoints



The screenshot shows the Visual Studio code editor with the `Program.cs` file open. A breakpoint (red dot) is set on the line `student.DisplayDetails();`. The code is as follows:

```
0 references
class Program
{
    0 references
    static void Main()
    {
        Student student = new Student("Sumeet", 22110234, 95);

        StudentIITGN studentIITGN = new StudentIITGN(student, "Emiet Hostel");

        Console.WriteLine("Displaying student details");
        student.DisplayDetails();

        Console.WriteLine("\nDisplaying IITGN student details");
        studentIITGN.DisplayIITGNDetails();
    }
}
```

12. Exception handling

```
using System;

0 references
class Program
{
    1 reference
    static void PerformOperations(double num1, double num2)
    {
        try
        {
            double sum = num1 + num2;
            double difference = num1 - num2;
            double product = num1 * num2;
            double quotient = num1 / num2;

            Console.WriteLine("Addition: " + sum);
            Console.WriteLine("Subtraction: " + difference);
            Console.WriteLine("Multiplication: " + product);
            Console.WriteLine("Division: " + quotient);

            if (sum % 2 == 0)
            {
                Console.WriteLine("The sum is even.");
            }
            else
            {
                Console.WriteLine("The sum is odd.");
            }
        }

        catch (DivideByZeroException)
        {
            Console.WriteLine("Cannot divide by zero");
        }
    }
}
```

```
0 references
static void Main()
{
    Console.Write("Enter the first number: ");
    double number1 = Convert.ToDouble(Console.ReadLine());

    Console.Write("Enter the second number: ");
    double number2 = Convert.ToDouble(Console.ReadLine());

    PerformOperations(number1, number2);
}
}
```


13. Executing the program



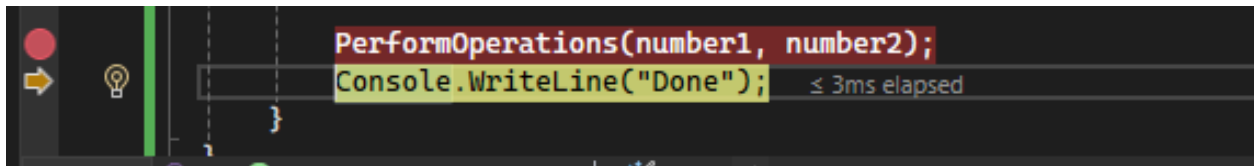
A screenshot of a Windows command prompt window. The title bar shows the file path "D:\STT_Lab_10_22110234\bin\" and standard window controls. The command prompt displays two lines of text: "Enter the first number: 4" and "Enter the second number: 4". The second line has a cursor at the end of the input "4".

```
D:\STT_Lab_10_22110234\bin> Enter the first number: 4
Enter the second number: 4|
```

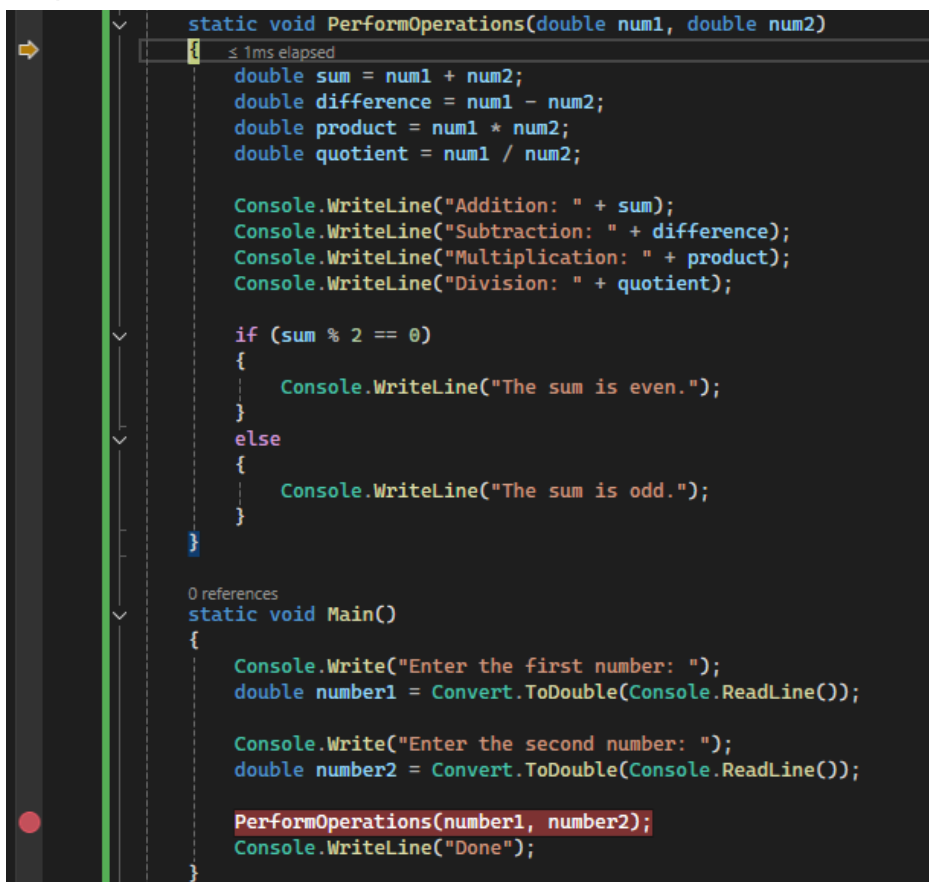
Results and Analysis:

Basic calculator program

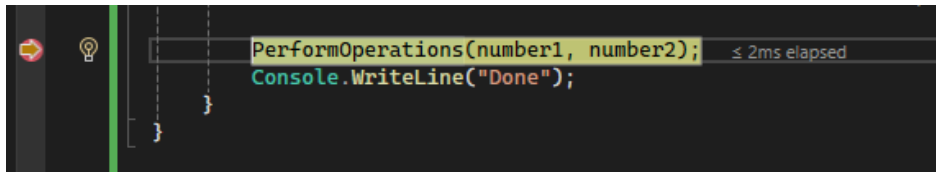
1. Step-over takes us to next line



2. Step-into takes us to the definition of the function

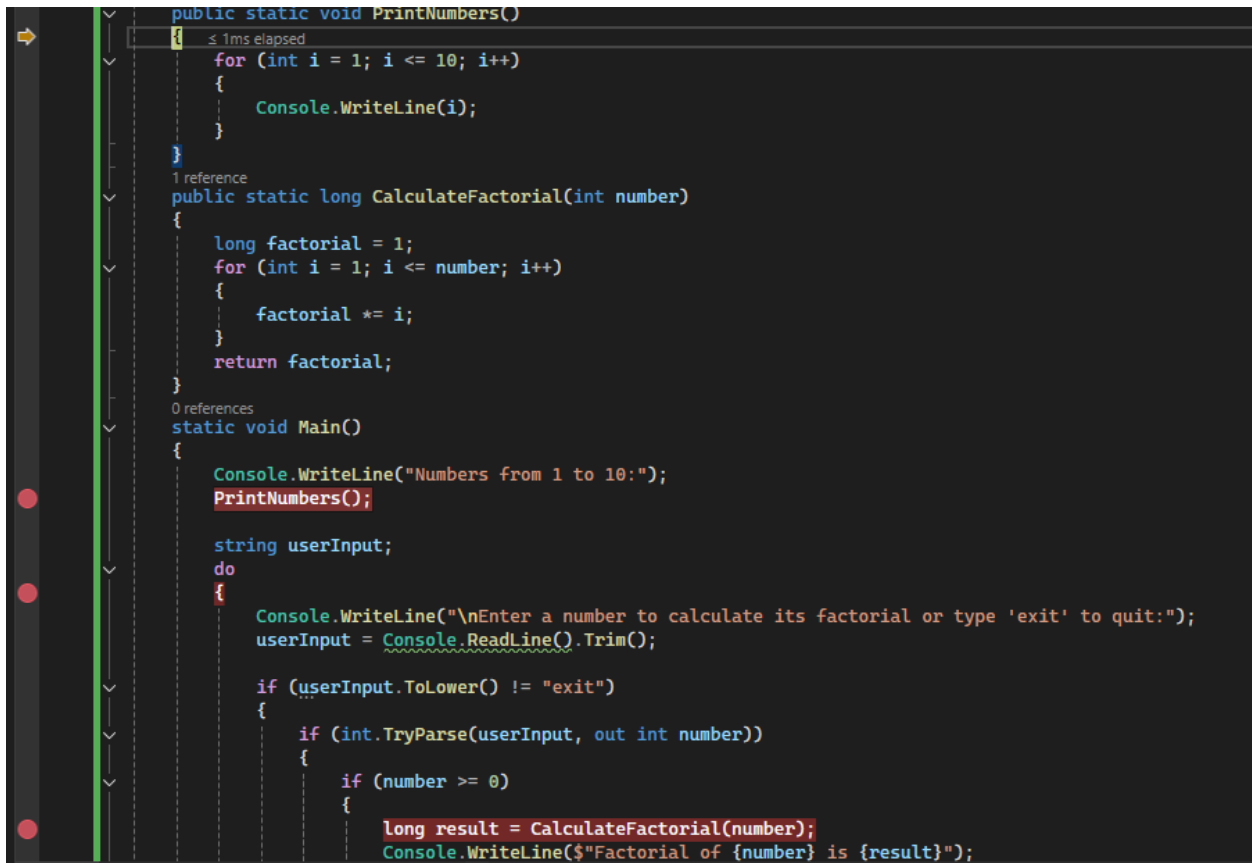


3. Step-out takes us out of the function execution and to the next line

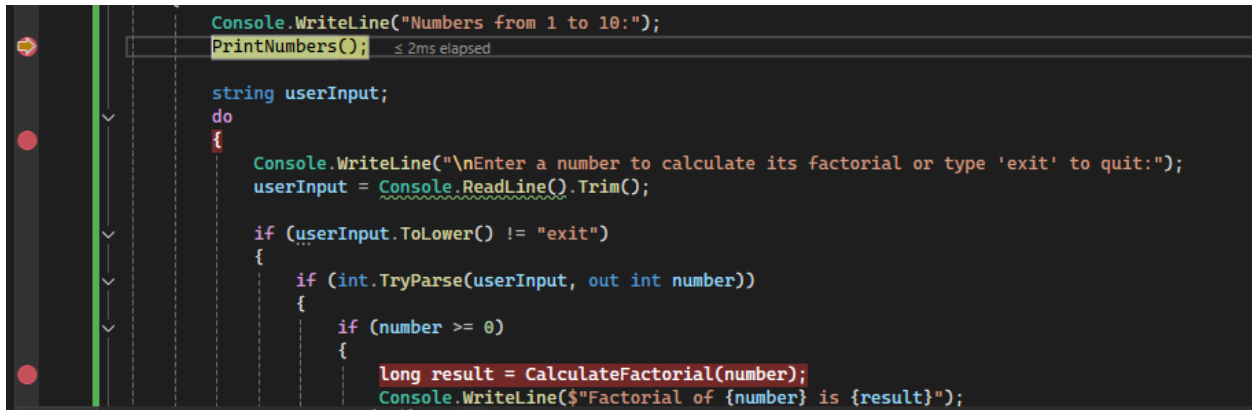


Loops and functions program

1. Step-into takes us to the start of function



2. Step-out takes us out of the function call

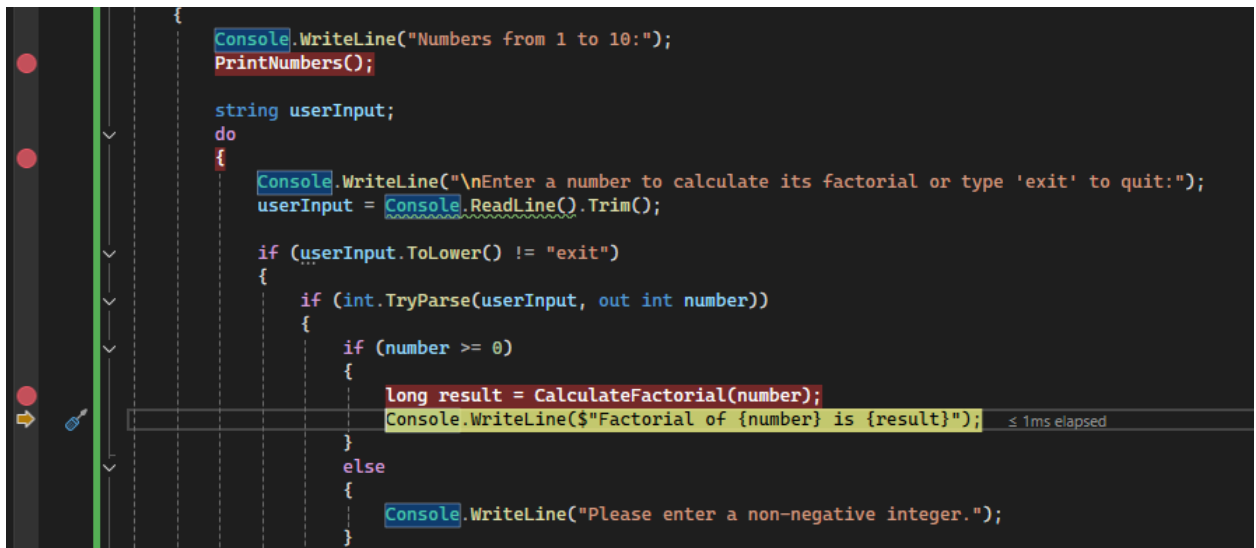


```
Console.WriteLine("Numbers from 1 to 10:");
PrintNumbers(); ≤ 2ms elapsed

string userInput;
do
{
    Console.WriteLine("\nEnter a number to calculate its factorial or type 'exit' to quit:");
    userInput = Console.ReadLine().Trim();

    if (userInput.ToLower() != "exit")
    {
        if (int.TryParse(userInput, out int number))
        {
            if (number >= 0)
            {
                long result = CalculateFactorial(number);
                Console.WriteLine($"Factorial of {number} is {result}");
            }
        }
    }
}
```

3. Step-over skips the function execution and take us to the next line



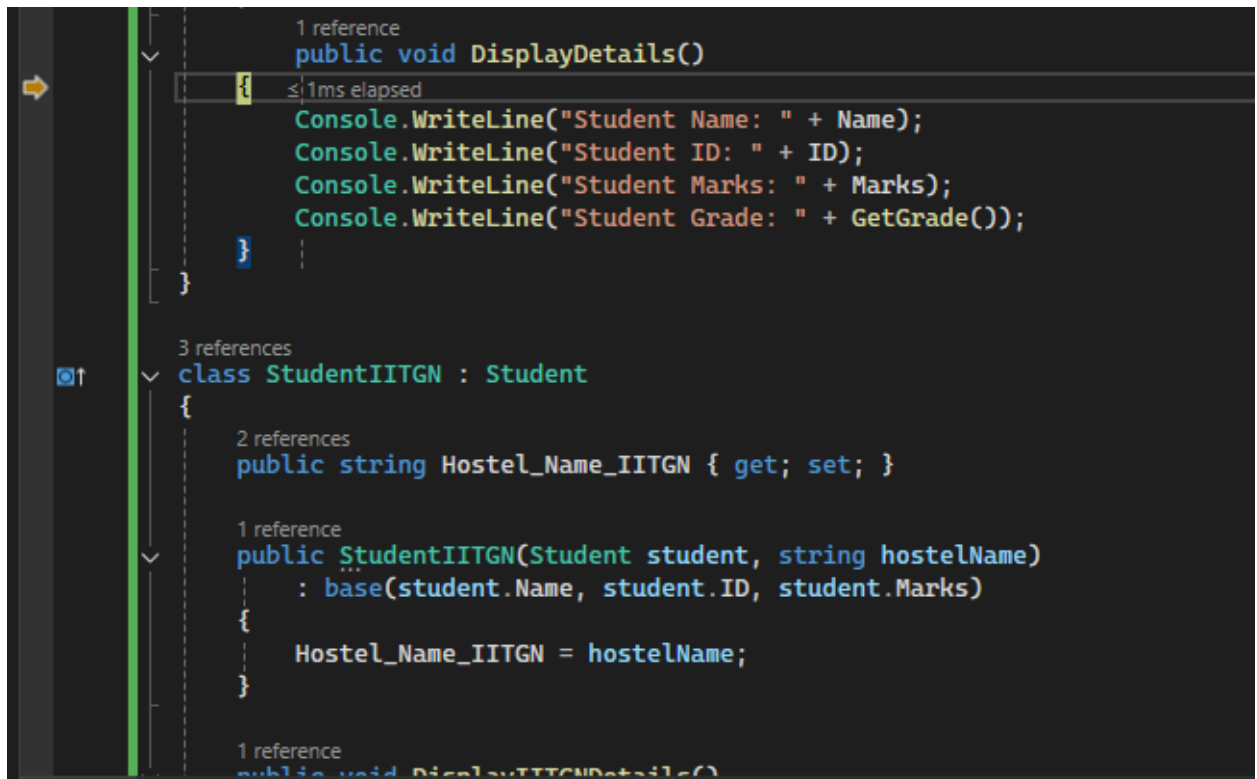
```
Console.WriteLine("Numbers from 1 to 10:");
PrintNumbers();

string userInput;
do
{
    Console.WriteLine("\nEnter a number to calculate its factorial or type 'exit' to quit:");
    userInput = Console.ReadLine().Trim();

    if (userInput.ToLower() != "exit")
    {
        if (int.TryParse(userInput, out int number))
        {
            if (number >= 0)
            {
                long result = CalculateFactorial(number);
                Console.WriteLine($"Factorial of {number} is {result}"); ≤ 1ms elapsed
            }
        }
        else
        {
            Console.WriteLine("Please enter a non-negative integer.");
        }
    }
}
```

Object-oriented program

1. Step-into takes us to the start of the function



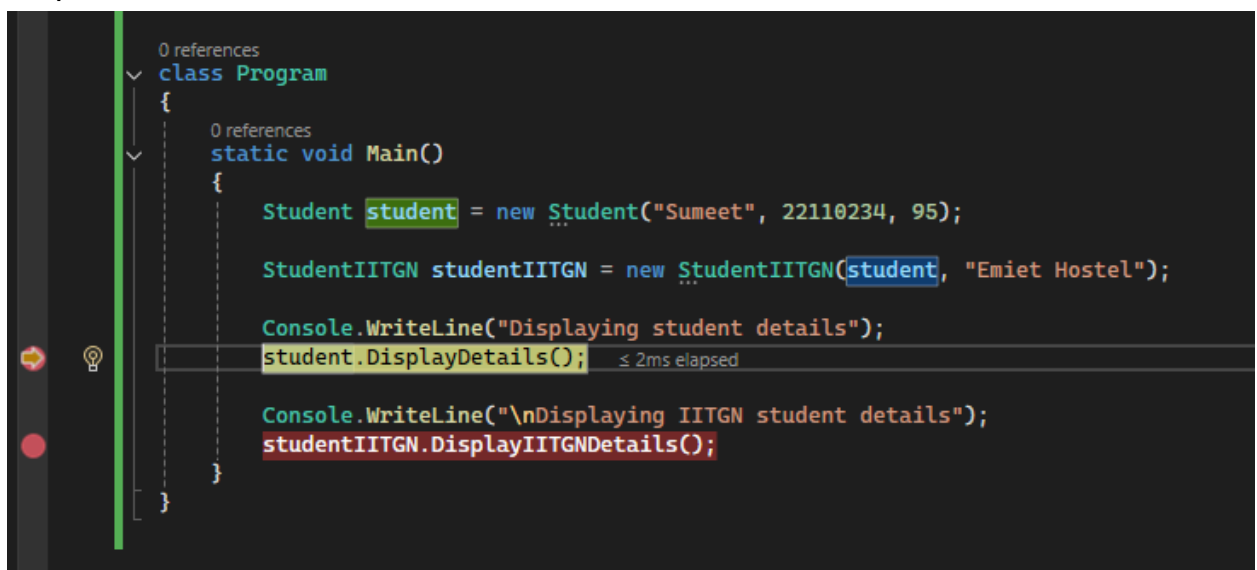
```
1 reference
public void DisplayDetails()
{
    Console.WriteLine("Student Name: " + Name);
    Console.WriteLine("Student ID: " + ID);
    Console.WriteLine("Student Marks: " + Marks);
    Console.WriteLine("Student Grade: " + GetGrade());
}

3 references
class StudentIITGN : Student
{
    2 references
    public string Hostel_Name_IITGN { get; set; }

    1 reference
    public StudentIITGN(Student student, string hostelName)
        : base(student.Name, student.ID, student.Marks)
    {
        Hostel_Name_IITGN = hostelName;
    }

    1 reference
    public void DisplayIITGNDetails()
```

2. Step-out takes us out of the function execution



```
0 references
class Program
{
    0 references
    static void Main()
    {
        Student student = new Student("Sumeet", 22110234, 95);

        StudentIITGN studentIITGN = new StudentIITGN(student, "Emiet Hostel");

        Console.WriteLine("Displaying student details");
        student.DisplayDetails();
        Console.WriteLine("\nDisplaying IITGN student details");
        studentIITGN.DisplayIITGNDetails();
    }
}
```

3. Step-over takes us to the next breakpoint



```
0 references
class Program
{
    0 references
    static void Main()
    {
        Student student = new Student("Sumeet", 22110234, 95);

        StudentIITGN studentIITGN = new StudentIITGN(student, "Emiet Hostel");

        Console.WriteLine("Displaying student details");
        student.DisplayDetails();

        Console.WriteLine("\nDisplaying IITGN student details");
        studentIITGN.DisplayIITGNDetails();
    }
}
```

Exception handling program

1. Step-into takes us to the function definition



```
static void PerformOperations(double num1, double num2)
{
    try
    {
        double sum = num1 + num2;
        double difference = num1 - num2;
        double product = num1 * num2;
        double quotient = num1 / num2;

        Console.WriteLine("Addition: " + sum);
        Console.WriteLine("Subtraction: " + difference);
        Console.WriteLine("Multiplication: " + product);
        Console.WriteLine("Division: " + quotient);

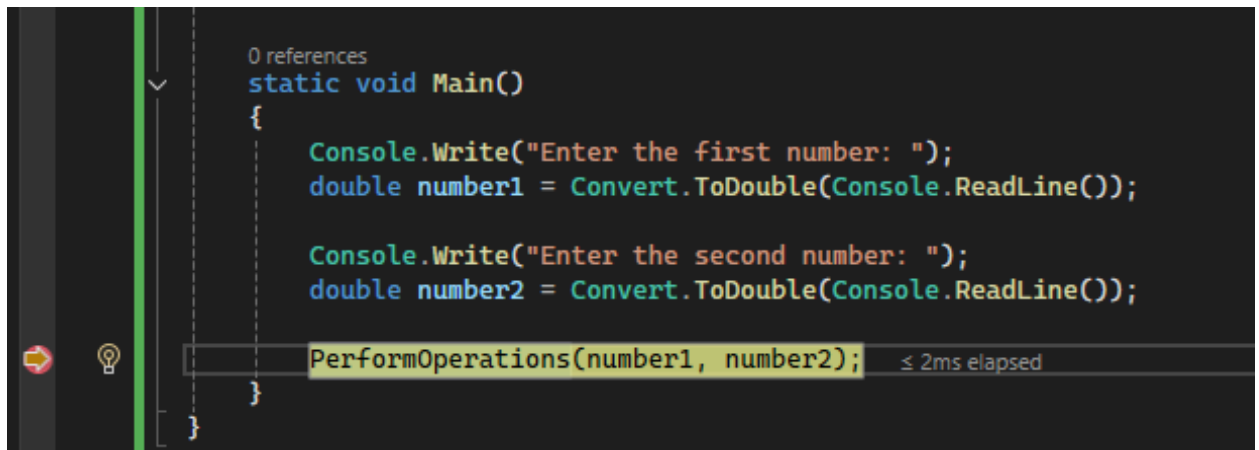
        if (sum % 2 == 0)
        {
            Console.WriteLine("The sum is even.");
        }
        else
        {
            Console.WriteLine("The sum is odd.");
        }
    }
    catch (DivideByZeroException)
    {
        Console.WriteLine("Cannot divide by zero");
    }
}

0 references
static void Main()
{
    Console.Write("Enter the first number: ");
    double number1 = Convert.ToDouble(Console.ReadLine());

    Console.Write("Enter the second number: ");
    double number2 = Convert.ToDouble(Console.ReadLine());

    PerformOperations(number1, number2);
}
```

2. Step-out takes us out of the function execution



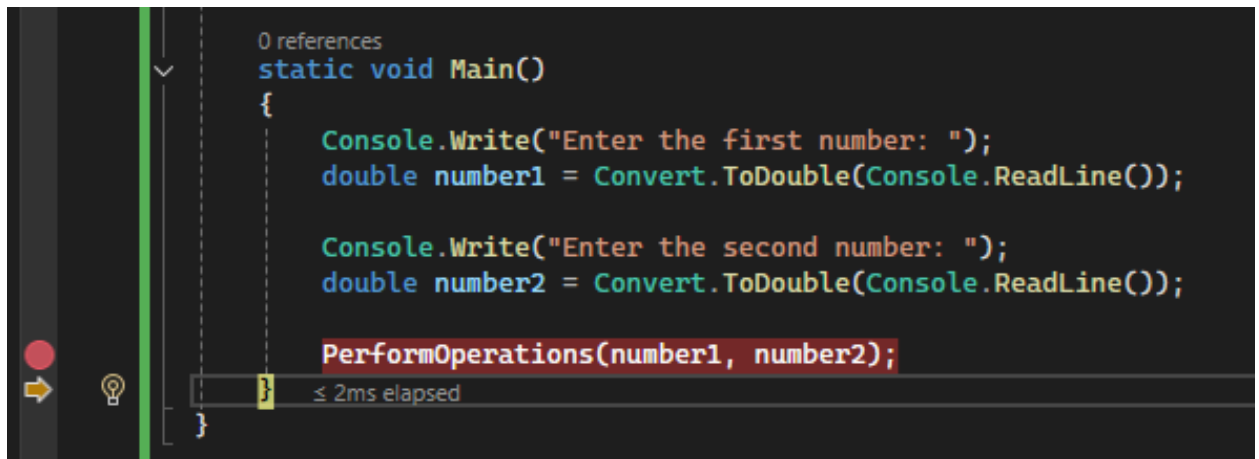
```
0 references
static void Main()
{
    Console.WriteLine("Enter the first number: ");
    double number1 = Convert.ToDouble(Console.ReadLine());

    Console.WriteLine("Enter the second number: ");
    double number2 = Convert.ToDouble(Console.ReadLine());

    PerformOperations(number1, number2);
}
```

≤ 2ms elapsed

3. Step-over takes us to the next line



```
0 references
static void Main()
{
    Console.WriteLine("Enter the first number: ");
    double number1 = Convert.ToDouble(Console.ReadLine());

    Console.WriteLine("Enter the second number: ");
    double number2 = Convert.ToDouble(Console.ReadLine());

    PerformOperations(number1, number2);
}
```

≤ 2ms elapsed

Conclusion:

In this lab, we were first introduced to the **C#** language and **Visual Studio debugger**. We explored the syntax of **C#** and learned about **object-oriented** design. We also did **exception-handling** such as **invalid input** and **division by zero** error. We used **Visual Studio's** debugging tools for **step-by-step** code analysis. We looked at **step-in**, **step-out** and **step-over**.