Software Tools and Techniques

LAB 9 and Lab10

LAB9:- Module Dependency and Cohesion Analysis

Overview:

This lab uses LCOM.jar for Java and pydeps for Python to analyze class cohesion and module dependencies. It assists in locating design issues that impact program modularity and maintainability, such as low cohesion, cyclic dependencies, and tight coupling.

Objective:

This lab aims to use cohesion and dependence metrics to study software design. We examine Python project dependencies using peeps to find cyclic, fan-in/fan-out, and highly connected modules. We measure class cohesiveness and identify low-cohesion classes for Java projects using LCOM.jar. This makes understanding design problems and directing reworking for improved modularity and maintainability easier.

Environmental Setup:

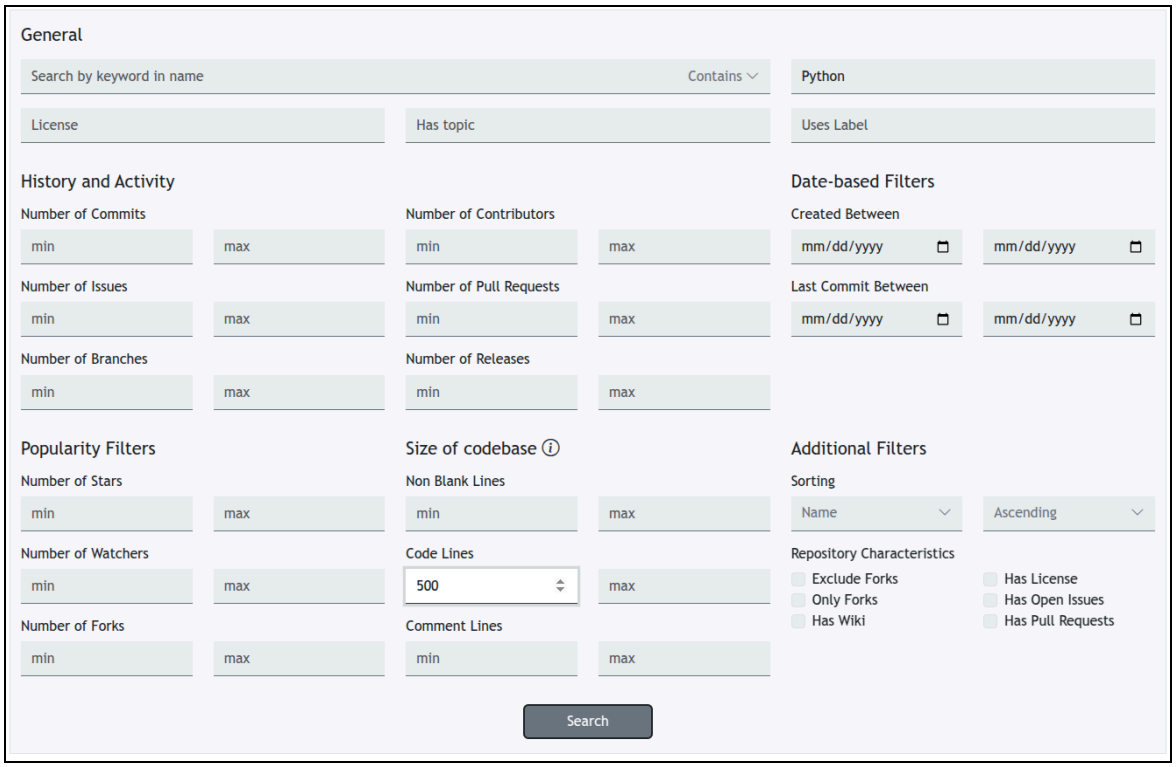
- Operating System: Ubuntu
- Programming language: Python, Java

Tools and versions used:

- Python version: 3.12.3
- Java version: 21.0.6
- Git version: 2.43.0

Methodology and Execution:

- Search the GitHub repository of the given criteria on the GitHub search engine.



https://github.com/pallets/jinja

- Create a folder named ‘Lab9’, and go to that folder, and clone the selected repository

```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ git clone https://github.com/pallets/jinja.git
Cloning into 'jinja'...
remote: Enumerating objects: 16506, done.
remote: Counting objects: 100% (472/472), done.
remote: Compressing objects: 100% (160/160), done.
remote: Total 16506 (delta 404), reused 312 (delta 312), pack-reused 16034 (from 4)
Receiving objects: 100% (16506/16506), 6.90 MiB | 6.20 MiB/s, done.
Resolving deltas: 100% (10981/10981), done.
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$
```

- Create and activate a virtual environment named ‘lab9_env’. Check python3 and git version.

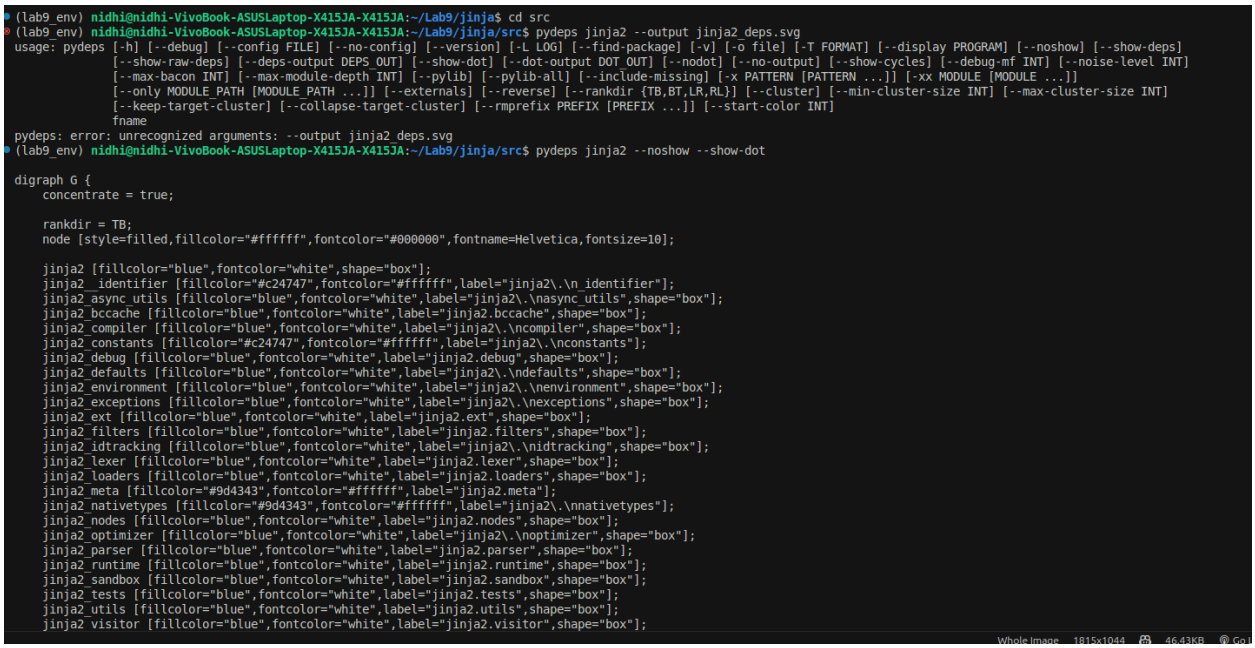
```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ python3 -m venv lab9_env
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ source lab9_env/bin/activate
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$
```

```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ python3 --version
Python 3.12.3
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ git --version
git version 2.43.0
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$
```

- Install pydeps and check its version.

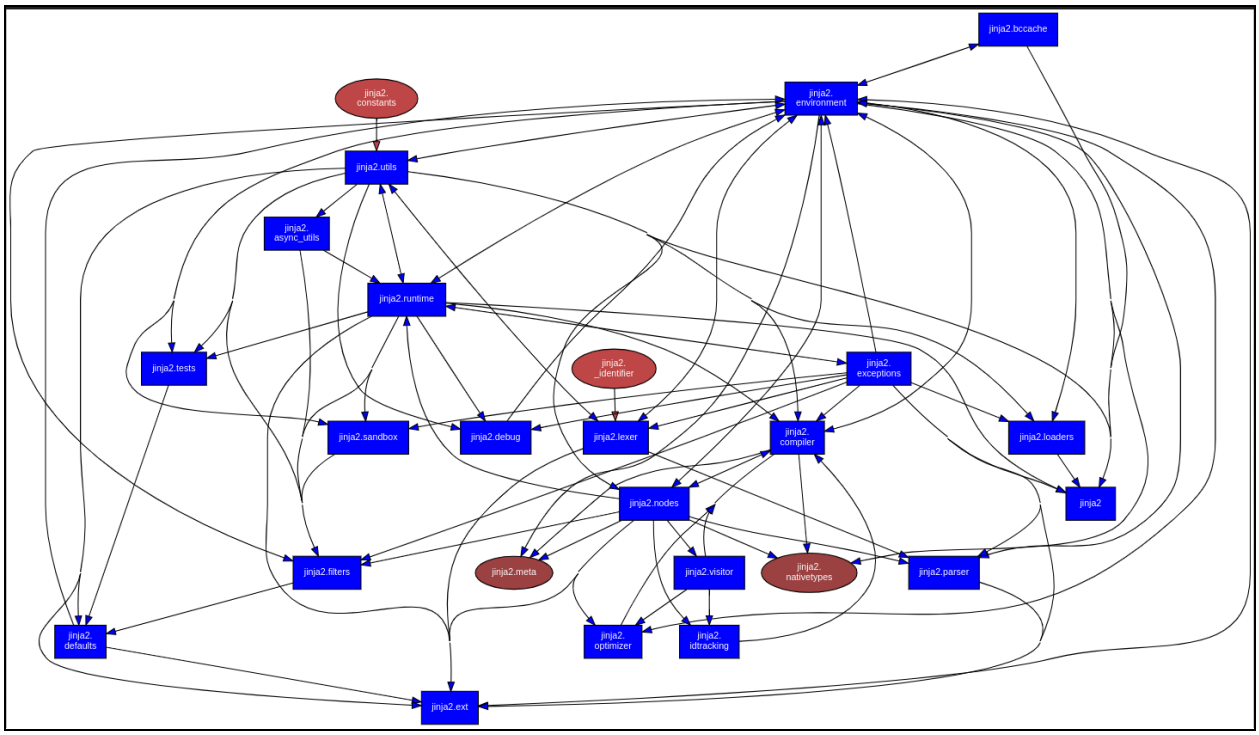
```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ pip install pydeps
Collecting pydeps
  Using cached pydeps-3.0.1-py3-none-any.whl.metadata (22 kB)
Collecting stdlib_list (from pydeps)
  Using cached stdlib_list-0.11.1-py3-none-any.whl.metadata (3.3 kB)
Using cached pydeps-3.0.1-py3-none-any.whl (47 kB)
Using cached stdlib_list-0.11.1-py3-none-any.whl (83 kB)
Installing collected packages: stdlib_list, pydeps
Successfully installed pydeps-3.0.1 stdlib_list-0.11.1
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ pydeps --version
pydeps v3.0.1
```

- To Generate Dependency Graph
    - Focus on the main folder
    - Generate SVG dependency graph using bash script: pydeps jinja2 --noshow --show-dot



```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja$ cd src
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ pydeps jinja2 --output jinja2_deps.svg
usage: pydeps [-h] [--debug] [--config FILE] [--no-config] [--version] [-L LOG] [--find-package] [-v] [-o file] [-T FORMAT] [--display PROGRAM] [--noshow] [--show-deps]
              [--show-raw-deps] [--deps-output DEPS_OUT] [--show-dot] [--dot-output DOT_OUT] [--nodot] [--no-output] [--show-cycles] [--debug-mf INT] [--noise-level INT]
              [--max-bacon INT] [--max-module-depth INT] [--pylib] [--pylib-all] [--include-missing] [-x PATTERN [PATTERN ...]] [-xx MODULE [MODULE ...]]
              [--only MODULE_PATH [MODULE_PATH ...]] [--externals] [--reverse] [--rankdir {TB,BT,LR,RL}] [--cluster] [--min-cluster-size INT] [--max-cluster-size INT]
              [--keep-target-cluster] [--collapse-target-cluster] [--rmprefix PREFIX [PREFIX ...]] [--start-color INT]
              fname
pydeps: error: unrecognized arguments: --output jinja2_deps.svg
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ pydeps jinja2 --noshow --show-dot

digraph G {
    concentrate = true;

    rankdir = TB;
    node [style=filled,fillcolor="#ffffff",fontcolor="#000000",fontname=Helvetica,fontsize=10];

    jinja2 [fillcolor="blue",fontcolor="white",shape="box"];
    jinja2__identifier [fillcolor="#c24747",fontcolor="#ffffff",label="jinja2\.\n_identifier"];
    jinja2_async_utils [fillcolor="blue",fontcolor="white",label="jinja2\.\nasync_utils",shape="box"];
    jinja2_bccache [fillcolor="blue",fontcolor="white",label="jinja2.bccache",shape="box"];
    jinja2_compiler [fillcolor="blue",fontcolor="white",label="jinja2\.\ncompiler",shape="box"];
    jinja2_constants [fillcolor="#c24747",fontcolor="#ffffff",label="jinja2\.\nconstants"];
    jinja2_debug [fillcolor="blue",fontcolor="white",label="jinja2.debug",shape="box"];
    jinja2_defaults [fillcolor="blue",fontcolor="white",label="jinja2\.\ndefaults",shape="box"];
    jinja2_environment [fillcolor="blue",fontcolor="white",label="jinja2\.\nenvironment",shape="box"];
    jinja2_exceptions [fillcolor="blue",fontcolor="white",label="jinja2\.\nexceptions",shape="box"];
    jinja2_ext [fillcolor="blue",fontcolor="white",label="jinja2.ext",shape="box"];
    jinja2_filters [fillcolor="blue",fontcolor="white",label="jinja2.filters",shape="box"];
    jinja2_idtracking [fillcolor="blue",fontcolor="white",label="jinja2\.\nidtracking",shape="box"];
    jinja2_lexer [fillcolor="blue",fontcolor="white",label="jinja2.lexer",shape="box"];
    jinja2_loaders [fillcolor="blue",fontcolor="white",label="jinja2.loaders",shape="box"];
    jinja2_meta [fillcolor="#9d4343",fontcolor="#ffffff",label="jinja2.meta"];
    jinja2_nativetypes [fillcolor="#9d4343",fontcolor="#ffffff",label="jinja2\.\nnativetypes"];
    jinja2_nodes [fillcolor="blue",fontcolor="white",label="jinja2.nodes",shape="box"];
    jinja2_optimizer [fillcolor="blue",fontcolor="white",label="jinja2\.\noptimizer",shape="box"];
    jinja2_parser [fillcolor="blue",fontcolor="white",label="jinja2.parser",shape="box"];
    jinja2_runtime [fillcolor="blue",fontcolor="white",label="jinja2.runtime",shape="box"];
    jinja2_sandbox [fillcolor="blue",fontcolor="white",label="jinja2.sandbox",shape="box"];
    jinja2_tests [fillcolor="blue",fontcolor="white",label="jinja2.tests",shape="box"];
    jinja2_utils [fillcolor="blue",fontcolor="white",label="jinja2.utils",shape="box"];
    jinja2_visitor [fillcolor="blue",fontcolor="white",label="jinja2.visitor",shape="box"];
```

    - This will generate a visual representation of the module dependencies in a file named: ‘jinja2.pydeps.svg’.



- Generate JSON output for Fan-In/Fan-Out
    - Save all dependencies in JSON format using bash script:

```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ pydeps jinja2 --noshow --show-deps --deps-output=jinja2_deps.json
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ cat jinja2_deps.json
{
    "__main__": {
        "bacon": 0,
        "imports": [
            "jinja2",
            "jinja2._identifier",
            "jinja2.async_utils",
            "jinja2.bccache",
            "jinja2.compiler",
            "jinja2.constants",
            "jinja2.debug",
            "jinja2.defaults",
            "jinja2.environment",
            "jinja2.exceptions",
            "jinja2.ext",
            "jinja2.filters",
            "jinja2.idtracking",
            "jinja2.lexer",
            "jinja2.loaders",
            "jinja2.meta",
            "jinja2.nativetypes",
            "jinja2.nodes",
            "jinja2.optimizer",
            "jinja2.parser",
            "jinja2.runtime",
                                                                    Ln 478, Col
```

This will create a json file inside src which contains all the dependencies.

- Create a fan_analysis.py file inside the same folder i.e. src, and write the following script to calculate fan-in and fan-out of each module.

Dependency_depth.py U | dependency_depths.csv U | dependencies_impact_analysis.py U | fan_analysis.py U

jinja > src > fan_analysis.py > ...

```python
import json
import csv
from collections import defaultdict

# Load the JSON dependency file
with open('jinja2_deps.json', 'r') as file:
    data = json.load(file)

# Parse the imports
imports_by_module = {module: set(info.get('imports', [])) for module, info in data.items()}

# Initialize fan-in and fan-out structures
fan_in = defaultdict(set)
fan_out = defaultdict(set)

# Calculate fan-in and fan-out
for module, imports in imports_by_module.items():
    for imported_module in imports:
        if imported_module in data and imported_module != module:
            fan_out[module].add(imported_module)
            fan_in[imported_module].add(module)

# Save detailed report
with open('fan_in_out_report.txt', 'w') as report:
    report.write("Fan-In and Fan-Out per module:\n\n")
    for module in sorted(data):
        report.write(f"Module: {module}\n")
        report.write(f"  Fan-In  = {len(fan_in[module])} (Used by: {sorted(fan_in[module])})\n")
        report.write(f"  Fan-Out = {len(fan_out[module])} (Uses:    {sorted(fan_out[module])})\n\n")

print("✔ Text report saved to 'fan_in_out_report.txt'")

# Save fan analysis to CSV
with open('fan_analysis.csv', 'w', newline='') as csvfile:
    writer = csv.writer(csvfile)
    writer.writerow(['Module', 'Fan-In', 'Fan-Out'])

    for module in sorted(data):
        writer.writerow([module, len(fan_in[module]), len(fan_out[module])])

print("✔ CSV file saved as 'fan_analysis.csv'")
```

This will create a CSV and txt file containing each module's fan-in and fan-out.

```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ python fan_analysis.py
✔ Text report saved to 'fan_in_out_report.txt'
✔ CSV file saved as 'fan_analysis.csv'
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$
```

fan_analysis

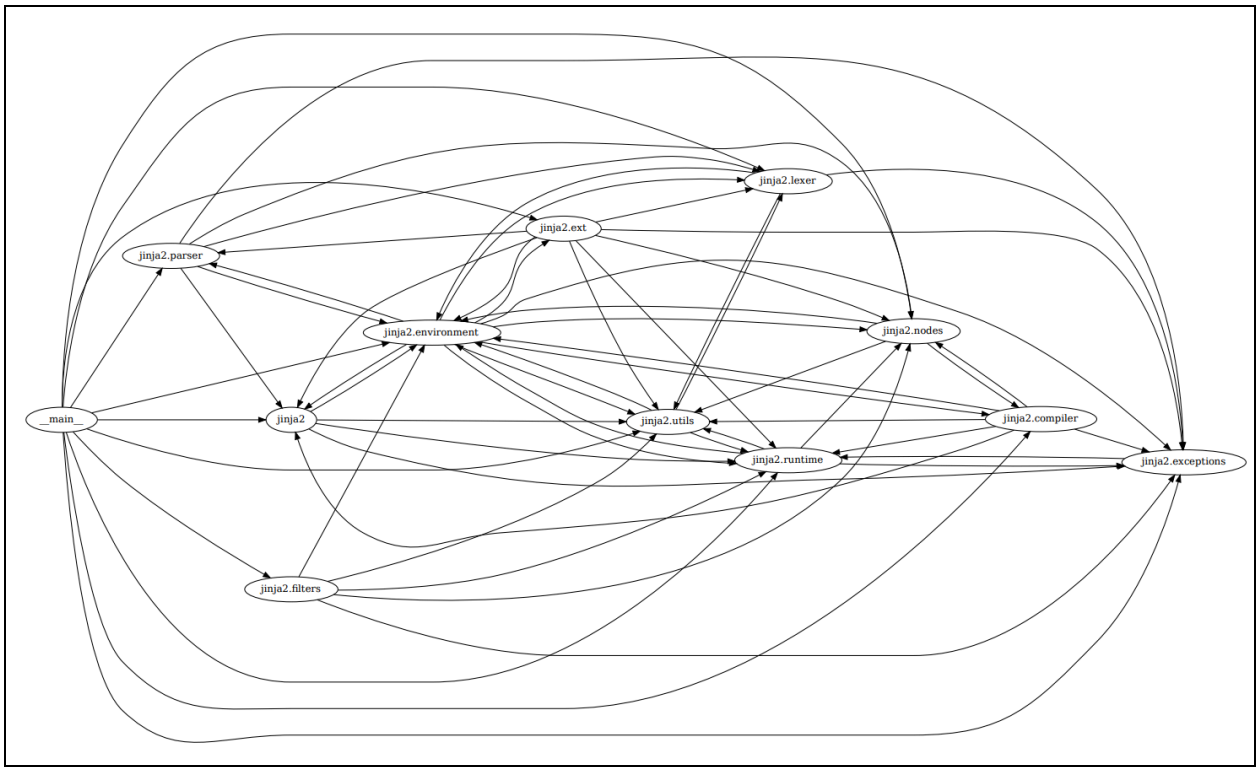
- **Analyze the dependencies graph:**
  Open the generated visual graph to identify the following:
    - Highly coupled modules (high fan-in/fan-out) and their dependencies.
      Write the following code in a .py file:

fan_analysis.py U highly_coupled.py U

jinja > src > highly_coupled.py > ...

```python
import pandas as pd
import json
from graphviz import Digraph

# Load fan analysis
fan_df = pd.read_csv("fan_analysis.csv")
threshold = 5  # Customize this as needed
highly_coupled = fan_df[(fan_df['Fan-In'] >= threshold) | (fan_df['Fan-Out'] >= threshold)]
highly_coupled_modules = set(highly_coupled['Module'])

# Load dependency data
with open("jinja2_deps.json") as f:
    deps = json.load(f)

# Output file to store textual summary
with open("coupled_module_dependencies.txt", "w") as out:
    out.write("Highly Coupled Modules and Their Dependencies:\n\n")

    for module in sorted(highly_coupled_modules):
        # Get dependencies and dependents
        dependencies = deps.get(module, {}).get("imports", [])
        dependents = [mod for mod, info in deps.items() if module in info.get("imports", [])]

        out.write(f"Module: {module}\n")
        out.write(f"  ➤ Depends on: {dependencies if dependencies else 'None'}\n")
        out.write(f"  ➤ Depended on by: {dependents if dependents else 'None'}\n\n")

print("✔ Text output saved to 'coupled_module_dependencies.txt'")

# --- Graphviz Visualization ---
dot = Digraph(comment='Highly Coupled Module Dependencies')
dot.attr(rankdir='LR', size='20,10', dpi='300')  # High-resolution wide graph
dot.format = 'pdf'  # Use PDF for better clarity

# Add nodes and edges (only between highly coupled modules)
for src in highly_coupled_modules:
    dot.node(src)

    for tgt in deps.get(src, {}).get("imports", []):
        if tgt in highly_coupled_modules:
            dot.edge(src, tgt)

# Render to file
dot.render('highly_coupled_modules_graph', view=True)
print("✔ Graphviz graph saved as 'highly_coupled_modules_graph.pdf'")
```

Here I have taken the threshold for finding the highly coupled modules as 5. This code loads fan_analysis.csv and jinja2_deps. This creates a .txt file containing a summary of who depends on whom and a graph file and image.



- Detect cyclic dependencies.
  When module A depends on B, which in turn depends on C,... which in turn depends on A, this is known as a cyclic dependency.
  To detect the cyclic dependencies, write the following code and run

This will detect cycles by performing DFS.

```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ python3 detect_cycle.py
Cycle(s) detected. Details stored in cyclic_dependencies.txt
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$
```

Cycle detected. It stored the details in the cyclic_dependencies.txt file.

- Unused and disconnected modules.
  Unused means modules with fan-in = 0 and not used by any other modules.
  Disconnected means modules that have fan-in = 0 and fan-out = 0 i.e. not used any other modules and not being used by any other modules.
  Write the following script to detect the unused modules:

unused_disconnected_modules.py U

jinja > src > unused_disconnected_modules.py > ...

```python
import pandas as pd

# Load the fan-in/fan-out CSV
df = pd.read_csv("fan_analysis.csv")

# Unused Modules: Fan-In == 0
unused = df[df['Fan-In'] == 0]
print("\nUnused Modules (Fan-In = 0):")
print(unused[['Module', 'Fan-In', 'Fan-Out']])
unused[['Module', 'Fan-In', 'Fan-Out']].to_csv("unused_modules.csv", index=False)

#Disconnected Modules: Fan-In == 0 AND Fan-Out == 0
disconnected = df[(df['Fan-In'] == 0) & (df['Fan-Out'] == 0)]
print("\nDisconnected Modules (Fan-In = 0 and Fan-Out = 0):")
print(disconnected[['Module', 'Fan-In', 'Fan-Out']])
disconnected[['Module', 'Fan-In', 'Fan-Out']].to_csv("disconnected_modules.csv", index=False)
```

This will store the output in the CSV files. There are no disconnected modules, and there are 1 unused module.

- The depth of dependencies.
  Write code in .py file:

Dependency_depth.py U dependency_depths.csv U

jinja > src > Dependency_depth.py > ...

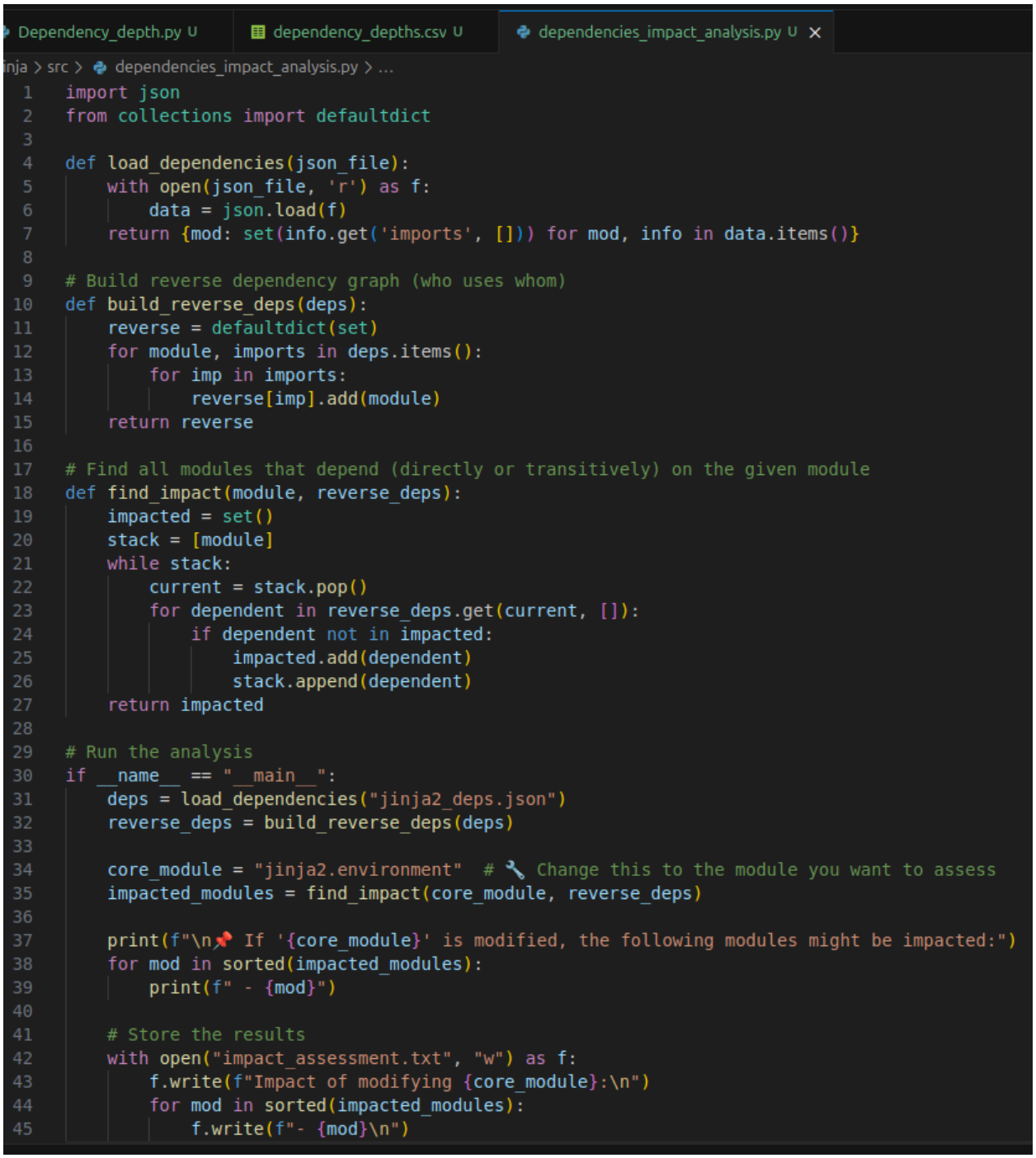
```python
import json
import pandas as pd

def load_dependencies(json_file):
    with open(json_file, 'r') as f:
        data = json.load(f)
    return {mod: set(info.get('imports', [])) for mod, info in data.items()}

def calculate_depths(deps):
    depths = {}

    def dfs(module, visited):
        if module in depths:
            return depths[module]
        if module in visited:
            return 0  # Avoid infinite recursion on cycles
        visited.add(module)
        max_depth = 0
        for dep in deps.get(module, []):
            max_depth = max(max_depth, 1 + dfs(dep, visited))
        visited.remove(module)
        depths[module] = max_depth
        return max_depth

    for module in deps:
        dfs(module, set())

    return depths

# Load and calculate
deps = load_dependencies("jinja2_deps.json")
depths = calculate_depths(deps)

# Save to CSV
depth_df = pd.DataFrame([
    {"Module": mod, "Dependency Depth": depth}
    for mod, depth in depths.items()
])
depth_df.sort_values(by="Dependency Depth", ascending=False, inplace=True)
depth_df.to_csv("dependency_depths.csv", index=False)

print("Dependency depth analysis saved to 'dependency_depths.csv'")
```

Run the script:

```
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ python3 Dependency_depth.py
Dependency depth analysis saved to 'dependency_depths.csv'
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$
```

This generates a CSV file showing the depth of dependencies of each module.
dependency_depths

- **Perform a dependency impact assessment:**
  From the fan_analysis.csv file, we can find that module ‘jinja2.environment’ has fan-in. Therefore, this module has been taken as core module1.
  Write the following script to find the effect of changes in the core module1:

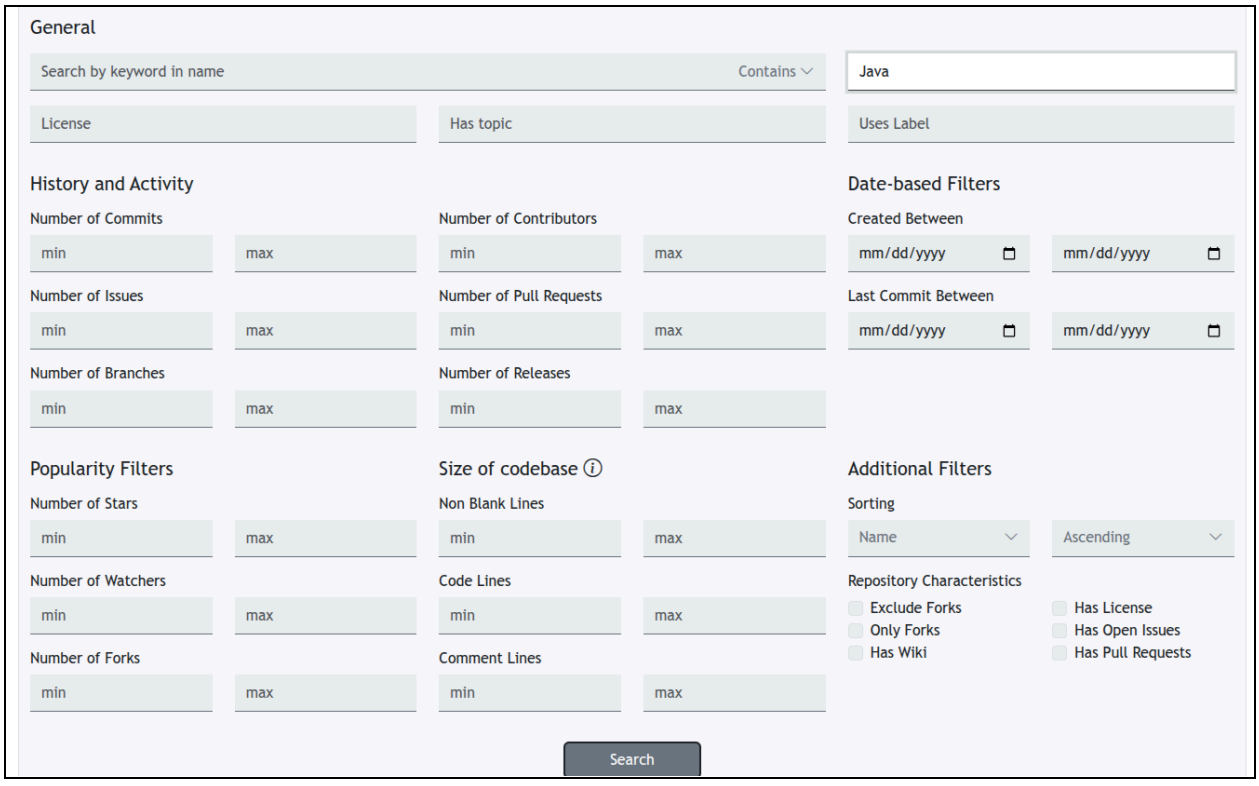


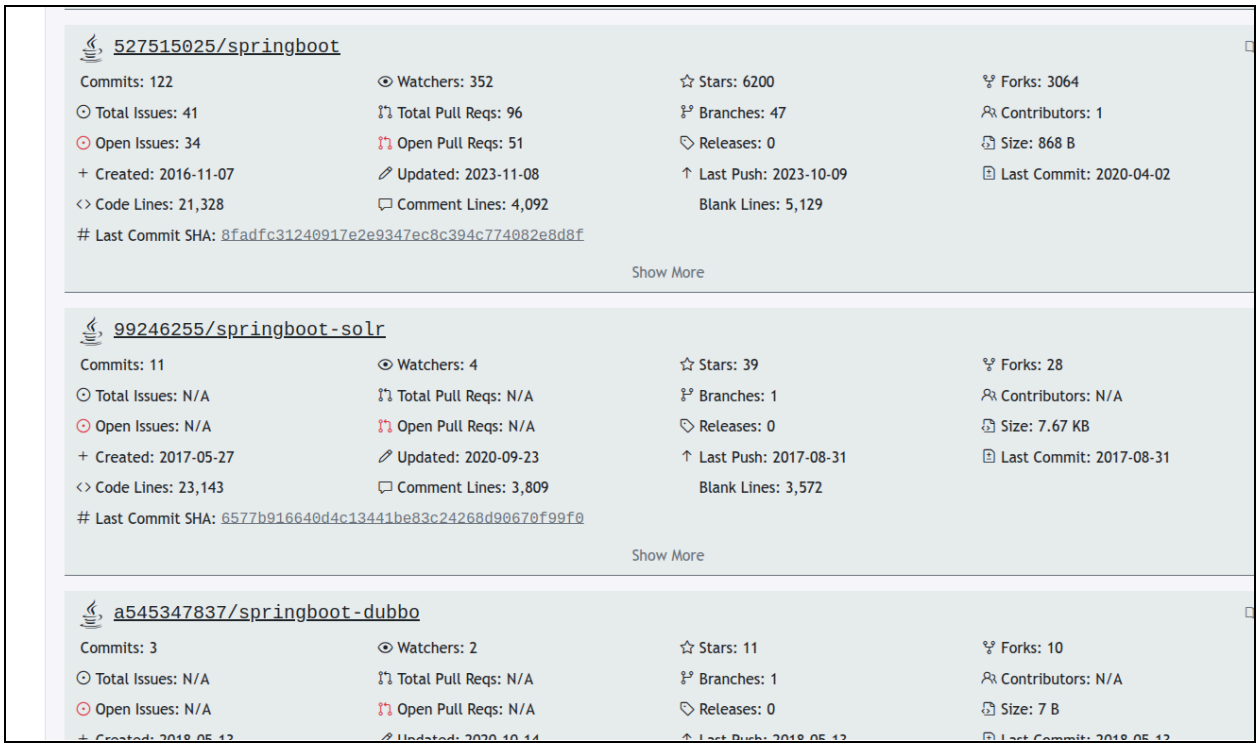
The output is:

```
 - jinja2.visitor
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ python3 dependencies_impact_analysis.py

 If 'jinja2.environment' is modified, the following modules might be impacted:
 - __main__
 - jinja2
 - jinja2.async_utils
 - jinja2.bccache
 - jinja2.compiler
 - jinja2.debug
 - jinja2.defaults
 - jinja2.environment
 - jinja2.exceptions
 - jinja2.ext
 - jinja2.filters
 - jinja2.idtracking
 - jinja2.lexer
 - jinja2.loaders
 - jinja2.meta
 - jinja2.nativetypes
 - jinja2.nodes
 - jinja2.optimizer
 - jinja2.parser
 - jinja2.runtime
 - jinja2.sandbox
 - jinja2.tests
 - jinja2.utils
 - jinja2.visitor
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$
```

5. Measuring Java Class Cohesi2on using LCOM:

- Java project with at least 10 classes.





https://github.com/527515025/springBoot

```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~$ cd Lab9
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ git clone https://github.com/527515025/springBoot.git
Cloning into 'springBoot'...
remote: Enumerating objects: 2105, done.
remote: Counting objects: 100% (76/76), done.
remote: Compressing objects: 100% (57/57), done.
remote: Total 2105 (delta 71), reused 19 (delta 19), pack-reused 2029 (from 2)
Receiving objects: 100% (2105/2105), 865.21 KiB | 5.18 MiB/s, done.
Resolving deltas: 100% (684/684), done.
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9$ cd springBoot
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$ python3 -m venv java_env
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$ source java_env/bin/activate
(java_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$ sudo apt install openjdk-17-jre-headless
Reading package lists... Done
Building dependency tree... Done
Reading state information... Done
openjdk-17-jre-headless is already the newest version (17.0.14+7-1~24.04).
0 upgraded, 0 newly installed, 0 to remove and 40 not upgraded.
(java_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$ java -jar LCOM.jar -i springboot-elasticsearch/src -o lcom_output
Parsing the source code ...
Resolving symbols...
Computing metrics...
Done.
(java_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$
```

```
(java_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$ java --version
openjdk 21.0.6 2025-01-21
OpenJDK Runtime Environment (build 21.0.6+7-Ubuntu-124.04.1)
OpenJDK 64-Bit Server VM (build 21.0.6+7-Ubuntu-124.04.1, mixed mode, sharing)
(java_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/springBoot$
```

This is the output:

Project Name	Package Name	Type Name	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	YALCOM
springboot-elasticsearch	cn.abel	BaseTest	0	0	1	1	0	0
springboot-elasticsearch	cn.abel	ComponentScan	0	0	0	0	0	-1

		Config						
springboot-elasticsearch	cn.abel	Application	0	0	1	1	0	0
springboot-elasticsearch	cn.abel.service	ServiceTest	0	0	1	1	0	0
springboot-elasticsearch	cn.abel.service	UserService	32	19	5	2	0.8888888889	0
springboot-elasticsearch	cn.abel.constants	Constants	0	0	0	0	0	-1
springboot-elasticsearch	cn.abel.config	ESRestClient2Config	1	1	2	2	1	1
springboot-elasticsearch	cn.abel.config	ESRestClientConfig	1	1	2	2	1	1
springboot-elasticsearch	cn.abel.bean	User	144	135	9	9	0.9411764706	0.5
springboot-elasticsearch	cn.abel.dao	UserDao	10	0	5	5	0	-1

TypeMetrics

Class with the highest LCOM value is '**cn.abel.bean**'.

Result and Analysis:

PYTHON:

- Fan-in means how many modules depend on this module, and fan-out means how many modules this module depends on. From the fan_analysis.csv file, we found that module '_main_' depends on 25 other modules, and 17 modules depend on module 59 'jinja2.environment'. So if we make any changes in '_make_', it will not affect other modules. However, if we make any changes in 'jinja2.environment', it will affect all 17 modules that depend on this. Therefore, modules with high fan-in are **risky to change**, as they are widely used across the system. So they need careful handling while refactoring.
- We detected cycles in the selected Python GitHub repository. Cycles connect system components tightly, making testing, debugging, and extending code difficult.
- When two or more modules depend on one another, modifying one without impacting the others becomes challenging, making it harder to maintain the code. This is how cycles affect maintainability. Cycles' implicit initialization requirements make unit testing more difficult. If not handled appropriately, it may result in infinite loops or import issues.
- We found 1 unused modules, and there are no disconnected modules.
    - Unused module: _main_
    - Disconnected module: NAN
- On calculating the depth of dependencies of each module, we found that the module '__main__' has the highest depth of dependencies.
- From dependency impact assessment, the output is:

```
 - jinja2.visitor
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$ python3 dependencies_impact_analysis.py

 If 'jinja2.environment' is modified, the following modules might be impacted:
 - __main__
 - jinja2
 - jinja2.async_utils
 - jinja2.bccache
 - jinja2.compiler
 - jinja2.debug
 - jinja2.defaults
 - jinja2.environment
 - jinja2.exceptions
 - jinja2.ext
 - jinja2.filters
 - jinja2.idtracking
 - jinja2.lexer
 - jinja2.loaders
 - jinja2.meta
 - jinja2.nativetypes
 - jinja2.nodes
 - jinja2.optimizer
 - jinja2.parser
 - jinja2.runtime
 - jinja2.sandbox
 - jinja2.tests
 - jinja2.utils
 - jinja2.visitor
(lab9_env) nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/Lab9/jinja/src$
```

This means our impact analysis revealed that **modifying it could affect 24 modules**, which includes both direct and **transitive dependencies**. Since this core module has a high fan-in, that is, it imports the maximum number of other modules, therefore, modules with high fan-in are **risky to change**, as they are widely used across the system.

- Since 'jinja2.environment' has the highest fan-in, this module risks breaking the system if modified. However, every module with a high fan-in is at risk of breaking the system.

JAVA:

- **What does a high LCOM value suggest about a class's design?**
  A class's methods are not cooperating or sharing common data if the LCOM value is high. This may be against the Single Responsibility Principle and demonstrates poor class design. These classes are frequently more difficult to comprehend, maintain, and assess.

- **Is there a chance for performing functional decomposition?**
  It is possible to divide high LCOM classes into smaller groups with more specialized duties. This method is referred to as functional decomposition. It facilitates clearer, more manageable, and modular code.

- **Visualizing Cohesion in Classes**

Class Name	Reason for Selection
User	Very high LCOM1–5 (144, 135, 9, 9) → poor cohesion, possibly needs refactoring.
UserService	LCOM1 = 32 and LCOM2 = 19 → handling many unrelated tasks.
UserDao	LCOM1 = 10, high LCOM3–4 → maybe doing more than just database

	operations.
Constants	LCOMs all zero → perfect cohesion, acts as a reference class.
ESRestClientConfig	Small but useful class, shows cohesive config logic.
Application	LCOM3–5 = 1 → clean and cohesive main class.

Metric Table:

Class Name	LCOM1	LCOM2	LCOM3	LCOM4	LCOM5	YALCOM
User	144	135	9	9	0.94	0.5
UserService	32	19	5	2	0.89	0
UserDao	10	0	5	5	0	-1
Constants	0	0	0	0	0	-1
ESRestClient Config	1	1	2	2	1	1
Application	0	0	1	1	0	0

- User & UserService have very high LCOM1 & LCOM2:- indicating low cohesion, and possible design flaws. Refactoring recommended.
- UserDao also shows moderate cohesion issues:- methods may not use shared fields.
- Constants & Config classes (like ESRestClientConfig) show high cohesion, as expected.
- Application class is well-structured with low LCOM:- no issues.

Discussion and Conclusions:

- Challenges:
    - In detecting the highly coupled modules:

fan_analysis.py U | detect_cycle.py U | unused_modules.py U | highly_coupled.py 1, U | highly_coupled_modules_g

jinja > src > highly_coupled.py > ...

```python
import pandas as pd
import json
from graphviz import Digraph

# Load fan analysis
fan_df = pd.read_csv("fan_analysis.csv")
threshold = 5
highly_coupled = fan_df[(fan_df['Fan-In'] >= threshold) | (fan_df['Fan-Out'] >= threshold)]
highly_coupled_modules = set(highly_coupled['Module'])

# Load dependencies JSON
with open("jinja2_deps.json") as f:
    deps = json.load(f)

# Output file to store text summary
with open("coupled_module_dependencies.txt", "w") as out:
    out.write("🔥 Highly Coupled Modules and Their Dependencies:\n\n")

    for module in sorted(highly_coupled_modules):
        dependencies = deps.get(module, [])
        dependents = [mod for mod, targets in deps.items() if module in targets]

        out.write(f"📁 Module: {module}\n")
        out.write(f"   ➤ Depends on: {dependencies if dependencies else 'None'}\n")
        out.write(f"   ➤ Depended on by: {dependents if dependents else 'None'}\n\n")

print("✅ Text output saved to coupled_module_dependencies.txt")

# --- Graphviz visualization ---
dot = Digraph(comment='Highly Coupled Module Dependencies', format='png')
dot.attr(rankdir='LR', size='10,5')

# Add nodes and edges for highly coupled modules only
for src in highly_coupled_modules:
    dot.node(src)

    for tgt in deps.get(src, []):
        if tgt in highly_coupled_modules:  # Only draw links to other highly coupled modules
            dot.edge(src, tgt)

# Save and render the graph
dot.render('highly_coupled_modules_graph', view=False)
print("✅ Graphviz graph saved as highly_coupled_modules_graph.png")
```

This makes the graph blurred. This is because the graph is too complex, and Graphviz squeezes everything into a small PNG. So to get a clear graph, we can store the graph in PDF instead of PNG because PDF handles scaling better.

- The created dependency graph, which showed too many overlapping edges and nodes, seemed complex and difficult to understand when identifying highly connected modules in the Python project. Graphviz reduced the output into a tiny PNG, rendering it unreadable.
- It was necessary to carefully navigate complex graphs in order to find cyclic dependencies, and a thorough comprehension of inter-module relationships was necessary to comprehend the effects of such cycles.

    - Understanding the various LCOM metric versions (LCOM1–5, YALCOM) and connecting them to actual design problems for Java takes some time and codebase cross-referencing.

- Reflections:
  Pydeps analysis of Python projects helped us identify cycles, coupling, and the effects of modifications by providing information about module-level dependencies. The cohesiveness of each Java class was examined using LCOM measures, which also highlighted design problems and areas for deconstruction. Visual tools and measurements in both cases made structural concerns more evident and easier to address.

- Lessons learned:
  This lab teaches us that Pydeps and LCOM are useful, real-world techniques for assessing software cohesion and modularity. Working with dependency graphs, we found that storing the result in PDF rather than PNG significantly enhances readability due to better scaling, particularly when working with complex module interconnections. Finding crucial modules that are dangerous to change because so many other modules rely on them was made easier by analyzing fan-in and fan-out. Classes with low internal cohesiveness were identified by high LCOM values in the Java project, suggesting possible design faults. We came to the conclusion that functional decomposition could help these classes by making the software more modular, readable, and maintainable over time.

- Summary:
  This lab helped identify code smells and design flaws in real-world repositories by applying theoretical metrics like fan-in/fan-out and LCOM. Using LCOM.jar for Java and pydeps for Python, we could identify low-cohesion and strongly related modules/classes. We were directed to possible reworking areas by visual tools and metric analysis, emphasizing the significance of a modular and maintainable software design.

LAB10:- Development of C# Console Applications

Overview:

This lab uses Visual Studio's.NET framework to explain the fundamentals of C# programming. Through practical experience with console applications, students learn about input/output, control structures, functions, OOP principles, and handling exceptions. It also covers using Visual Studio Debugger practically to comprehend program flow.

Objectives:

The main goal of this lab is to familiarize students with C# programming using Visual Studio's.NET framework. Helping students with setting up development environments, writing and running simple C# console applications, and comprehending fundamental programming ideas like loops, conditionals, and functions are the goals. Using classes, inheritance, and methods also highlights object-oriented programming. Students will also learn how to use Visual Studio's debugging tools, such as breakpoints and step operations, to trace and comprehend program execution and handle exceptions using try-catch blocks.

Environmental Setup:

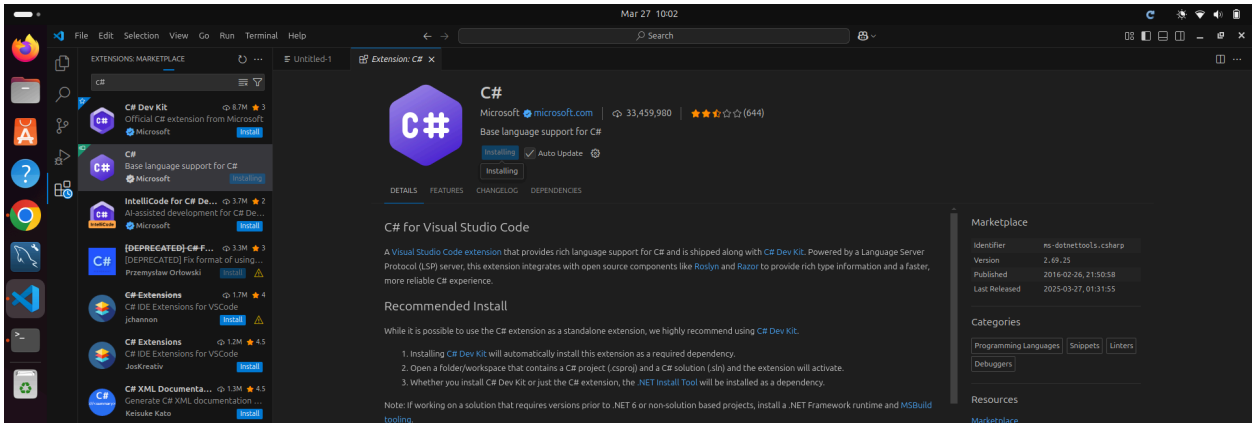
- Operating System: Linux
- Programming language: c#

Tools and versions used:

- Software: Visual Studio 2022 (Community Edition)
- Visual Studio with .NET SDK
- Dotnet version: 8.0.114

Methodology and Execution:

1) Installing C# extension by Microsoft.

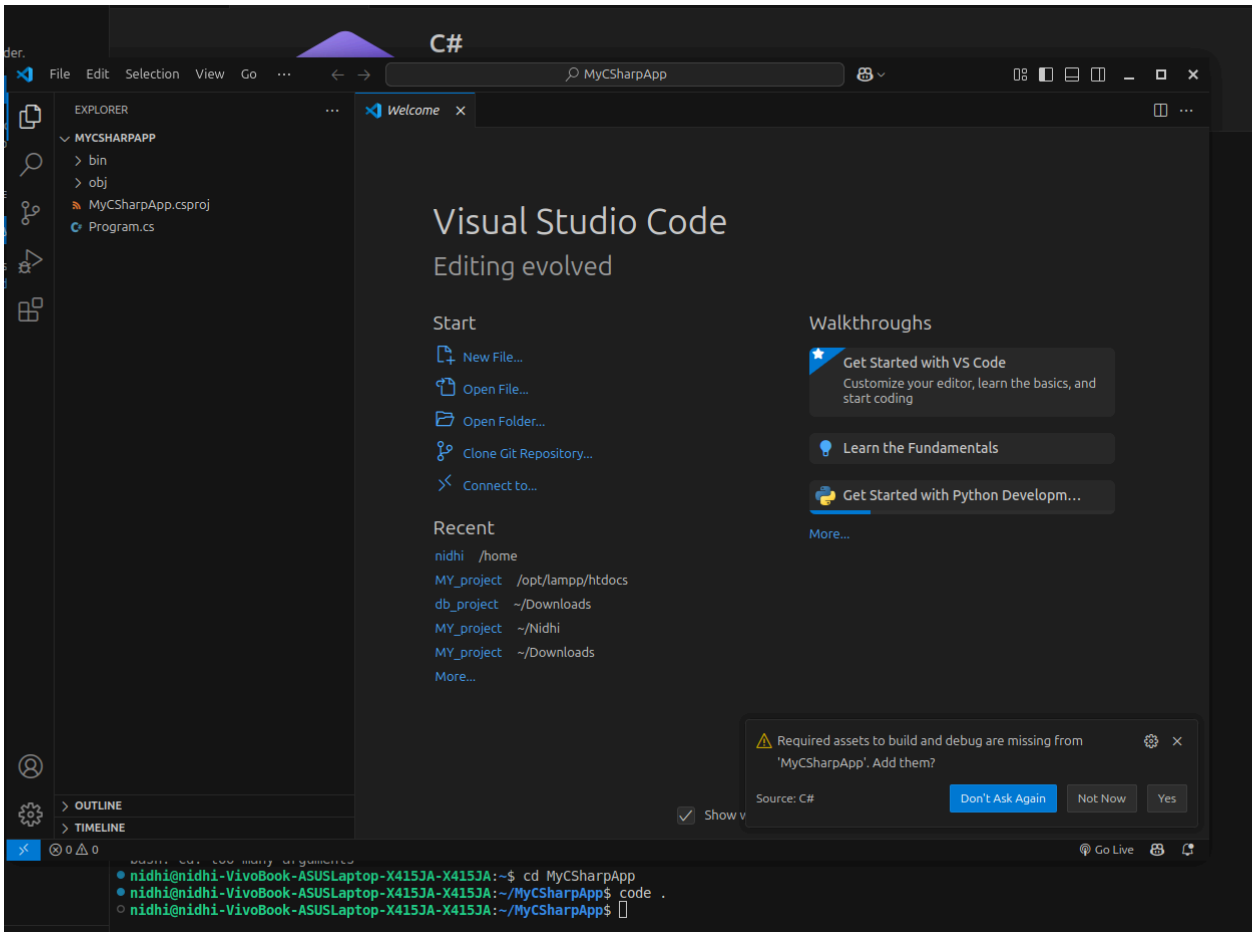


Creates a new C# console application inside the MyCSharpApp folder

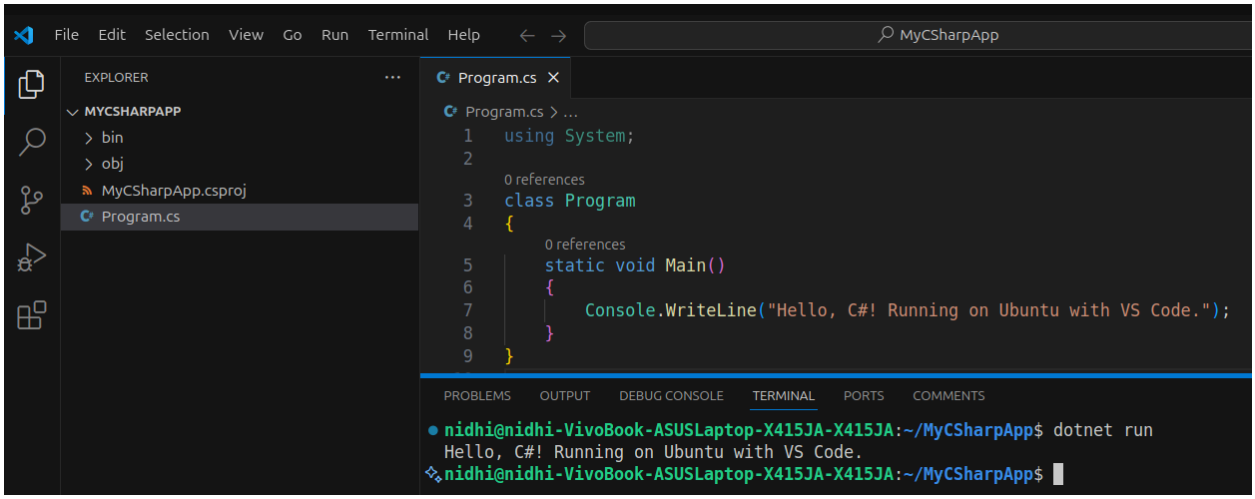
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  COMMENTS

nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~$ mkdir MyCSharpApp && cd MyCSharpApp
dotnet new console
mkdir: cannot create directory ‘MyCSharpApp’: File exists
The template "Console App" was created successfully.

Processing post-creation actions...
Restoring /home/nidhi/nidhi.csproj:
  Determining projects to restore...
^[  Restored /home/nidhi/nidhi.csproj (in 2.89 sec).
Restore succeeded.


nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~$ dotnet new console
```

Go to MyCSharpApp and then run “code .”
This opens the entire MyCSharpApp folder, including MyCSharpApp.csproj, Program.cs, and other files.

```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$ dotnet --version
8.0.114
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$
```



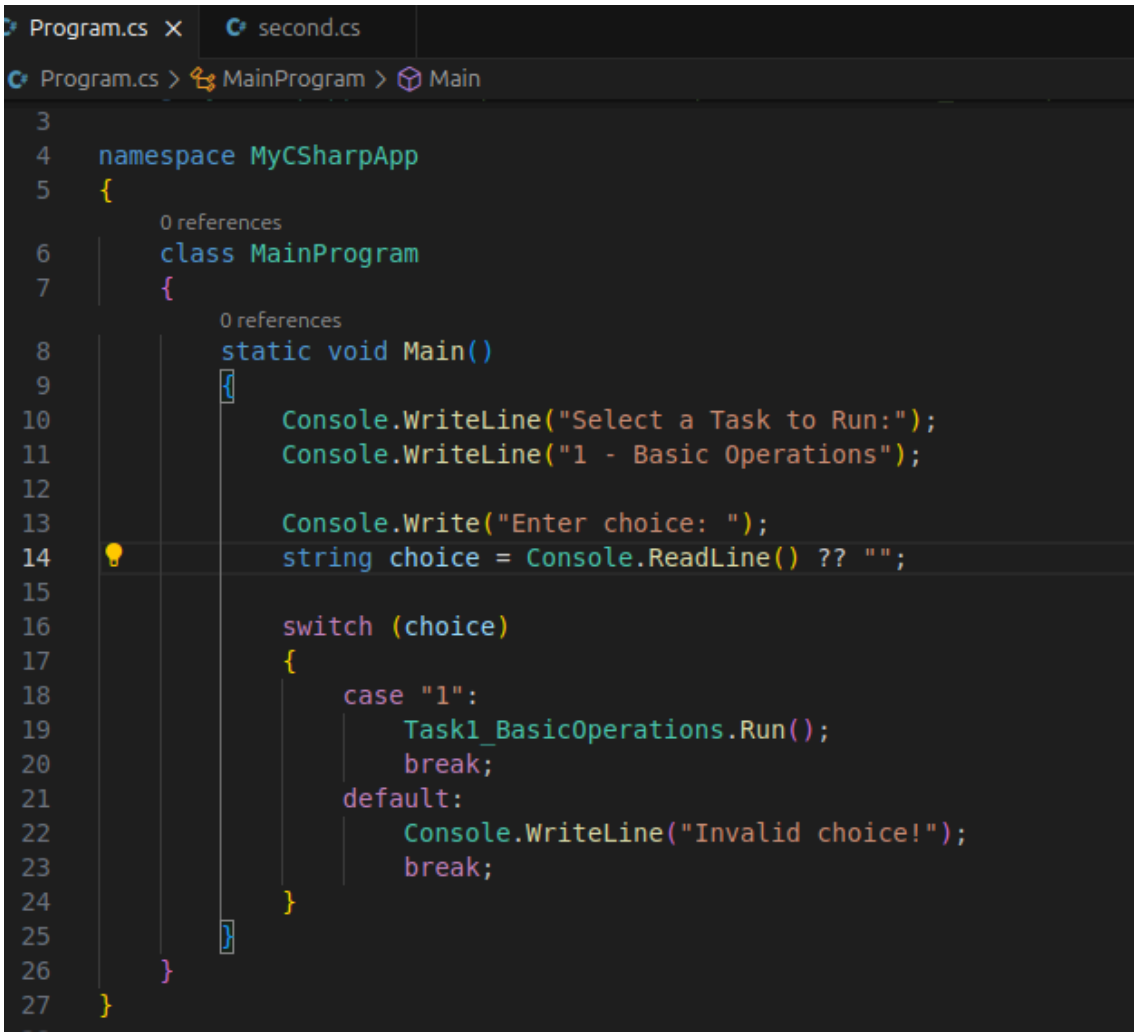
Replace the program.cs content to the below bash script:



Then run the command: ‘dotnet run’ in the terminal.

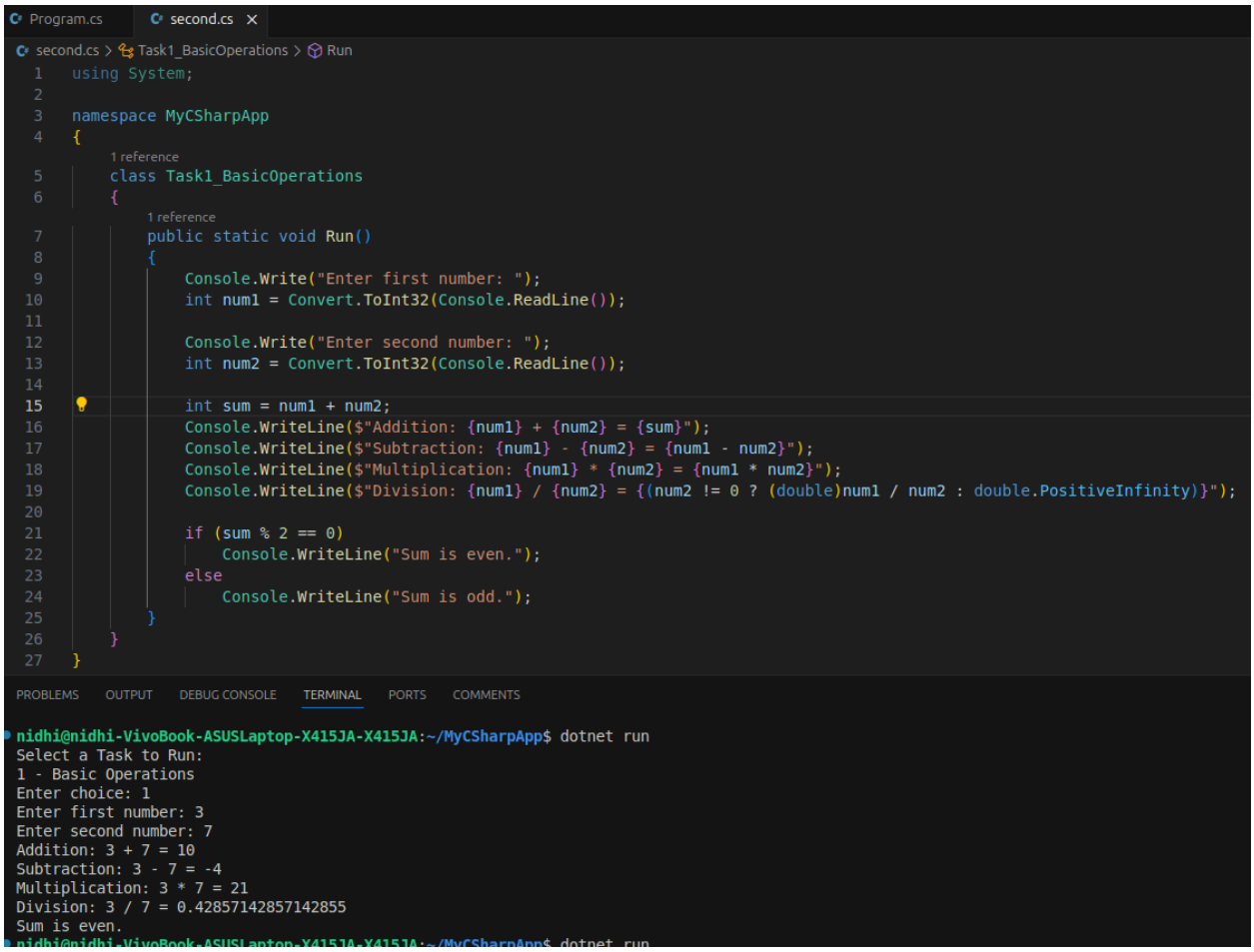
2)

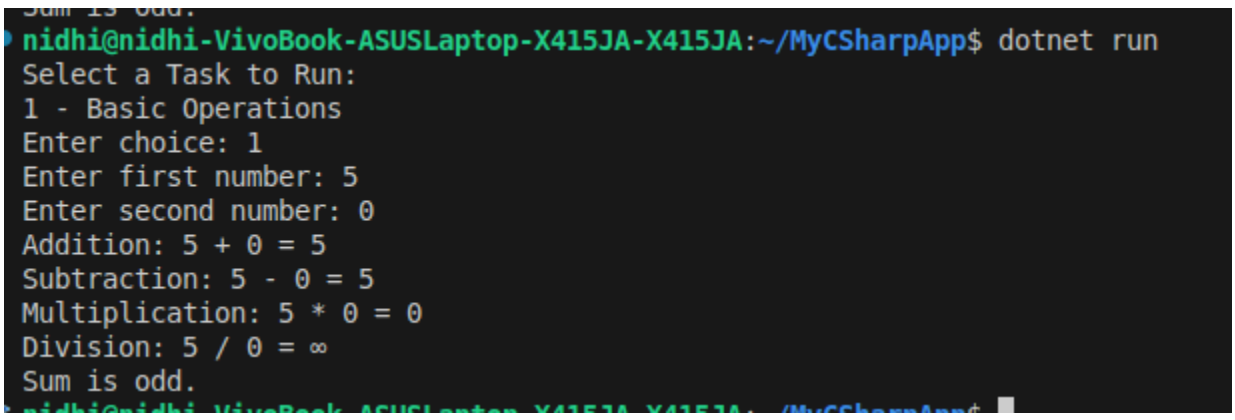
Replace the program.cs script by below script:



This will allow program.cs to select the file dynamically.

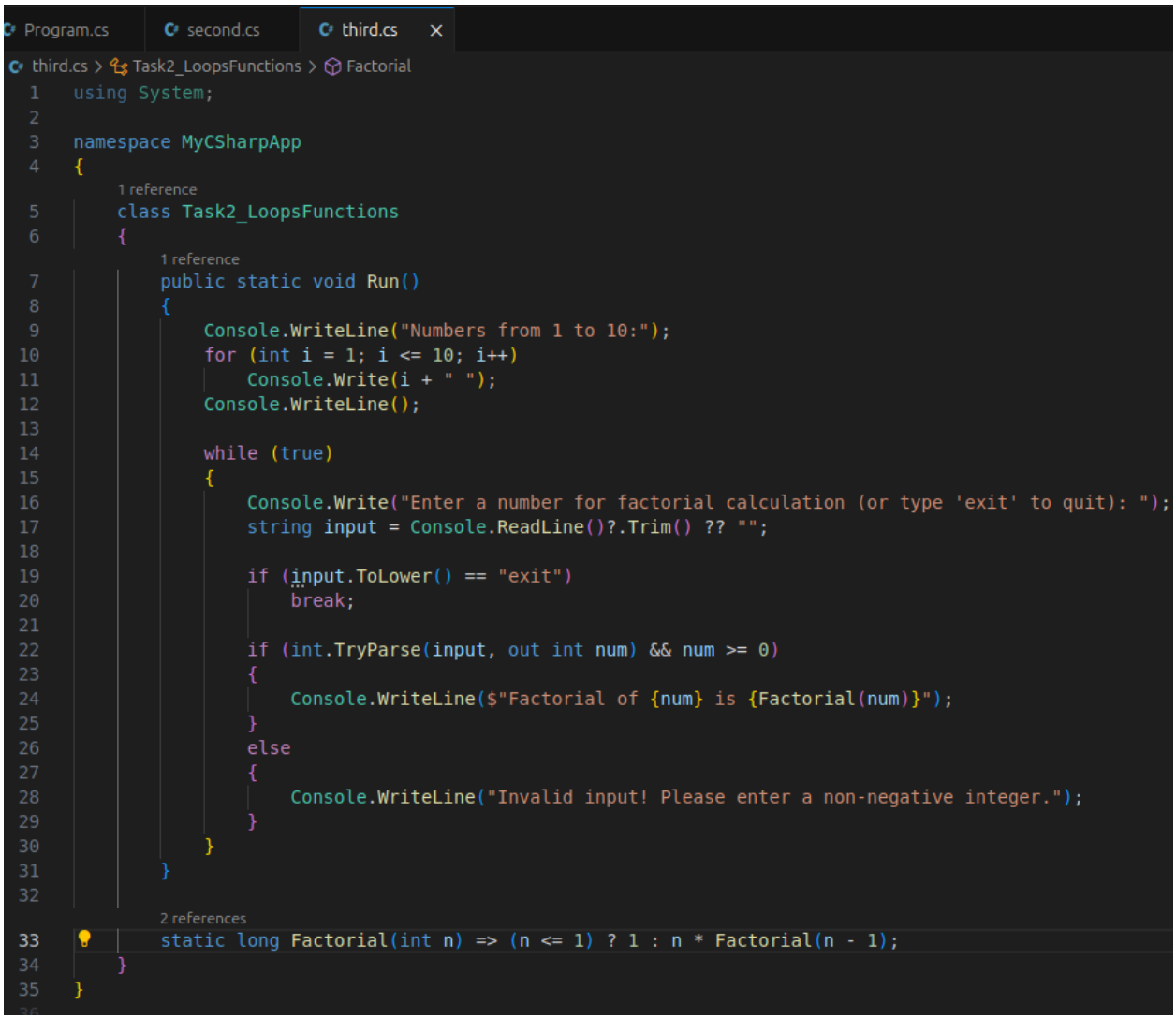
Create a file named: second.cs and write a script to accept user input1 for two numbers, perform addition, subtraction, multiplication, and division, use if-else conditions to determine if the sum is even or odd display the results using Console.WriteLine().





3)

Create a file name: third.cs, and write a script for printing numbers from 1 to 10. Asking the user for input until the user enters "exit", only accept non-negative integers for factorial, and use recursion to compute the factorial:



Update the program.cs content with the below bash script:

```
Program.cs  second.cs  third.cs 2
Program.cs > MainProgram > Main
1   using System;
2   using MyCSharpApp;  // Import the namespace where Task1_BasicOperations is define
3
4   namespace MyCSharpApp
5   {
        0 references
6       class MainProgram
7       {
            0 references
8           static void Main()
9           {
10              Console.WriteLine("Select a Task to Run:");
11              Console.WriteLine("1 - Basic Operations");
12              Console.WriteLine("2 - Loops Functions");
13
14              Console.Write("Enter choice: ");
15              string choice = Console.ReadLine() ?? "";
16
17              switch (choice)
18              {
19                  case "1":
20                      Task1_BasicOperations.Run();
21                      break;
22                  case "2":
23                      Task2_LoopsFunctions.Run();
24                      break;
25                  default:
26                      Console.WriteLine("Invalid choice!");
27                      break;
28              }
29          }
30      }
31  }
```

So that it can take multiple inputs.

```
A: command not found
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$ dotnet run
Select a Task to Run:
1 - Basic Operations
2 - Loops Functions
Enter choice: 2
Numbers from 1 to 10:
1 2 3 4 5 6 7 8 9 10
Enter a number for factorial calculation (or type 'exit' to quit): 3
Factorial of 3 is 6
Enter a number for factorial calculation (or type 'exit' to quit): 4
Factorial of 4 is 24
Enter a number for factorial calculation (or type 'exit' to quit): 1
Factorial of 1 is 1
Enter a number for factorial calculation (or type 'exit' to quit): -1
Invalid input! Please enter a non-negative integer.
Enter a number for factorial calculation (or type 'exit' to quit): exit
```

4)

Create a file named: fourth.cs and write the following script:

gram.cs fourth.cs ●

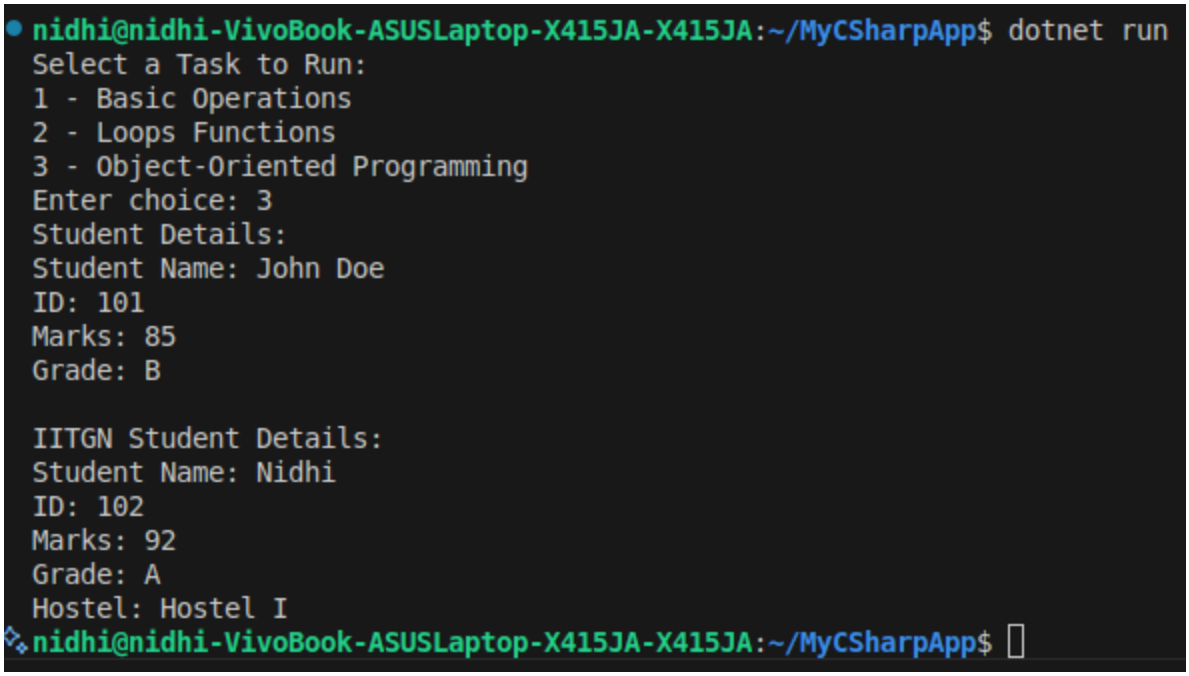
urth.cs > Student > DisplayDetails

```
using System;

namespace MyCSharpApp
{
    1 reference
    class Task3_StudentOOP
    {
        1 reference
        public static void Run()
        {
            Console.WriteLine("Student Details:");
            Student student1 = new Student("John Doe", 101, 85);
            student1.DisplayDetails();

            Console.WriteLine("\nIITGN Student Details:");
            StudentIITGN student2 = new StudentIITGN("Nidhi", 102, 92, "Hostel I");
            student2.DisplayDetails();
        }
    }
    5 references
    class Student
    {
        2 references
        public string Name { get; set; }
        2 references
        public int ID { get; set; }
        6 references
        public double Marks { get; set; }
        2 references
        public Student(string name, int id, double marks)
        {
            Name = name;
            ID = id;
            Marks = marks;
        }
        1 reference
        public string GetGrade()
        {
            if (Marks >= 90) return "A";
            if (Marks >= 80) return "B";
            if (Marks >= 70) return "C";
            if (Marks >= 60) return "D";
            return "F";
        }
        4 references
```

```
        4 references
        public virtual void DisplayDetails()
        {
            Console.WriteLine($"Student Name: {Name}");
            Console.WriteLine($"ID: {ID}");
            Console.WriteLine($"Marks: {Marks}");
            Console.WriteLine($"Grade: {GetGrade()}");
        }
    }

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

        1 reference
        public StudentIITGN(string name, int id, double marks, string hostel)
            : base(name, id, marks)
        {
            Hostel_Name_IITGN = hostel;
        }

        4 references
        public override void DisplayDetails()
        {
            base.DisplayDetails();
            Console.WriteLine($"Hostel: {Hostel_Name_IITGN}");
        }
```

Update the program.cs:

```
            Console.WriteLine("2 - Loops Functions");
            Console.WriteLine("3 - Object-Oriented Programming");

            Console.Write("Enter choice: ");
            string choice = Console.ReadLine() ?? "";

            switch (choice)
            {
                case "1":
                    Task1_BasicOperations.Run();
                    break;
                case "2":
                    Task2_LoopsFunctions.Run();
                    break;
                case "3":
                    Task3_StudentOOP.Run();
                    break;
                default:
                    Console.WriteLine("Invalid choice!");
                    break;
```



5)

- Activity_2: Create a file named: second_update.cs, and write the following script to wrap input conversion in a ‘try-catch’ block to handle non-integer inputs, and wrap division operation in a ‘try-catch’ block to handle division-by-zero exceptions

Program.cs | second.cs | second_update.cs | fourth.cs

second_update.cs > Task1_BasicOperations_update > Run

```cs
using System;

namespace MyCSharpApp
{
    1 reference
    class Task1_BasicOperations_update
    {
        1 reference
        public static void Run()
        {
            int num1, num2;

            // Handle invalid input for num1
            while (true)
            {
                Console.Write("Enter first number: ");
                string input1 = Console.ReadLine() ?? ""; // Prevents null warning
                if (int.TryParse(input1, out num1))
                    break;
                Console.WriteLine("Invalid input! Please enter an integer.");
            }

            // Handle invalid input for num2
            while (true)
            {
                Console.Write("Enter second number: ");
                string input2 = Console.ReadLine() ?? ""; // Prevents null warning
                if (int.TryParse(input2, out num2))
                    break;
                Console.WriteLine("Invalid input! Please enter an integer.");
            }

            int sum = num1 + num2;
            Console.WriteLine($"Addition: {num1} + {num2} = {sum}");
            Console.WriteLine($"Subtraction: {num1} - {num2} = {num1 - num2}");
            Console.WriteLine($"Multiplication: {num1} * {num2} = {num1 * num2}");
```

```
            // Handle division safely
            if (num2 != 0)
            {
                double result = (double)num1 / num2;
                Console.WriteLine($"Division: {num1} / {num2} = {result}");
            }
            else
            {
                Console.WriteLine("Error: Division by zero is not allowed.");
            }

            Console.WriteLine(sum % 2 == 0 ? "Sum is even." : "Sum is odd.");
        }
    }
}
```

Here when we enter invalid input, then the program will prompt again instead of crashing

```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$ dotnet run
Select a Task to Run:
1 - Basic Operations
2 - Loops Functions
3 - Object-Oriented Programming
4 - Task1_BasicOperations_update
Enter choice: 4
Enter first number: r
Invalid input! Please enter an integer.
Enter first number: 1
Enter second number: d
Invalid input! Please enter an integer.
Enter second number: 5
Addition: 1 + 5 = 6
Subtraction: 1 - 5 = -4
Multiplication: 1 * 5 = 5
Division: 1 / 5 = 0.2
Sum is even.
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$
```

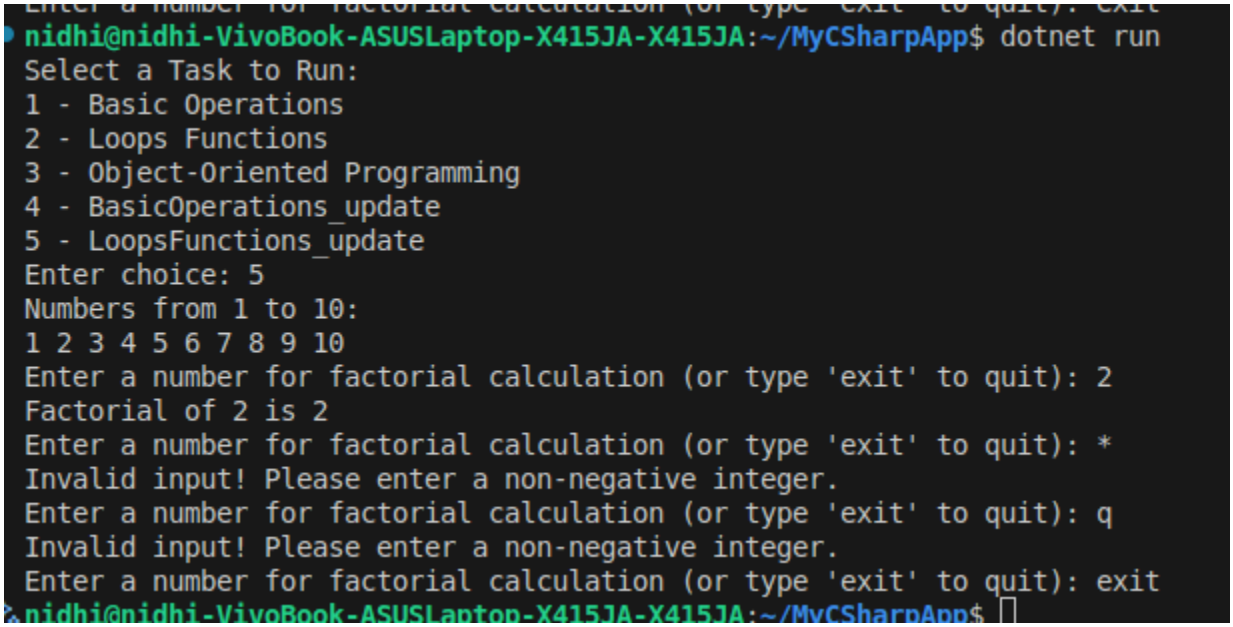
- Activity_3: Create a file named: third_update.cs, and write the following script. This includes the **try-catch block** around user input to handle errors, **try-catch block** in the factorial function to catch any stack overflow issues, and it will prevent crashes on invalid inputs (non-numeric, negative, etc.).

Program.cs | second_update.cs | third.cs | third_update.cs | fourth.cs

third_update.cs > ...

```
using System;

namespace MyCSharpApp
{
    1 reference
    class LoopsFunctions_update
    {
        1 reference
        public static void Run()
        {
            Console.WriteLine("Numbers from 1 to 10:");
            for (int i = 1; i <= 10; i++)
                Console.Write(i + " ");
            Console.WriteLine();

            while (true)
            {
                Console.Write("Enter a number for factorial calculation (or type 'exit' to quit): ");
                string input = Console.ReadLine()?.Trim() ?? "";

                if (input.ToLower() == "exit")
                    break;

                try
                {
                    if (int.TryParse(input, out int num) && num >= 0)
                    {
                        Console.WriteLine($"Factorial of {num} is {Factorial(num)}");
                    }
                    else
                    {
                        throw new FormatException("Invalid input! Please enter a non-negative integer.");
                    }
                }
                catch (FormatException ex)
                {
                    Console.WriteLine(ex.Message);
                }
                catch (OverflowException)
                {
                    Console.WriteLine("Error: The number is too large to calculate its factorial.");
                }
                catch (Exception ex)
                {
                    Console.WriteLine($"Unexpected error: {ex.Message}");
                }
            }
        }

        2 references
        static long Factorial(int n)
        {
            try
            {
                return (n <= 1) ? 1 : checked(n * Factorial(n - 1)); // checked() ensures overflow is caught
            }
            catch (OverflowException)
            {
                throw new OverflowException("Factorial calculation resulted in an overflow.");
            }
        }
    }
}
```

Here when we enter invalid input, then the program will prompt again instead of crashing



- Activity_4: Create a file named: fourth_update.cs, and write the following script. This includes the **try-catch block** around user input to handle errors, **try-catch block** in the factorial function to catch any stack overflow issues., and it will prevents crashes on invalid inputs (non-numeric, negative, etc.).

```
using System;

namespace MyCSharpApp.updated
{
    class StudentOOP_update
    {
        public static void Run()
        {
            try
            {
                Console.WriteLine("Student Details:");
                Student student1 = new Student("John Doe", 101, 85);
                student1.DisplayDetails();

                Console.WriteLine("\nIITGN Student Details:");
                StudentIITGN student2 = new StudentIITGN("Nidhi", 102, 92, "Hostel I");
                student2.DisplayDetails();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Unexpected error: {ex.Message}");
            }
        }
    }

    class Student
    {
        public string Name { get; set; }
        public int ID { get; set; }
        public double Marks { get; set; }

        public Student(string name, int id, double marks)
        {
            try
            {
                if (id <= 0)
                    throw new ArgumentException("ID must be a positive integer.");
                if (marks < 0 || marks > 100)
```

```
                throw new ArgumentOutOfRangeException(nameof(marks), "Marks must be between 0 and 100.");

            Name = name;
            ID = id;
            Marks = marks;
        }
        catch (ArgumentException ex)
        {
            Console.WriteLine($"Error: {ex.Message}");
            Name = "Unknown"; // Assign default name
            ID = 0;
            Marks = 0;
        }
    }

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

    public virtual void DisplayDetails()
    {
        Console.WriteLine($"Student Name: {Name}");
        Console.WriteLine($"ID: {ID}");
        Console.WriteLine($"Marks: {Marks}");
        Console.WriteLine($"Grade: {GetGrade()}");
    }
}

class StudentIITGN : Student
{
    public string Hostel_Name_IITGN { get; set; }
```

```
        public StudentIITGN(string name, int id, double marks, string hostel)
            : base(name, id, marks)
        {
            if (string.IsNullOrWhiteSpace(hostel))
            {
                Console.WriteLine("Error: Hostel name cannot be empty.");
                Hostel_Name_IITGN = "Unknown"; // Assign a default value
            }
            else
            {
                Hostel_Name_IITGN = hostel;
            }
        }

        public override void DisplayDetails()
        {
            base.DisplayDetails();
            Console.WriteLine($"Hostel: {Hostel_Name_IITGN}");
        }
    }
}
```

Also, update the program.cs:

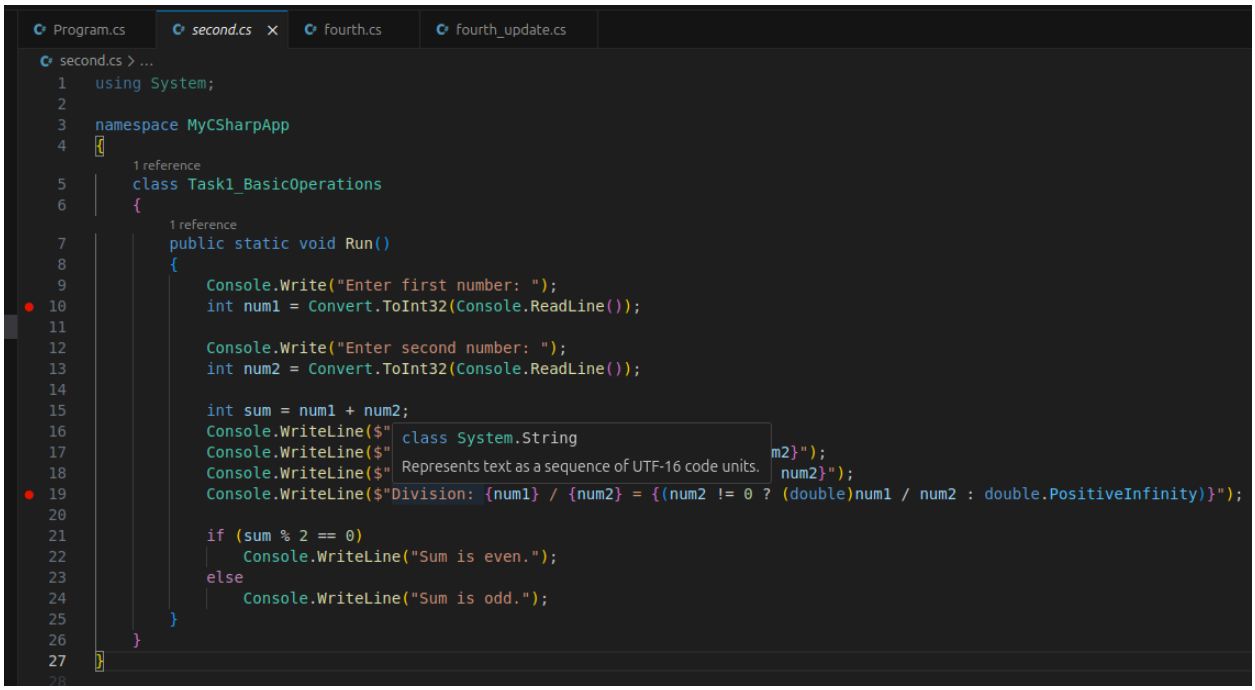
Program.cs

```
using System;
using MyCSharpApp;
using MyCSharpApp.updated;

namespace MyCSharpApp
{
    class MainProgram
    {
        static void Main()
        {
            Console.WriteLine("Select a Task to Run:");
            Console.WriteLine("1 - Basic Operations");
            Console.WriteLine("2 - Loops Functions");
            Console.WriteLine("3 - Object-Oriented Programming");
            Console.WriteLine("4 - BasicOperations_update");
            Console.WriteLine("5 - LoopsFunctions_update");
            Console.WriteLine("6 - StudentOOP_update");


            Console.Write("Enter choice: ");
            string choice = Console.ReadLine() ?? "";

            switch (choice)
            {
                case "1":
                    Task1_BasicOperations.Run();
                    break;
                case "2":
                    Task2_LoopsFunctions.Run();
                    break;
                case "3":
                    Task3_StudentOOP.Run();
                    break;
                case "4":
                    BasicOperations_update.Run();
                    break;
                case "5":
                    LoopsFunctions_update.Run();
                    break;
                case "6":
                    StudentOOP_update.Run();
                    break;
                default:
                    Console.WriteLine("Invalid choice!");
                    break;
            }
        }
    }
}
```

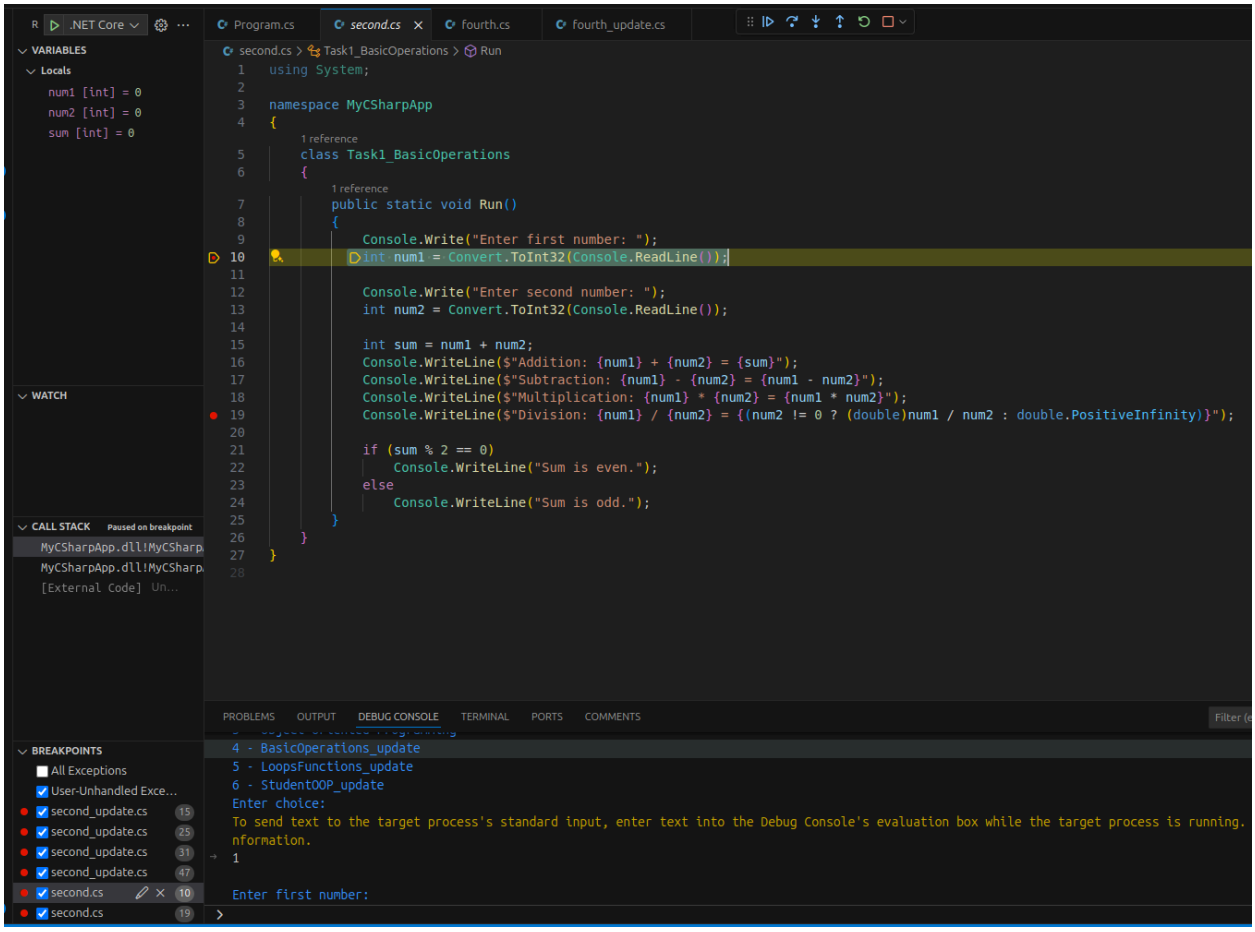
This block will only throw an error if something goes wrong inside the Student or StudentIITGN constructor or DisplayDetails() method that their internal try-catch blocks haven't already caught.

6) Debugging using Visual Studio Debugger

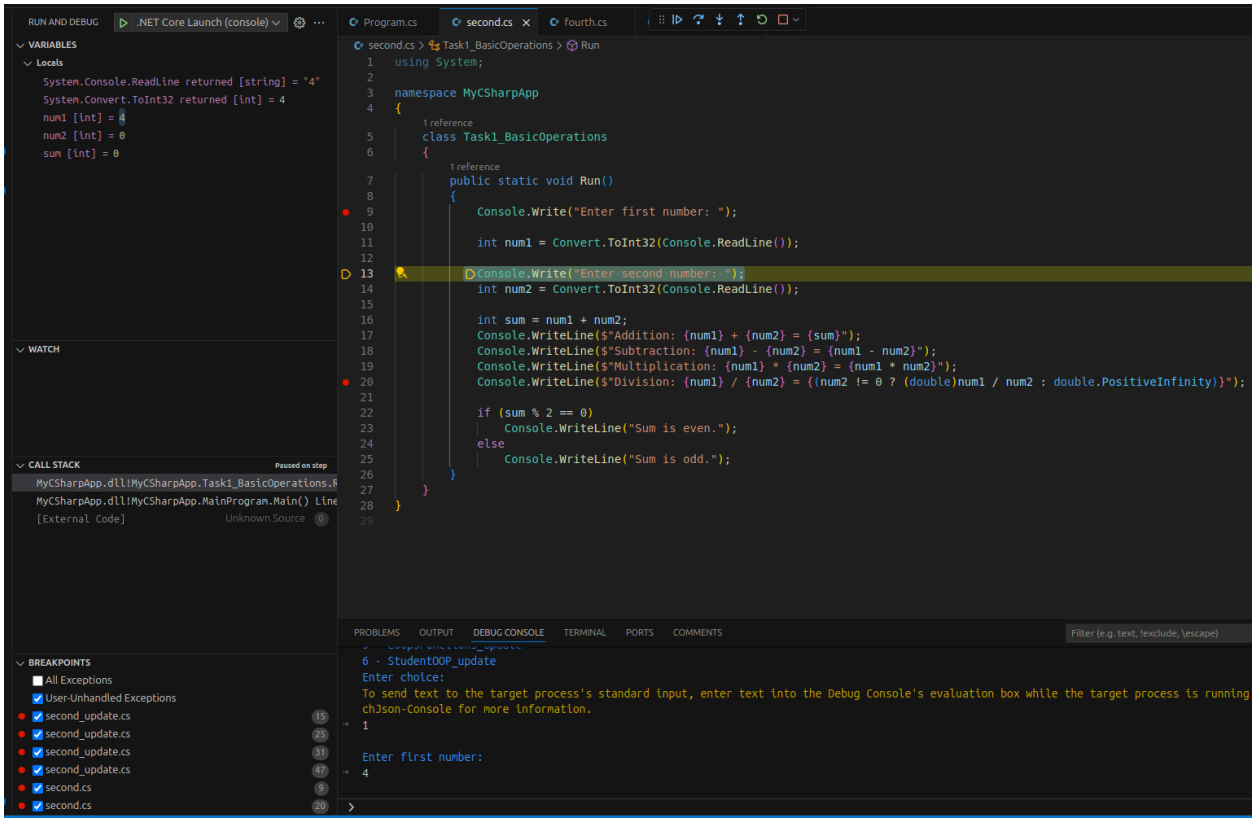
- **Activity_2:**
  - In this code, using two breakpoints:



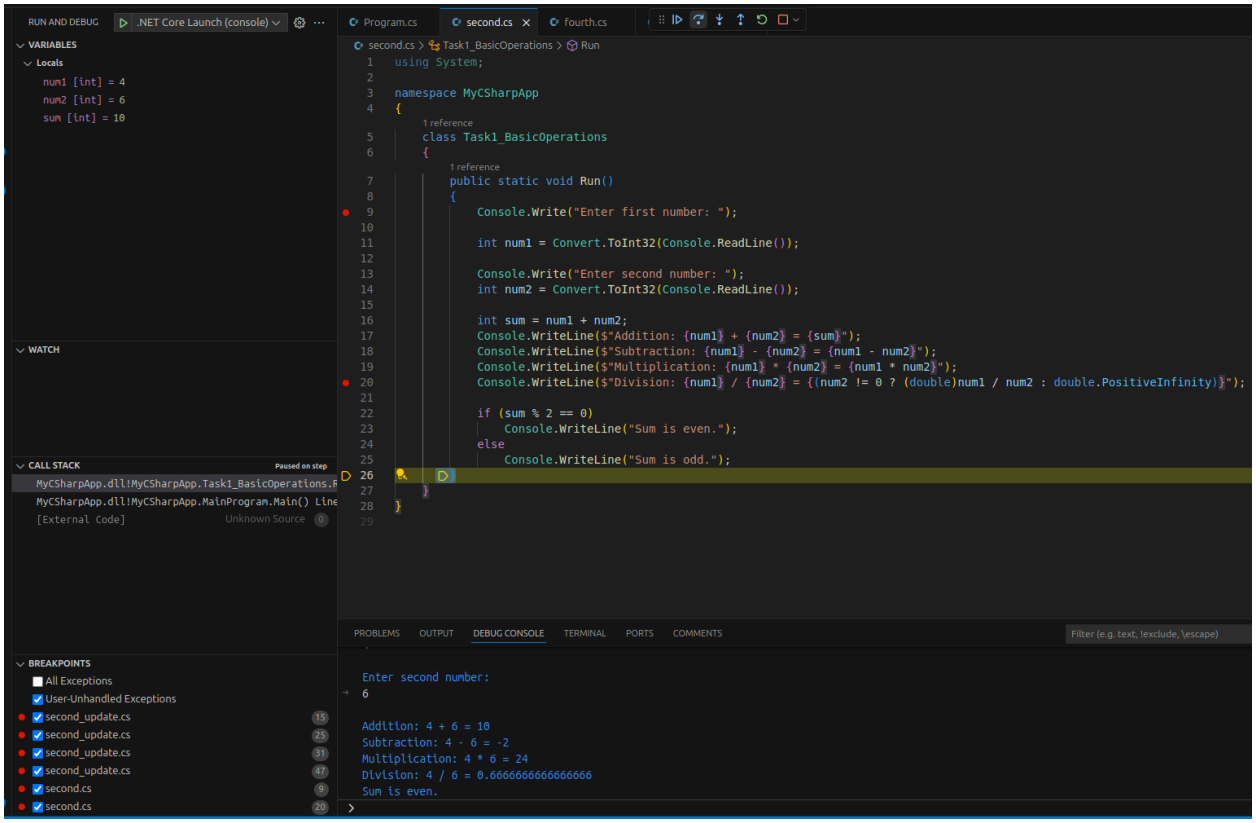
  - Start debugging:



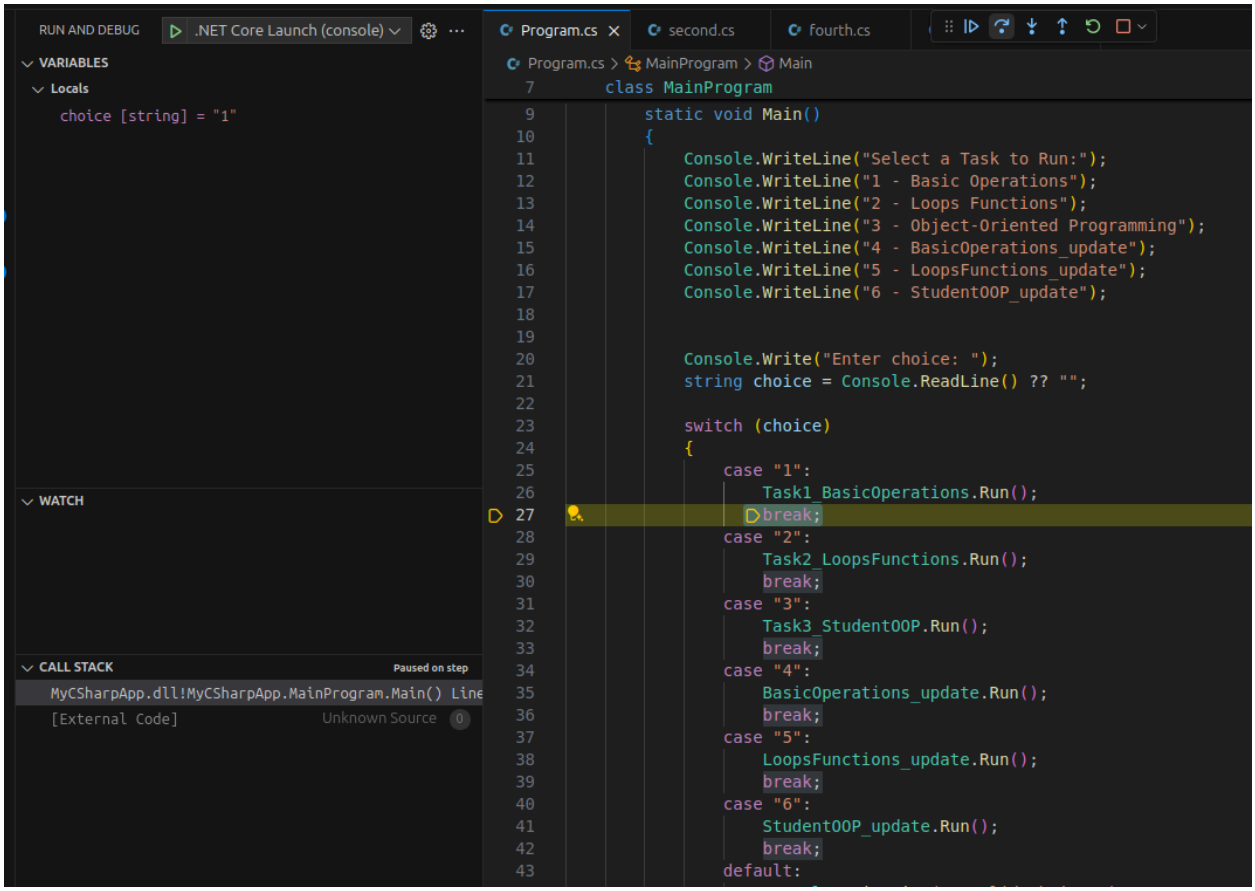
- Click on the step into to looki into method, how it is taking input and assigning to the variable input1.


RUN AND DEBUG
.NET Core Launch (console)
VARIABLES
Locals
System.Console.ReadLine returned [string] = "4"
System.Convert.ToInt32 returned [int] = 4
num1 [int] = 4
num2 [int] = 0
sum [int] = 0
WATCH
CALL STACK
Paused on step
MyCSharpApp.dll!MyCSharpApp.Task1_BasicOperations.F
MyCSharpApp.dll!MyCSharpApp.MainProgram.Main() Line
[External Code]
Unknown Source
BREAKPOINTS
All Exceptions
User-Unhandled Exceptions
second_update.cs 15
second_update.cs 25
second_update.cs 31
second_update.cs 47
second.cs 9
second.cs 20
Program.cs
second.cs
fourth.cs
second.cs > Task1_BasicOperations > Run
using System;
namespace MyCSharpApp
{
1 reference
class Task1_BasicOperations
{
1 reference
public static void Run()
{
Console.Write("Enter first number: ");
int num1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter second number: ");
int num2 = Convert.ToInt32(Console.ReadLine());
int sum = num1 + num2;
Console.WriteLine($"Addition: {num1} + {num2} = {sum}");
Console.WriteLine($"Subtraction: {num1} - {num2} = {num1 - num2}");
Console.WriteLine($"Multiplication: {num1} * {num2} = {num1 * num2}");
Console.WriteLine($"Division: {num1} / {num2} = {(num2 != 0 ? (double)num1 / num2 : double.PositiveInfinity)}");
if (sum % 2 == 0)
Console.WriteLine("Sum is even.");
else
Console.WriteLine("Sum is odd.");
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
Filter (e.g. text, !exclude, \escape)
6 - StudentOOP_update
Enter choice:
To send text to the target process's standard input, enter text into the Debug Console's evaluation box while the target process is running
chJson-Console for more information.
1
Enter first number:
4

- Last step:


RUN AND DEBUG
.NET Core Launch (console)
VARIABLES
Locals
num1 [int] = 4
num2 [int] = 6
sum [int] = 10
WATCH
CALL STACK
Paused on step
MyCSharpApp.dll!MyCSharpApp.Task1_BasicOperations.F
MyCSharpApp.dll!MyCSharpApp.MainProgram.Main() Line
[External Code]
Unknown Source
BREAKPOINTS
All Exceptions
User-Unhandled Exceptions
second_update.cs 15
second_update.cs 25
second_update.cs 31
second_update.cs 47
second.cs 9
second.cs 20
Program.cs
second.cs
fourth.cs
second.cs > Task1_BasicOperations > Run
using System;
namespace MyCSharpApp
{
1 reference
class Task1_BasicOperations
{
1 reference
public static void Run()
{
Console.Write("Enter first number: ");
int num1 = Convert.ToInt32(Console.ReadLine());
Console.Write("Enter second number: ");
int num2 = Convert.ToInt32(Console.ReadLine());
int sum = num1 + num2;
Console.WriteLine($"Addition: {num1} + {num2} = {sum}");
Console.WriteLine($"Subtraction: {num1} - {num2} = {num1 - num2}");
Console.WriteLine($"Multiplication: {num1} * {num2} = {num1 * num2}");
Console.WriteLine($"Division: {num1} / {num2} = {(num2 != 0 ? (double)num1 / num2 : double.PositiveInfinity)}");
if (sum % 2 == 0)
Console.WriteLine("Sum is even.");
else
Console.WriteLine("Sum is odd.");
}
}
}
PROBLEMS
OUTPUT
DEBUG CONSOLE
TERMINAL
PORTS
COMMENTS
Filter (e.g. text, !exclude, \escape)
Enter second number:
6
Addition: 4 + 6 = 10
Subtraction: 4 - 6 = -2
Multiplication: 4 * 6 = 24
Division: 4 / 6 = 0.6666666666666666
Sum is even.

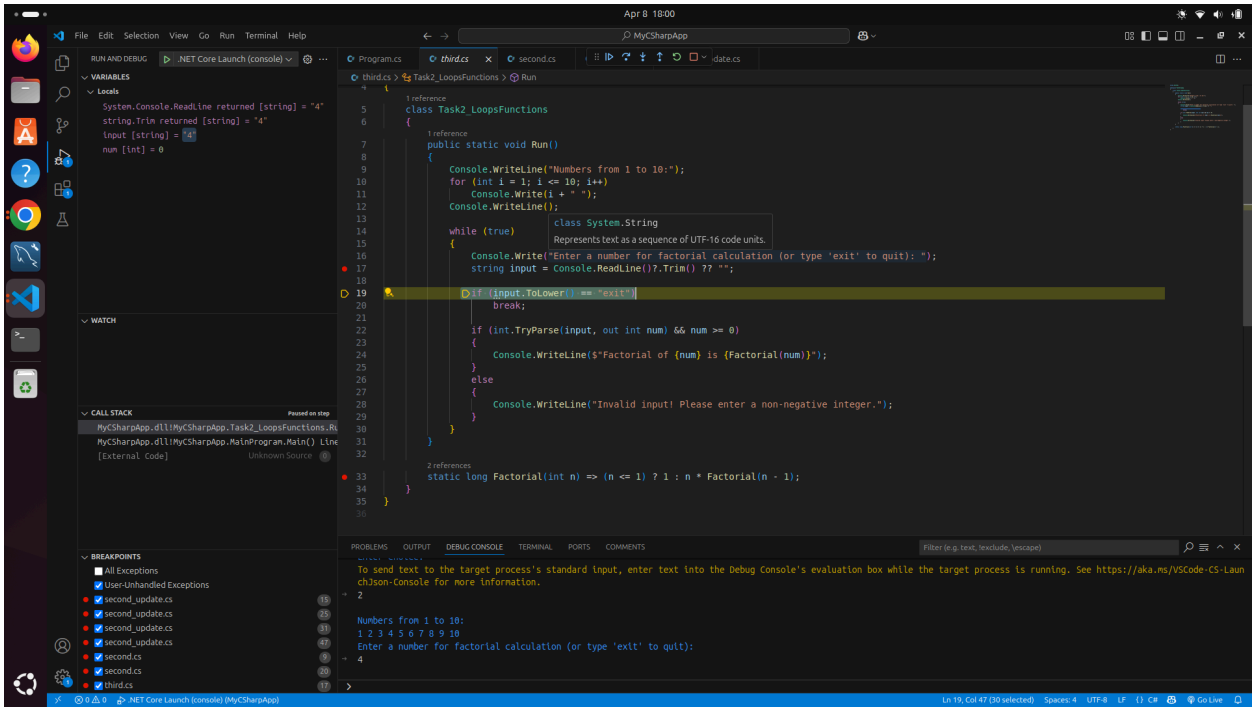
- Last step to break from program.cs



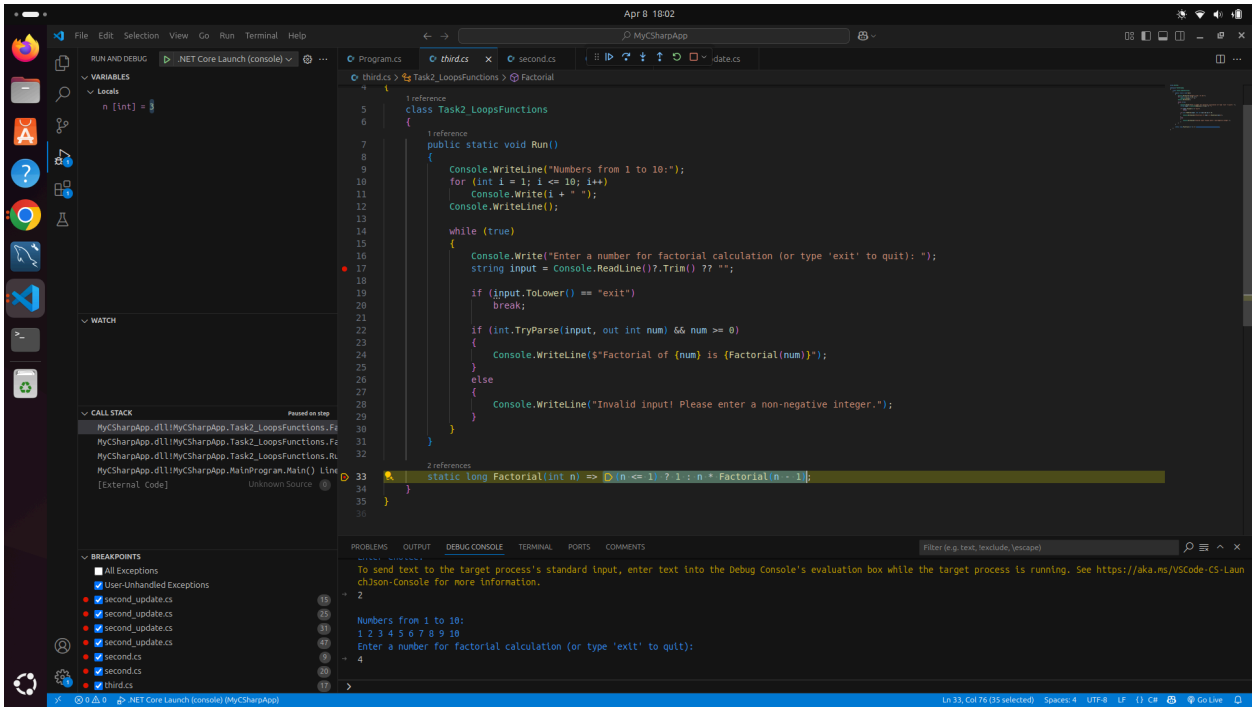
- **Activity_3:**
  - Using two breakpoints:

```
Program.cs | third.cs | second.cs | fourth.cs | fourth_update.cs
third.cs > Task2_LoopsFunctions > Run
{
    1 reference
    class Task2_LoopsFunctions
    {
        1 reference
        public static void Run()
        {
            Console.WriteLine("Numbers from 1 to 10:");
            for (int i = 1; i <= 10; i++)
                Console.Write(i + " ");
            Console.WriteLine();

            while (true)
            {
                Console.Write("Enter a number for factorial calculation (or type 'exit' to quit): ");
                string input = Console.ReadLine()?.Trim() ?? "";

                if (input.ToLower() == "exit")
                    break;

                if (int.TryParse(input, out int num) && num >= 0)
                {
                    Console.WriteLine($"Factorial of {num} is {Factorial(num)}");
                }
                else
                {
                    Console.WriteLine("Invalid input! Please enter a non-negative integer.");
                }
            }
        }

        2 references
        static long Factorial(int n) => (n <= 1) ? 1 : n * Factorial(n - 1);
    }
```

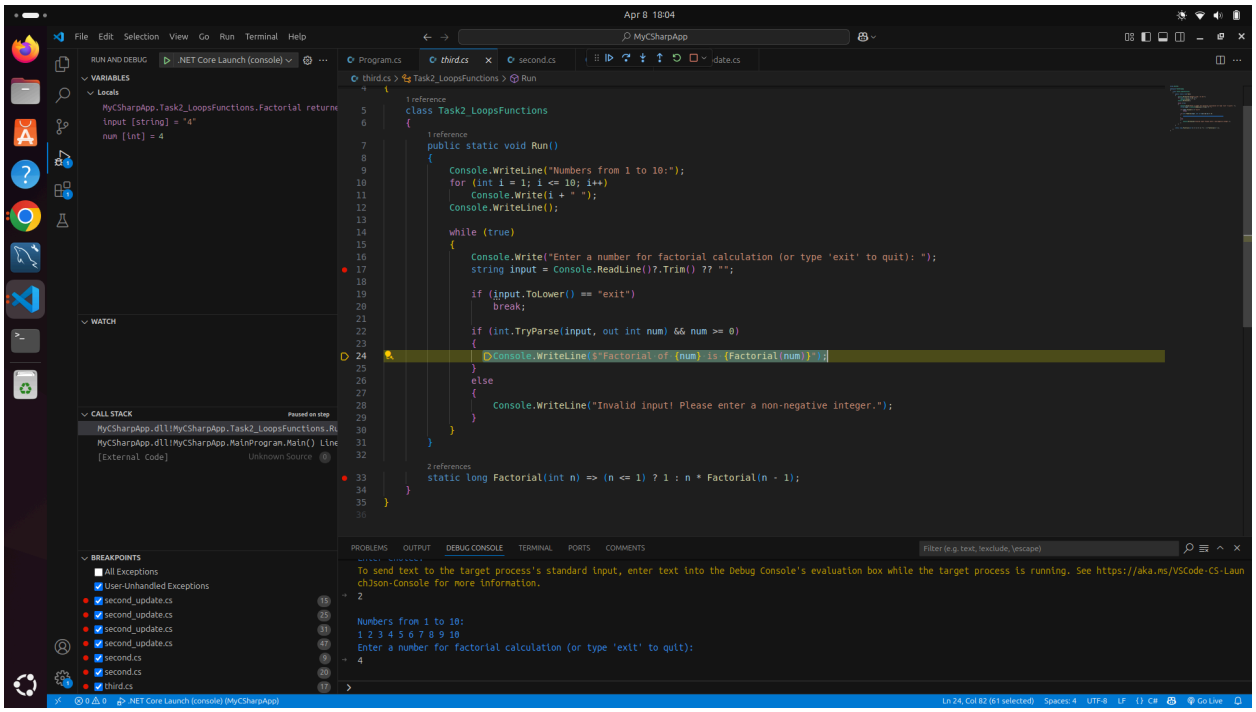
- Enter the input for calculating factorial:



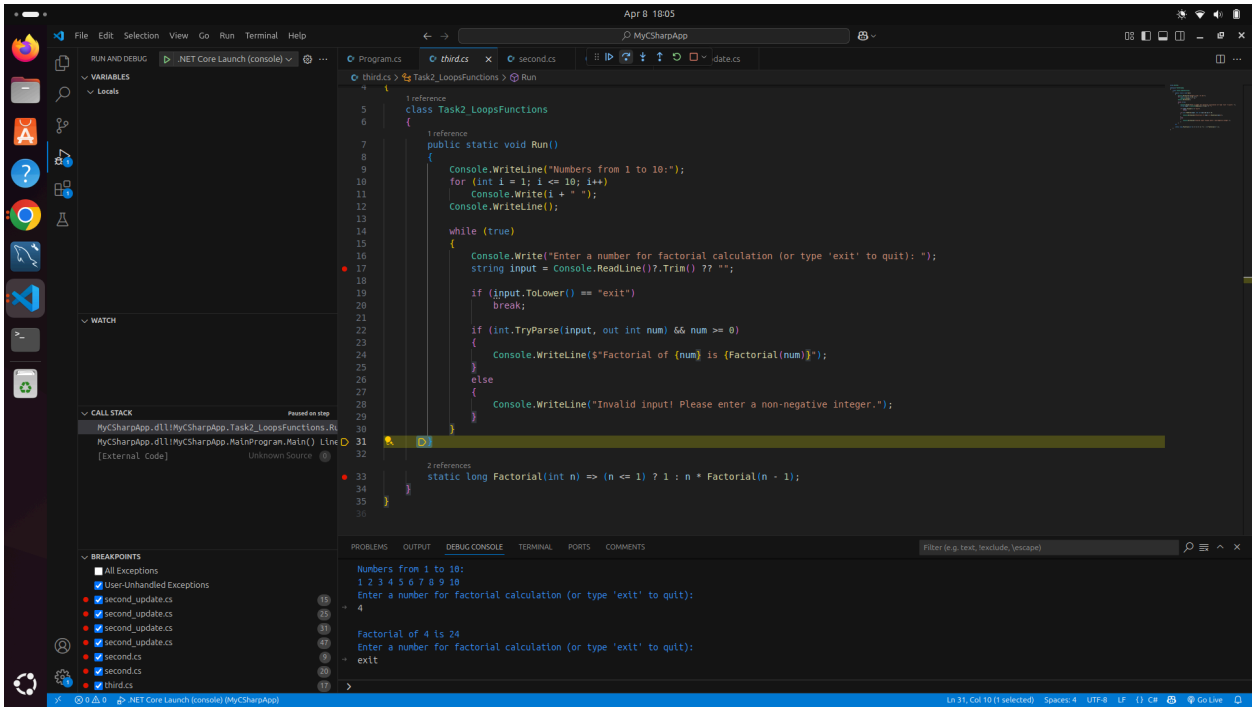
- Use step-into here to see the working of factorial:



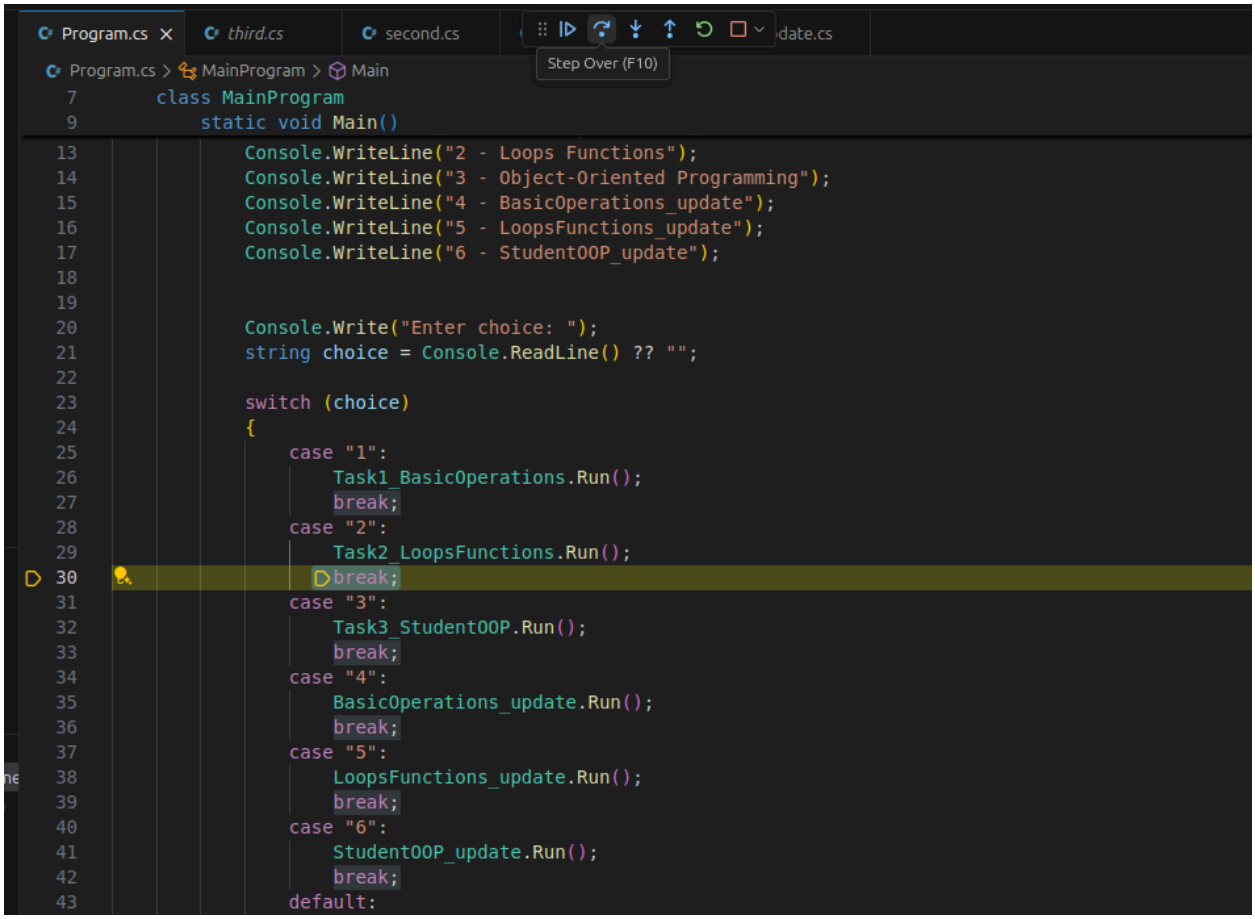
- Use step-out to come out of the method:



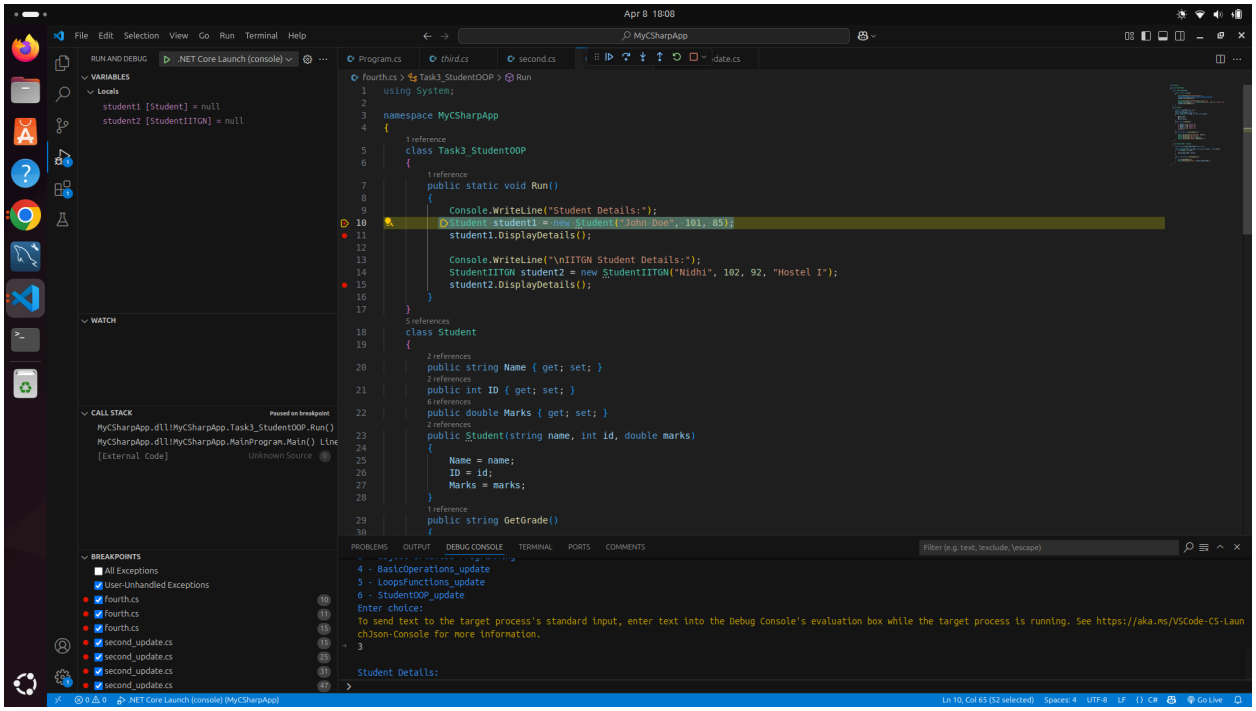
- Use exit to come out of the loop:



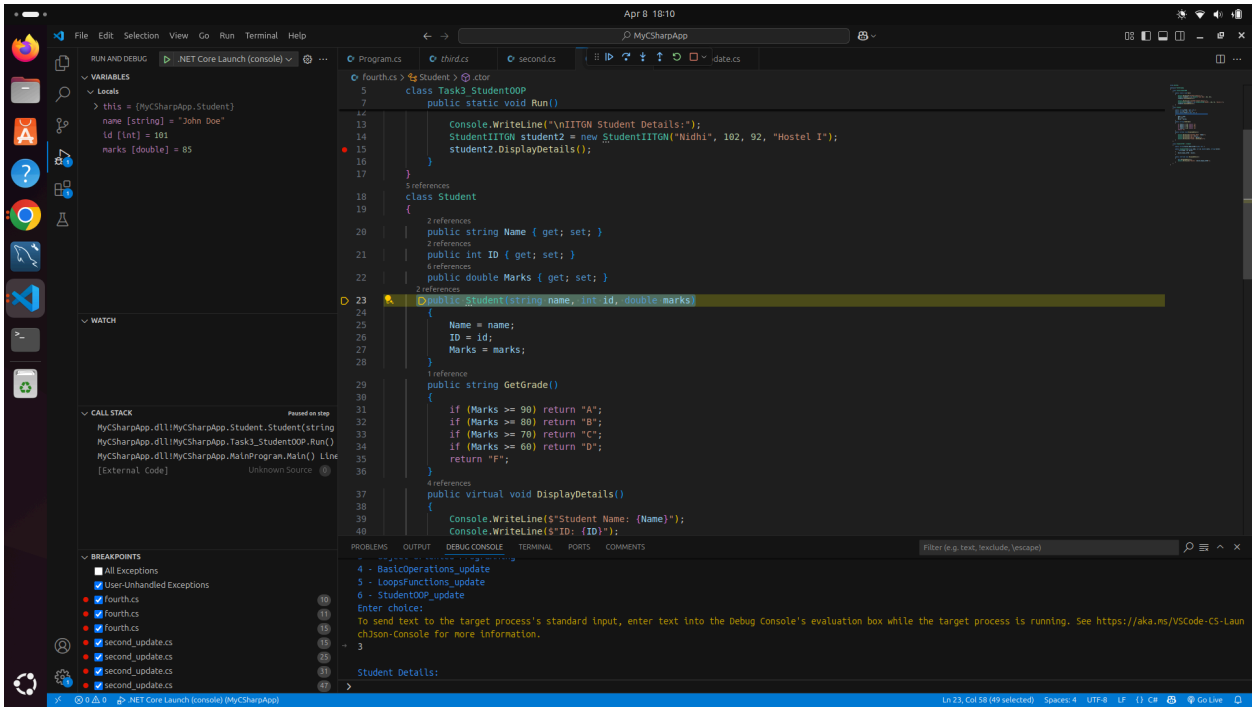
- The last step is to break from the program.cs:



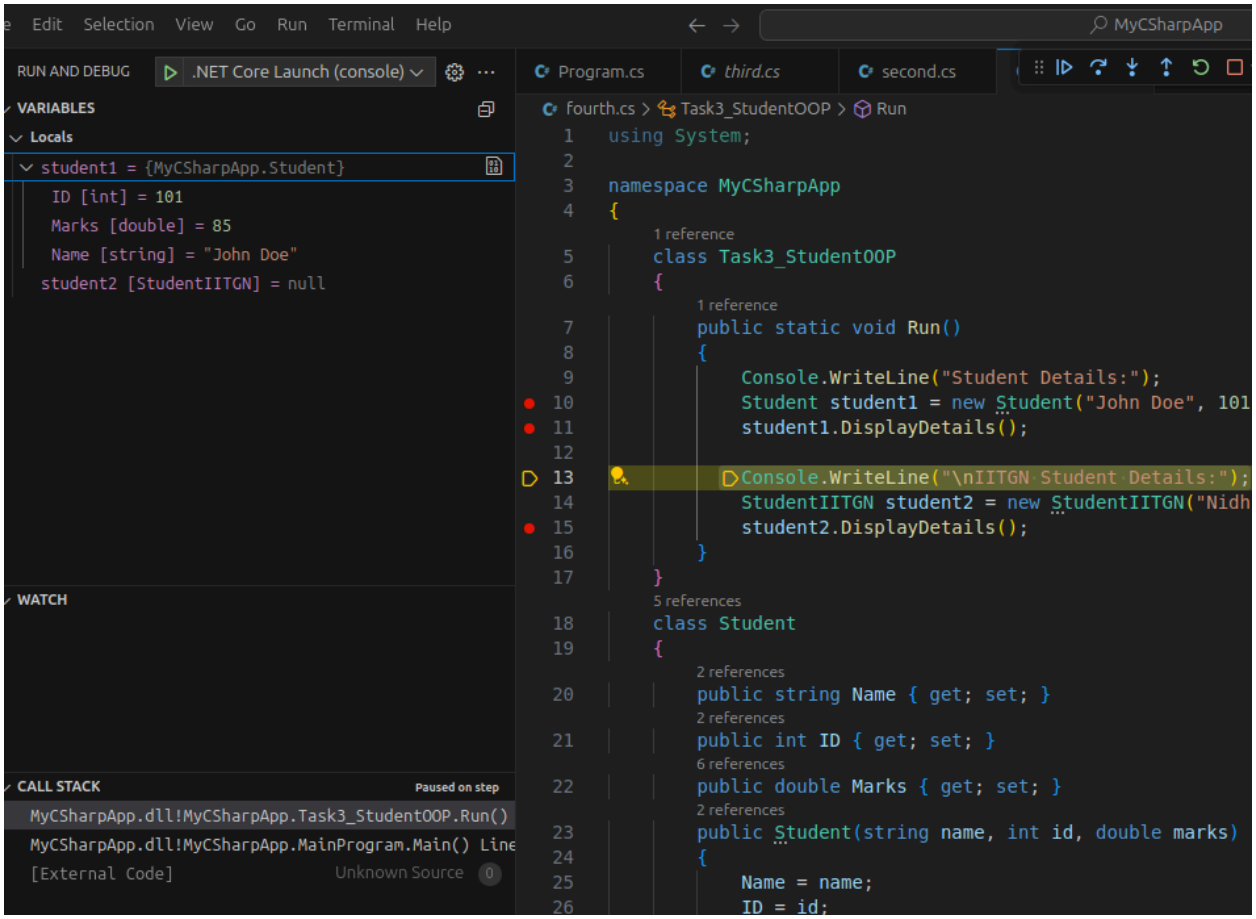
- **Activity_4:**
  - For this, using 3 breakpoints:



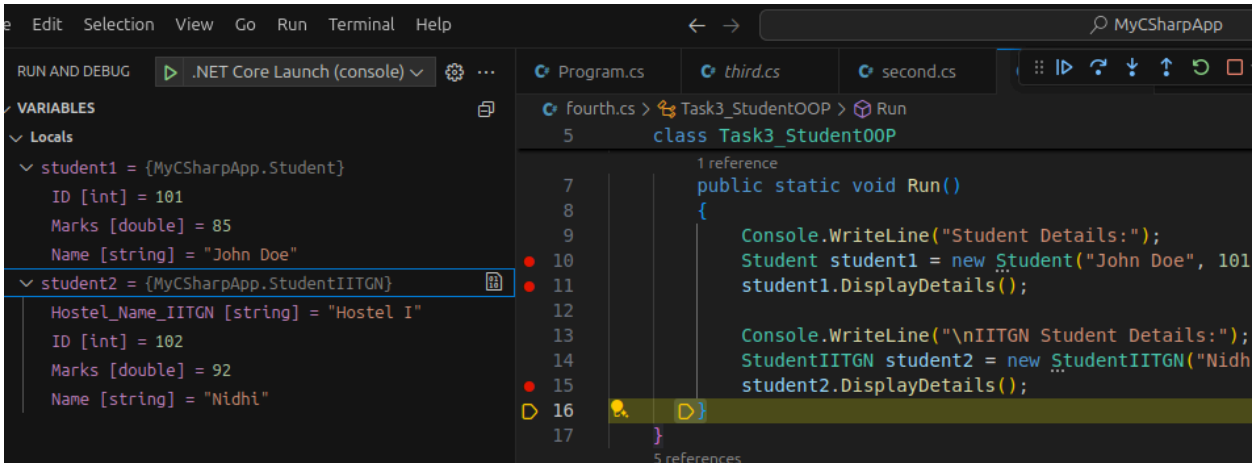
  - Use the step into here to go inside the constructor and see how Name, ID, and Marks are initialized.



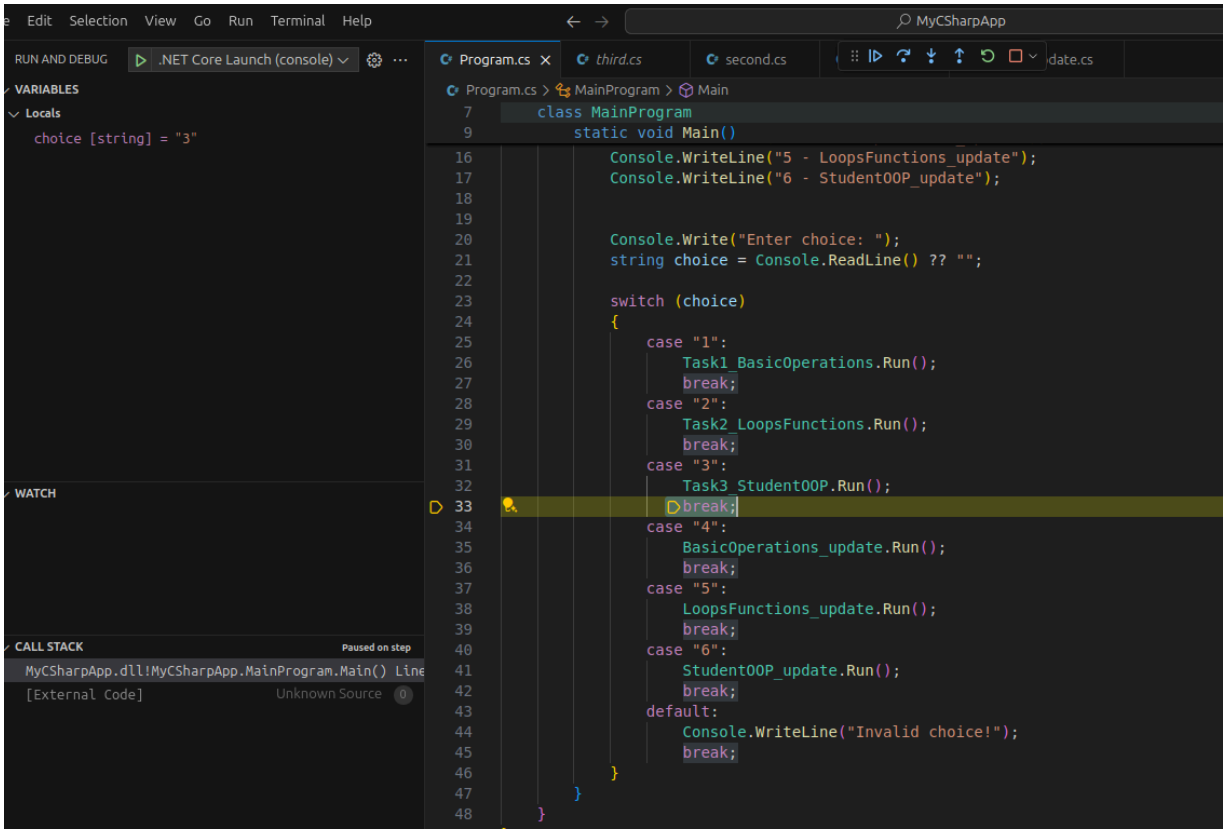
- We can see the details of student1:



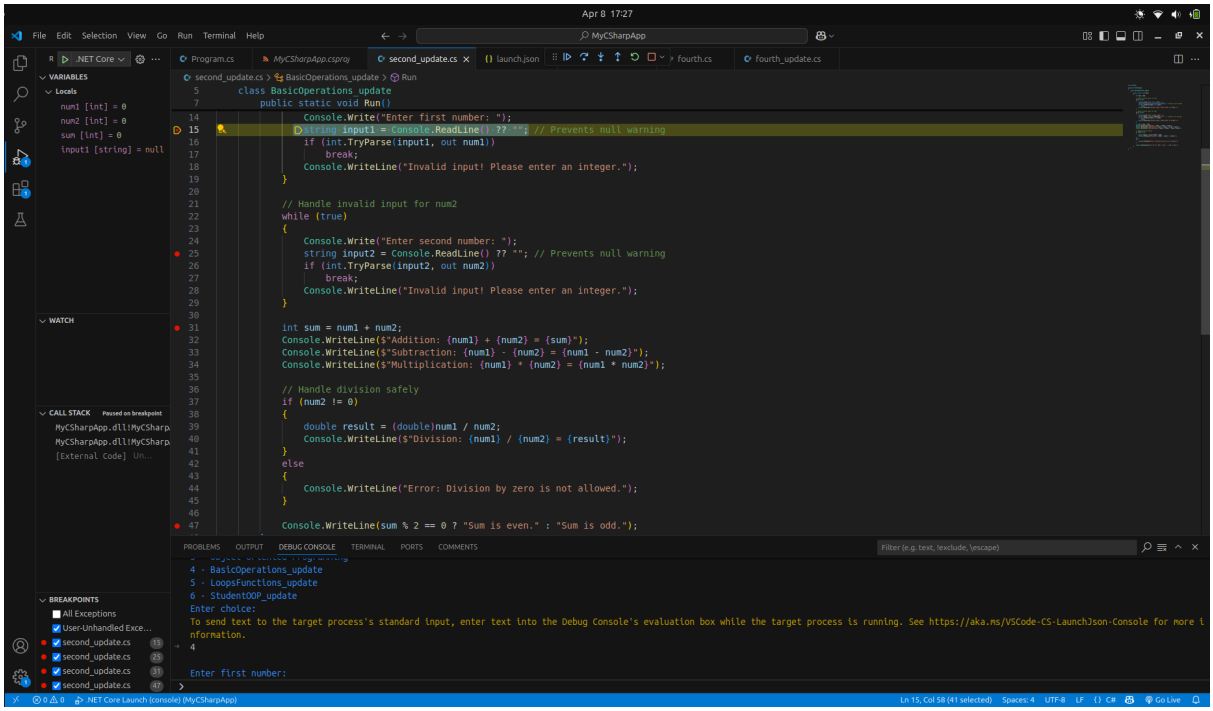
- Similarly, see the student2's detail.



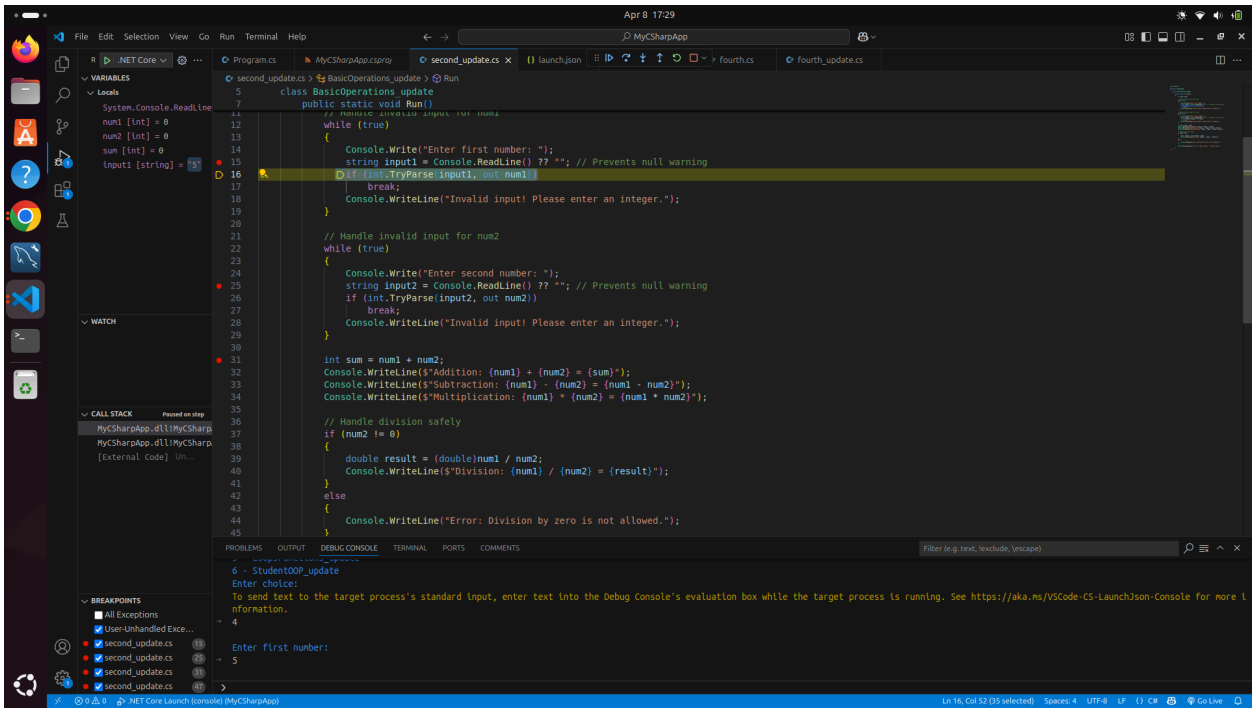
- Last step to break:



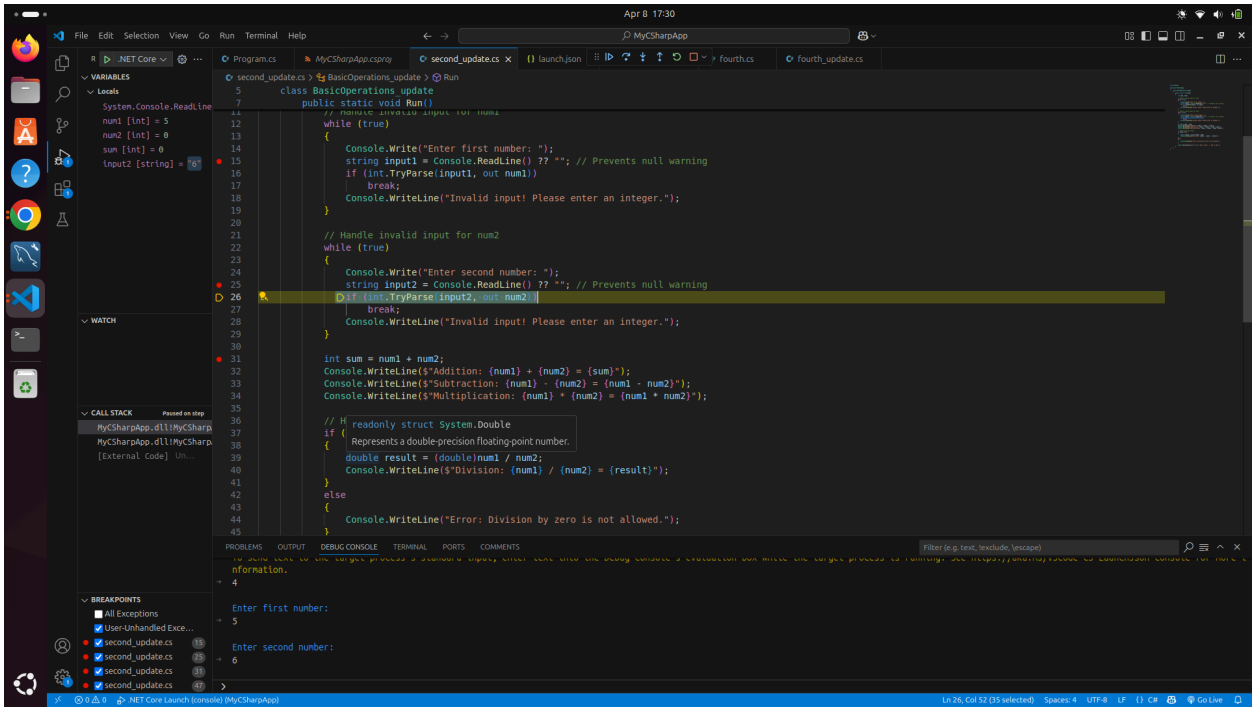
- **Activity_5:**
  - Activity_2_update: In this code, I use 4 breakpoints:

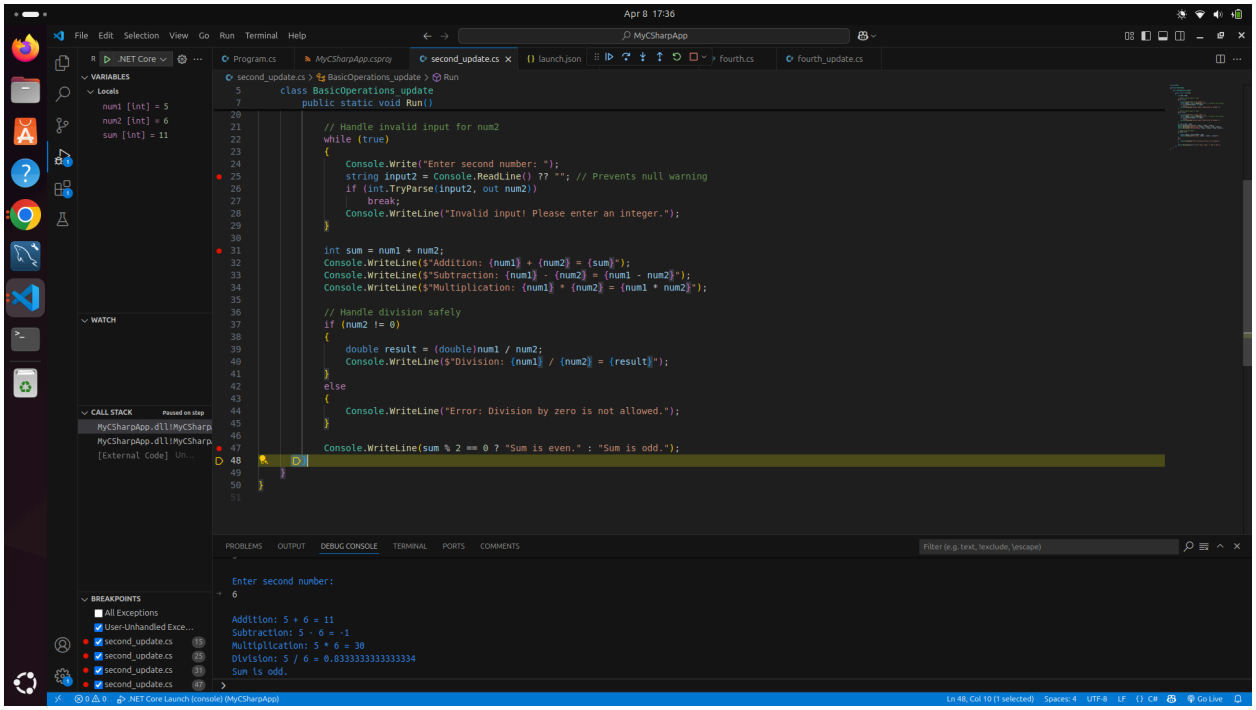


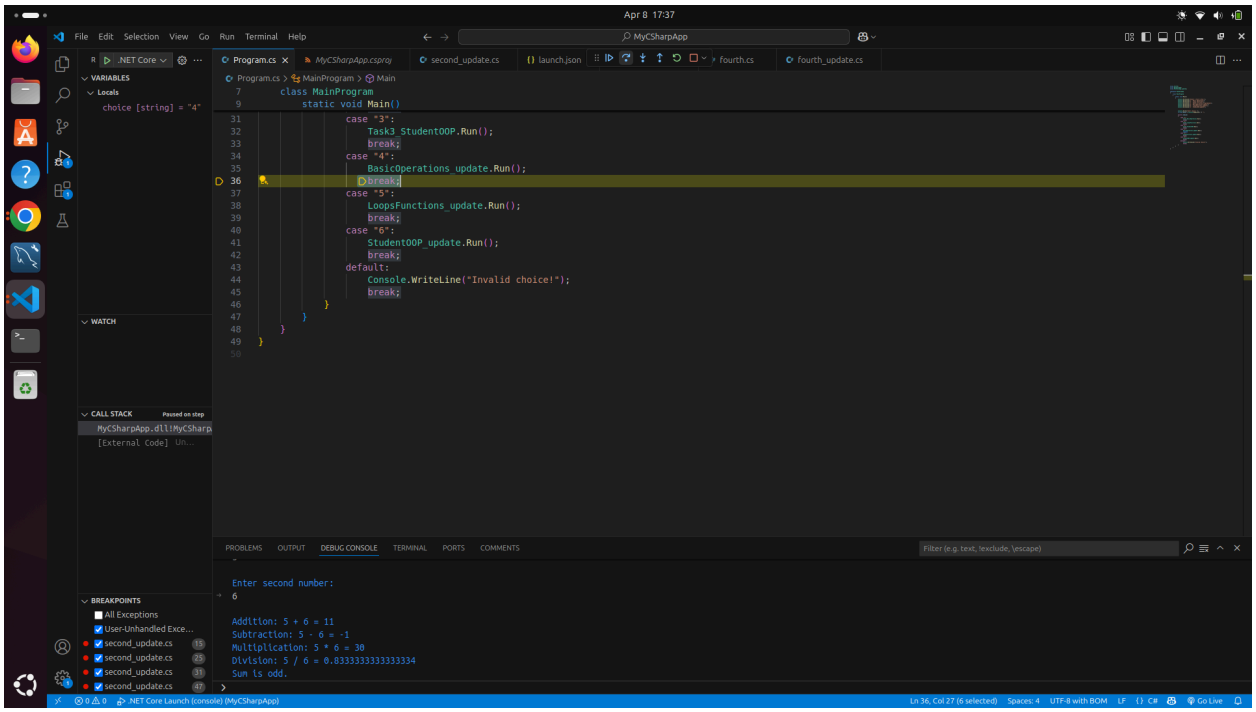
- Start debugging (click f5).
- On clicking the step over, and give the input1.



- Give the value of input2

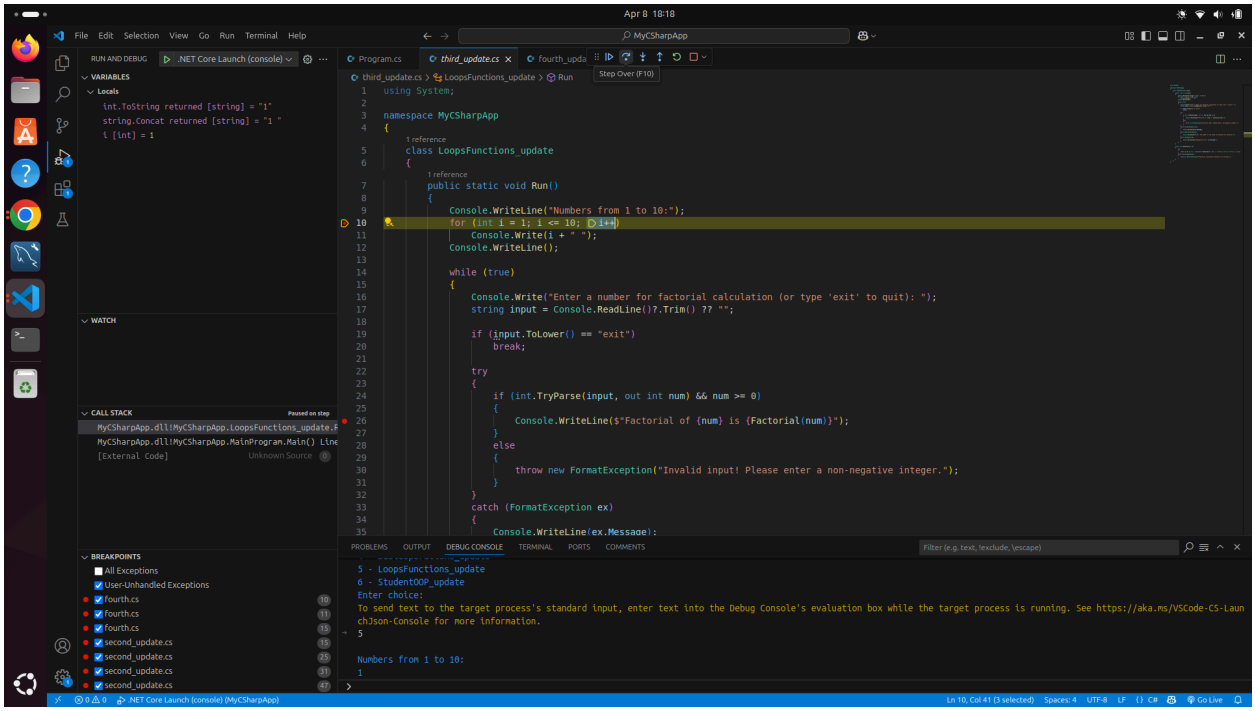




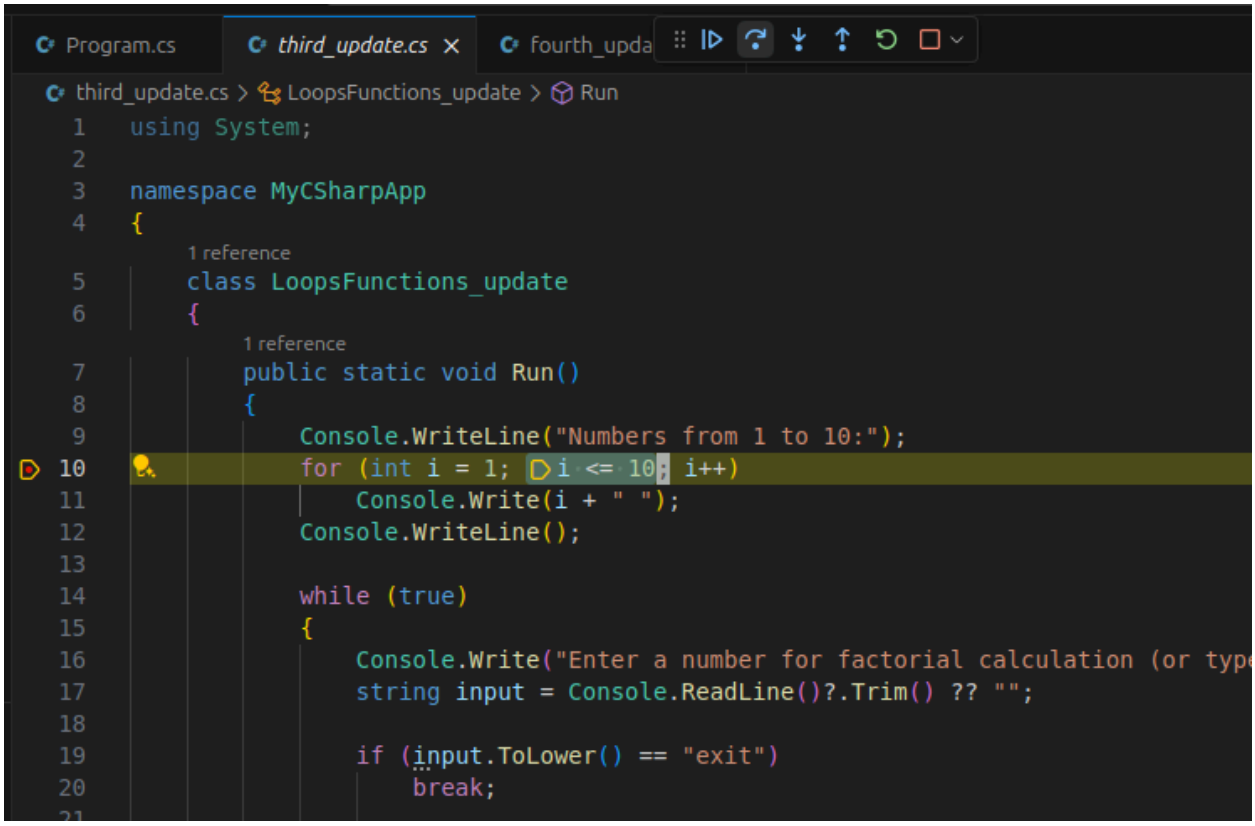


- This is when the last line gets executed using step-over.

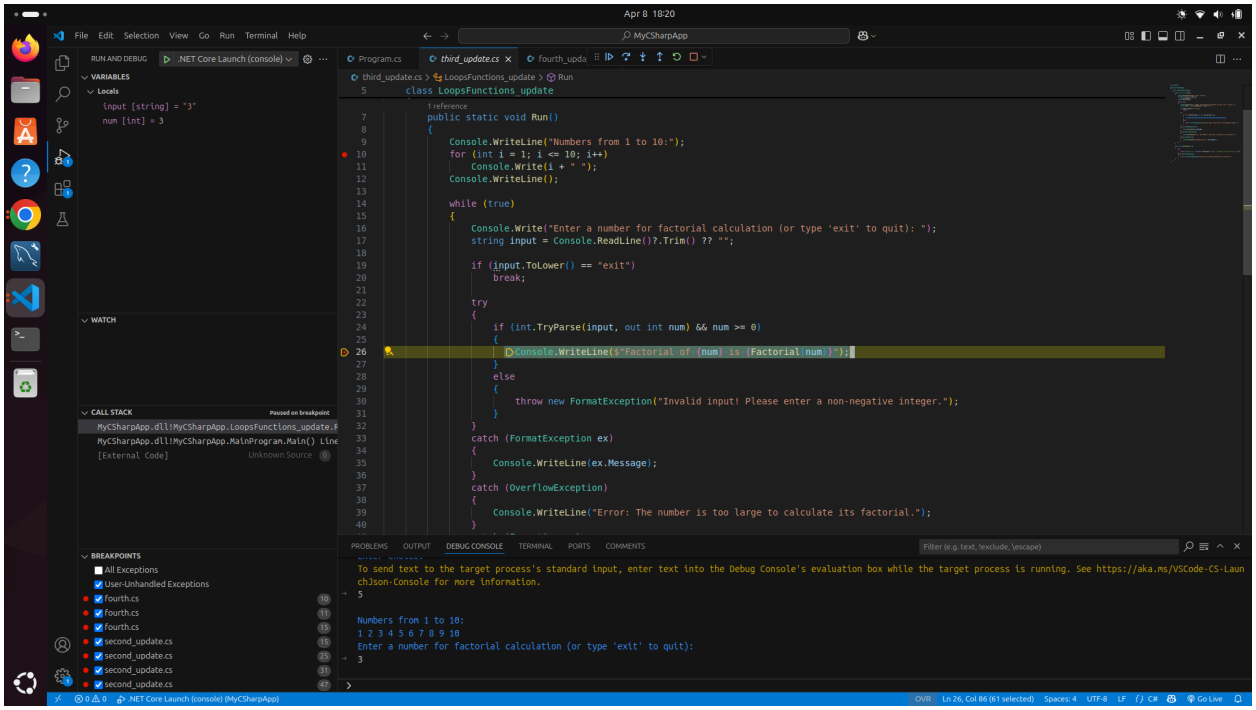
- Activity_3_update: In this code, I use 2 breakpoints:



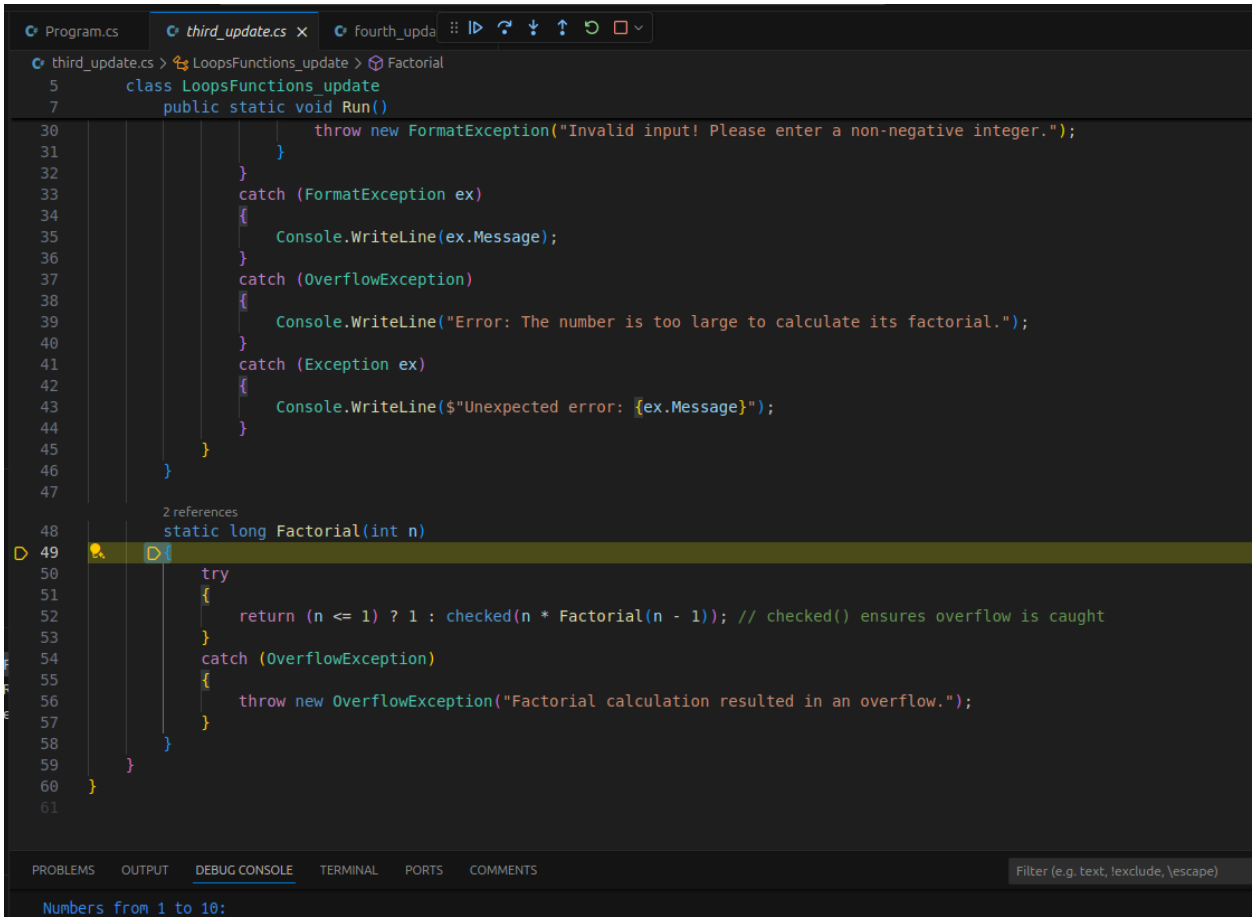
- Use step-over here to observe how the loop prints numbers from 1 to 10.



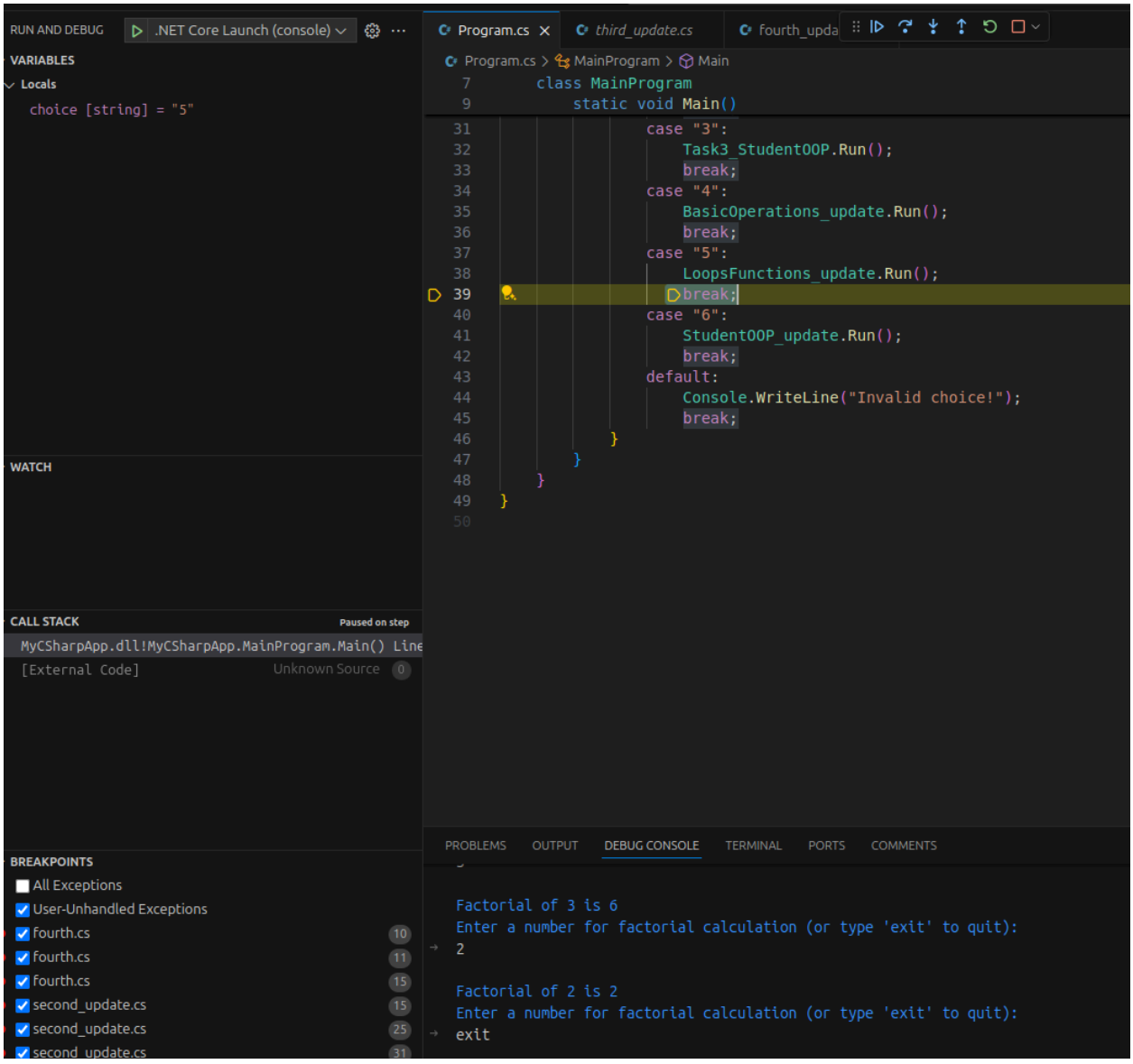
- Use step out to come out of the loop without waiting for the loop to print all numbers:



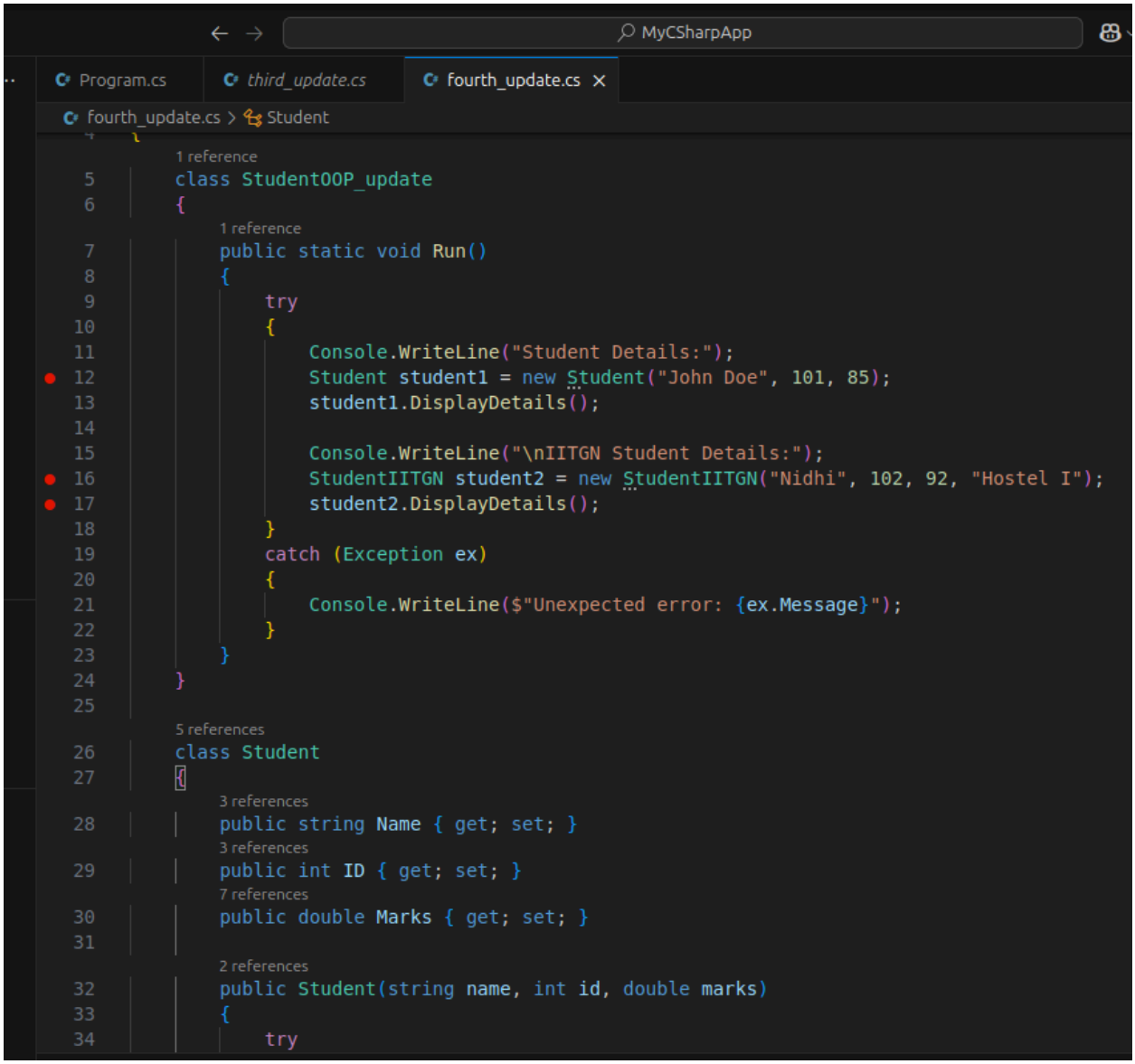
- Use step into at 2nd breakpoint to go inside ‘Factorial(num)’, and observe recursion in action.



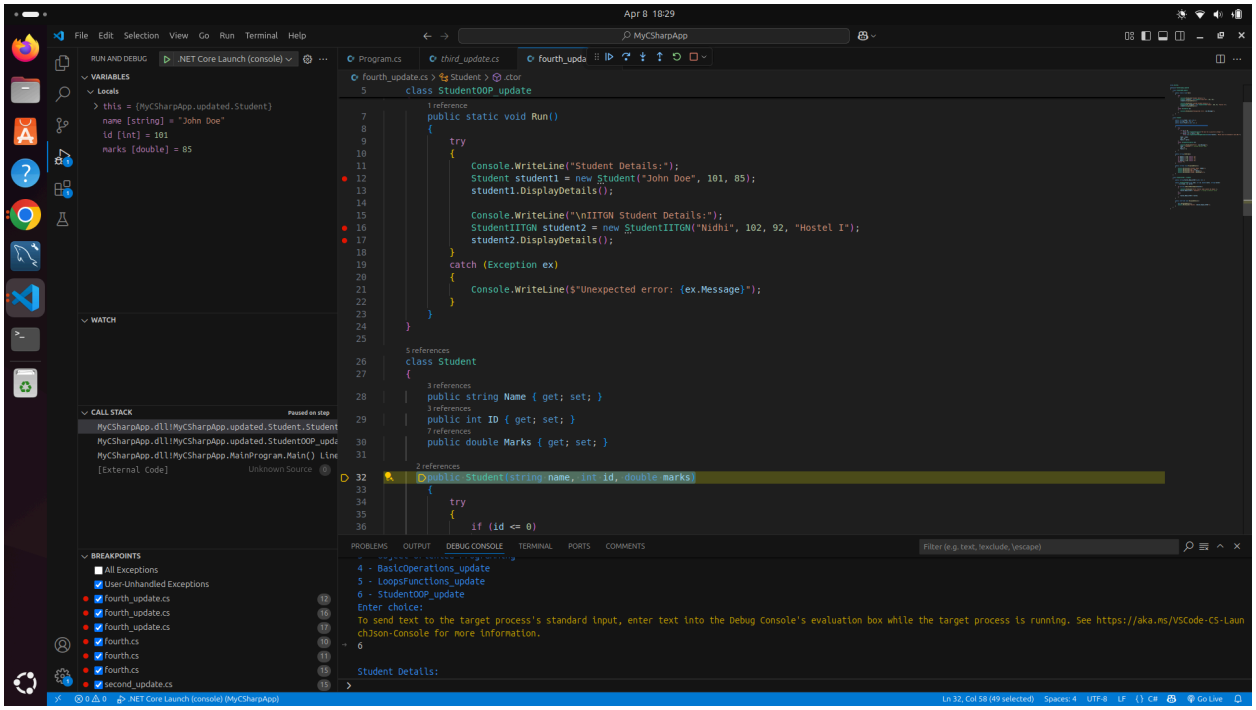
- Use exit to break the loop and then last step in the program.cs:



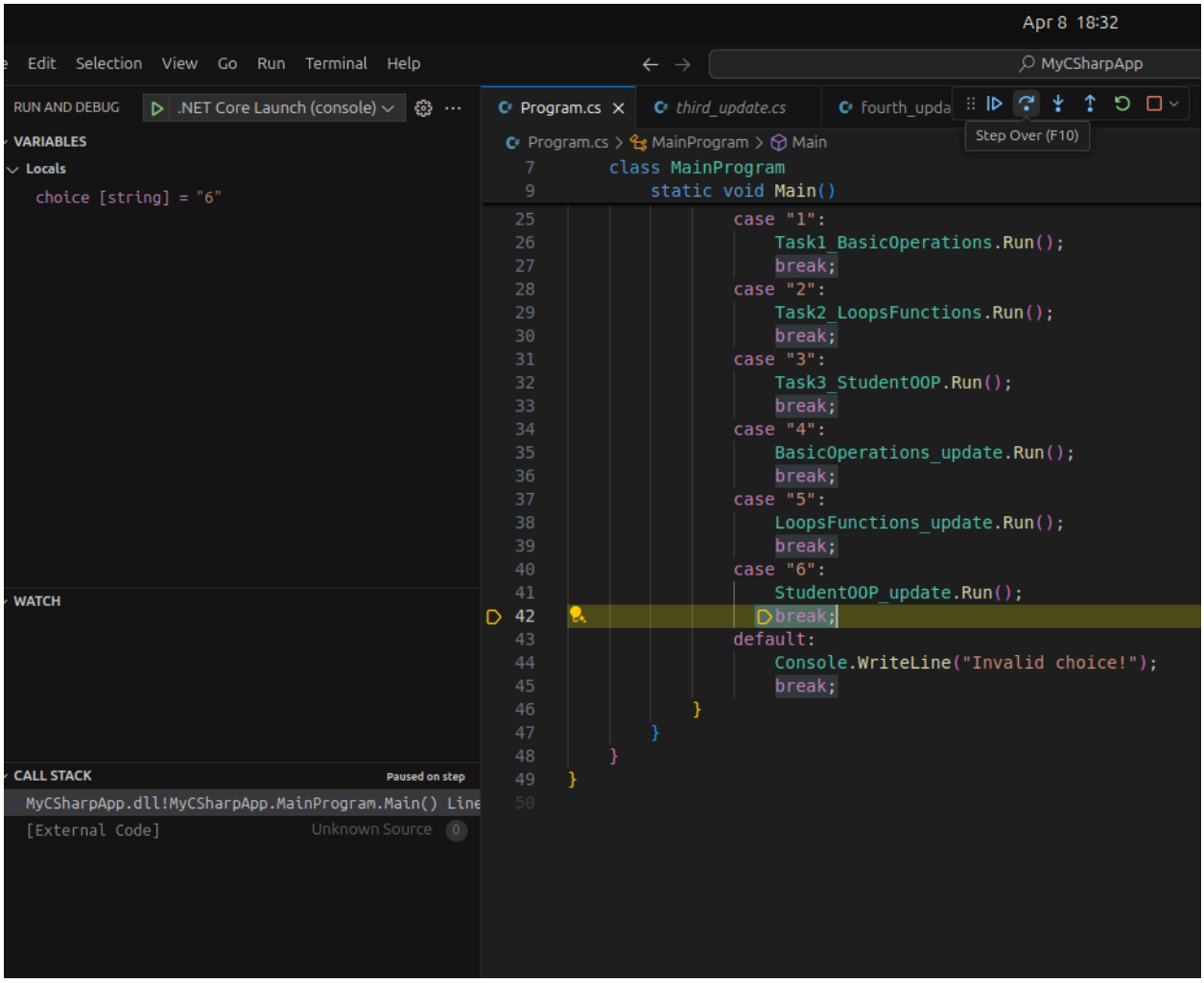
- Activity_4_update: In this code, I use 3 breakpoints:



- Start debugging and use the step-into to go into the constructor logic and see error handling.



- Same for the second and third breakpoints, and then finally, it complete and exits:



Result and Analysis:

- Successfully implemented basic syntax, conditionals, loops, and functions.
- Created classes using object-oriented principles like inheritance and method overriding.
- Used try-catch blocks to manage runtime errors like invalid user input or division by zero, ensuring the program doesn't crash and shows a proper error message instead.
- Debugging:
    - Use step-over to move line by line within the same method (without jumping inside the function).
    - Use step-in to go inside the method and know what exactly is happening inside the method.
    - Use step-out when we are inside the method and want to exit back to the caller quicker.
- Even when unexpected inputs were entered, programs operated without crashing.
- Developed a hands-on understanding of C# debugging and development methods.

Q) 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?

- A copy constructor generates a new object by copying values from an existing object. In C#, we can define it explicitly, but it's not built-in as in C++.
  For example, we have:

```
public Student(Student existing)
{
    Name = existing.Name;
    ID = existing.ID;
    Marks = existing.Marks;
}
```

  We can create copy using:

```
Student s2 = new Student(s1);
```

- In C#, it is possible to overload constructors by declaring several constructors with distinct parameters.
  For example, we have:

```
public Student(string name, int id)
                {
            Name = name;
              ID = id;
        Marks = 0; // default
                }
```

  so now we can create:

```
Student s1 = new Student("Nidhi", 102); // overloaded
Student s2 = new Student("Nidhi", 102, 92); // original
```

Q) 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()?

- Main() method needs to be static because no objects are created before the application begins. Because it is static, that is, it is a member of the class rather than an object. The runtime can call it directly without creating any instances.
- If we call Main() function from itself, then recursion is the result of that. Without a base case, stack overflow (a crash brought on by an endless number of calls) occurs.

```
static void Main()
{
Console.WriteLine("Calling myself");
Main(); // BAD: infinite recursion!
}
```

- If we spell Main() as main(), it won't work. C# is **case-sensitive**, so main() is just another method, not the program entry point. We will get:
  'Program does not contain a static 'Main' method suitable for an entry point'.

Q) What happens when both base-class and derived-class have their corresponding Main() methods?

C# allows this, but only one Main() is the actual entry point when running the program.

For example:

```
class BaseClass{
public static void Main() { Console.WriteLine("Base"); }
}

class DerivedClass : BaseClass{
public static void Main() { Console.WriteLine("Derived"); }
}
```

If there is ambiguity (such as multiple classes), the compiler will ask us to specify which Main() to use when we run this program. It can be fixed by:

- In Visual Studio, setting the Startup object, or
- eliminating a single Main().

Conclusion and Discussion:

- Challenges:
  - Getting error:

fourth_update.cs

```
using System;

namespace MyCSharpApp
{
    class StudentOOP_update
    {
        public static void Run()
        {
            try
            {
                Console.WriteLine("Student Details:");
                Student student1 = new Student("John Doe", 101, 85);
                student1.DisplayDetails();

                Console.WriteLine("\nIITGN Student Details:");
                StudentIITGN student2 = new StudentIITGN("Nidhi", 102, 92, "Hostel I");
                student2.DisplayDetails();
            }
            catch (Exception ex)
            {
                Console.WriteLine($"Unexpected error: {ex.Message}");
            }
        }
    }

    class Student
    {
        public string Name { get; set; }
        public int ID { get; set; }
        public double Marks { get; set; }

        public Student(string name, int id, double marks)
        {
            try
            {
                if (id <= 0)
                    throw new ArgumentException("ID must be a positive integer.");
                if (marks < 0 || marks > 100)
                    throw new ArgumentOutOfRangeException(nameof(marks), "Marks must be between 0 and 100.");
```

fourth_update.cs

```
                Name = name;
                ID = id;
                Marks = marks;
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
                ID = 0; // Assigning a default value to prevent errors.
            }
            catch (ArgumentOutOfRangeException ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
                Marks = 0; // Assigning a default value to prevent errors.
            }
        }

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

        public virtual void DisplayDetails()
        {
            Console.WriteLine($"Student Name: {Name}");
            Console.WriteLine($"ID: {ID}");
            Console.WriteLine($"Marks: {Marks}");
            Console.WriteLine($"Grade: {GetGrade()}");
        }
    }

    class StudentIITGN : Student
    {
        public string Hostel_Name_IITGN { get; set; }
```

```
        public StudentIITGN(string name, int id, double marks, string hostel)
            : base(name, id, marks)
        {
            try
            {
                if (string.IsNullOrWhiteSpace(hostel))
                    throw new ArgumentException("Hostel name cannot be empty.");

                Hostel_Name_IITGN = hostel;
            }
            catch (ArgumentException ex)
            {
                Console.WriteLine($"Error: {ex.Message}");
                Hostel_Name_IITGN = "Unknown"; // Assigning a default value
            }
        }

        public override void DisplayDetails()
        {
            base.DisplayDetails();
            Console.WriteLine($"Hostel: {Hostel_Name_IITGN}");
        }
    }
}
```

```
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$ dotnet run
/home/nidhi/MyCSharpApp/fourth_update.cs(26,11): error CS0101: The namespace 'MyCSharpApp' already contains a definition for 'Student' [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(75,11): error CS0101: The namespace 'MyCSharpApp' already contains a definition for 'StudentIITGN' [/home/nidhi/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(32,16): error CS0111: Type 'Student' already defines a member called 'Student' with the same parameter types [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(79,16): error CS0111: Type 'StudentIITGN' already defines a member called 'StudentIITGN' with the same parameter types [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(96,30): error CS0111: Type 'StudentIITGN' already defines a member called 'DisplayDetails' with the same parameter types [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(57,23): error CS0111: Type 'Student' already defines a member called 'GetGrade' with the same parameter types [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]
/home/nidhi/MyCSharpApp/fourth_update.cs(66,29): error CS0111: Type 'Student' already defines a member called 'DisplayDetails' with the same parameter types [/home/nidhi/MyCSharpApp/MyCSharpApp.csproj]

The build failed. Fix the build errors and run again.
nidhi@nidhi-VivoBook-ASUSLaptop-X415JA-X415JA:~/MyCSharpApp$
```

This is because the namespace 'MyCSharpApp' already contains a definition for Student (in fourth.cs file). So to remove this issue, we can use a separate namespace here and import it into the program.cs

- I initially faced difficulty understanding how and when to effectively use debugging tools like step-over, step-into, and step-out.
- At first, handling exceptions correctly was a little challenging, particularly when handling unusual inputs like empty strings or negative values. To ensure that the program could handle every possible edge case without crashing, it was necessary to conduct thorough testing and use `try-catch` blocks cautiously.

- Reflections:

I now have a better understanding of fundamental C# concepts, including object-oriented programming, loops, functions, conditionals, input handling, and exception handling. I practiced debugging with Visual Studio 2022 using breakpoints and step-into, step-over,

and step-out tools. My understanding of clean and reusable code improved due to implementing constructor overloading, inheritance, and method overriding.

- Lesson learned:
  Learned to create reliable C# programs with appropriate structure and error handling. Additionally, I became more comfortable with debugging techniques and had a better understanding of how `Main()` functions as the program's entry point. I also realized how crucial object-oriented concepts like constructor overloading, inheritance, and method overriding are to writing modular and maintainable code.

- Summary:
  Use Visual Studio 2022 to create and test a number of C# console applications in this lab. The assignments required putting fundamental programming ideas like conditional statements, loops, functions, and basic syntax into practice. Additionally, classes, inheritance, copy constructors, constructor overloading, and method overriding were used to create object-oriented programming. Also worked on exception handling using `try-catch` blocks to strengthen the programs' resistance to runtime errors and invalid inputs. Debugging techniques such as breakpoints, step-over, step-into, and step-out were used to track the execution of the code and comprehend the flow. All things considered, this lab strengthened my knowledge of C# structured and object-oriented programming.