

# CE103 Algorithms and Programming I

Week-4

Fall Semester, 2021-2022

Download [DOC](#), [SLIDE](#), [PPTX](#)



## Introduction to Code Reusability and Automated Testing

During this course we will use entry level of shared library development and their tests and test automations. Also we will see TDD(Test Driven Development) approach.

During this course we will use **Windows OS, Eclipse and Visual Studio Community Edition** environments for examples.

Each example will include two function

"Hello " printing function with name sayHelloTo(name) and

sum of two variable function for basic, sum = sum(a,b).

This sum function will add a to b and return result to sum variable.

We will locate them in library and use them from a console application, also we will create unit tests for testing their functionalities and return variables

# Shared Library Development

## C Programming (Static Library)

### Visual Studio Community Edition

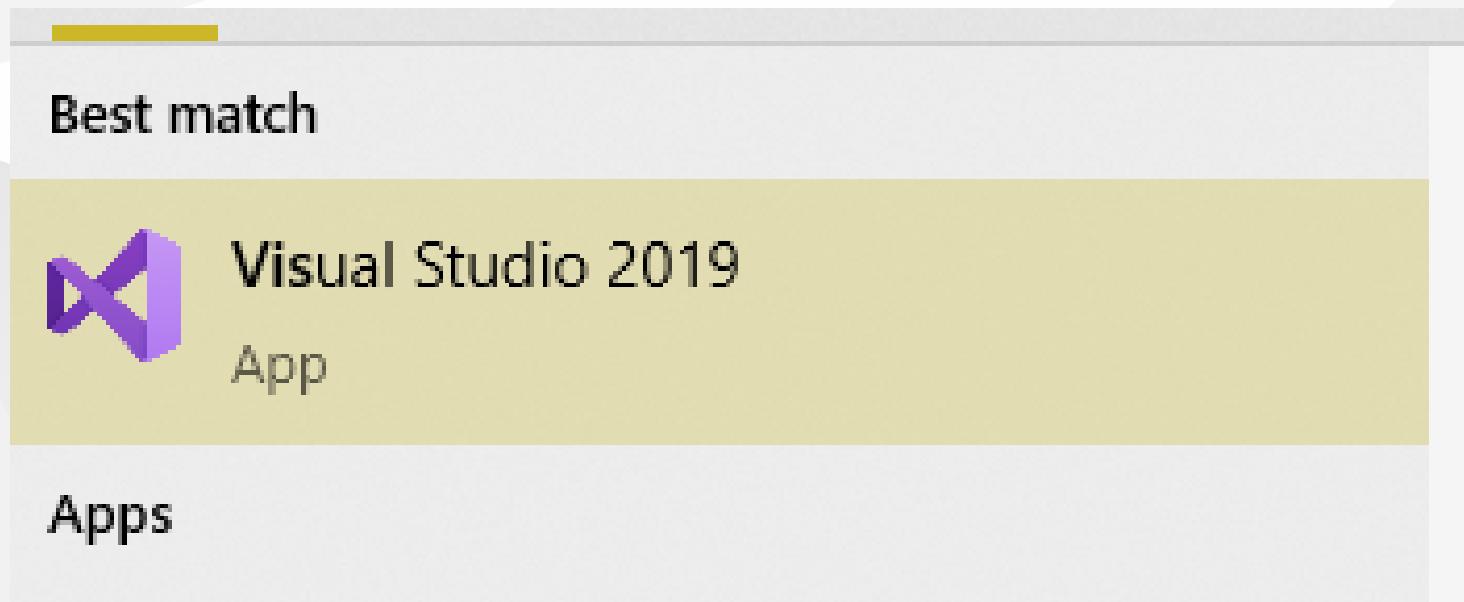
In this sample we will create **c-lib-sample** project that contains library, executable, unit tests and unit test runners.

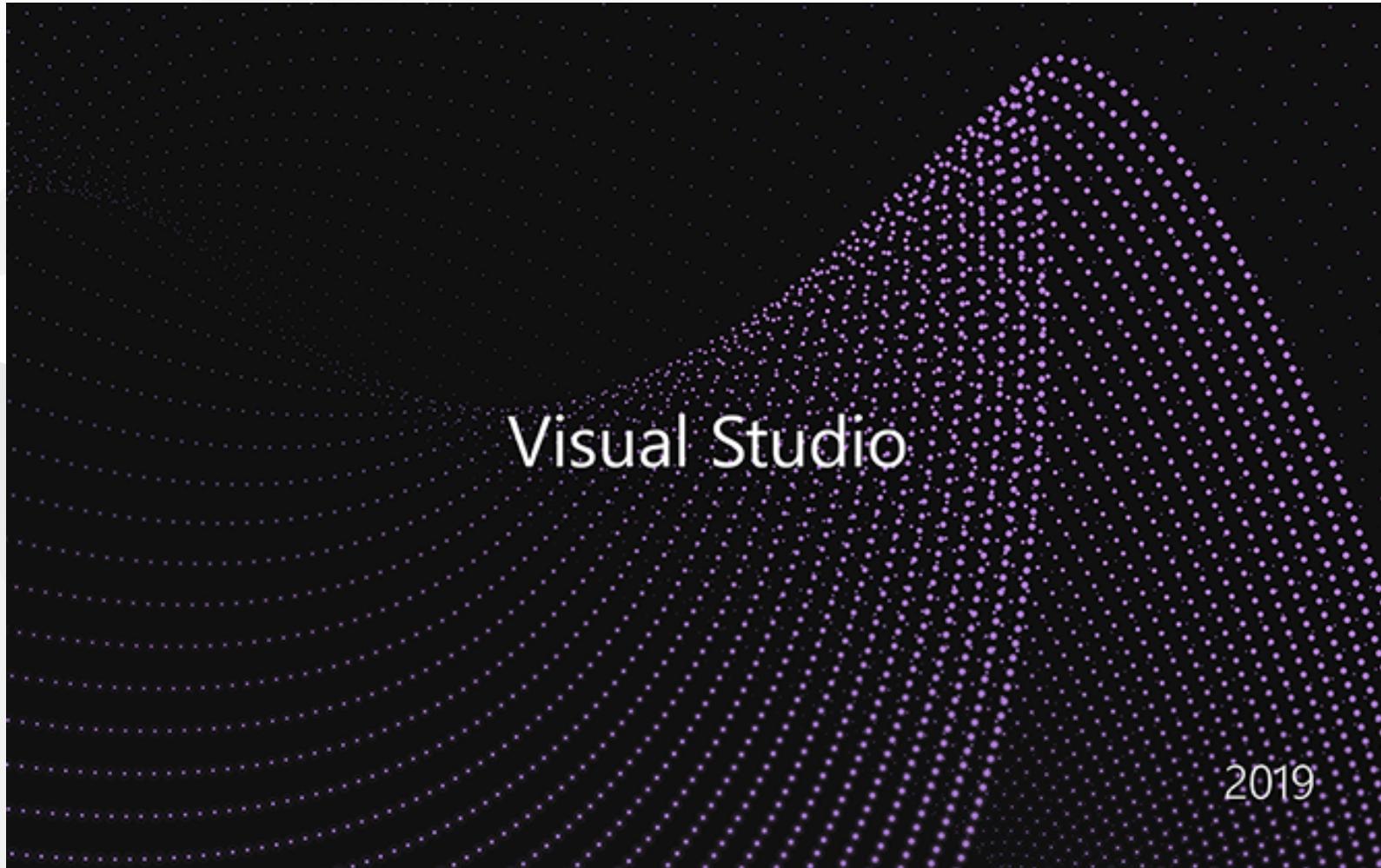
First of all you install Visual Studio Community Edition from website

[Visual Studio 2019 Community Edition - Son Ücretsiz Sürümü İndir](#)



Open visual studio community edition and select create a new project





## Select create a new project

### Get started



#### Clone a repository

Get code from an online repository like GitHub or Azure DevOps



#### Open a project or solution

Open a local Visual Studio project or .sln file



#### Open a local folder

Navigate and edit code within any folder

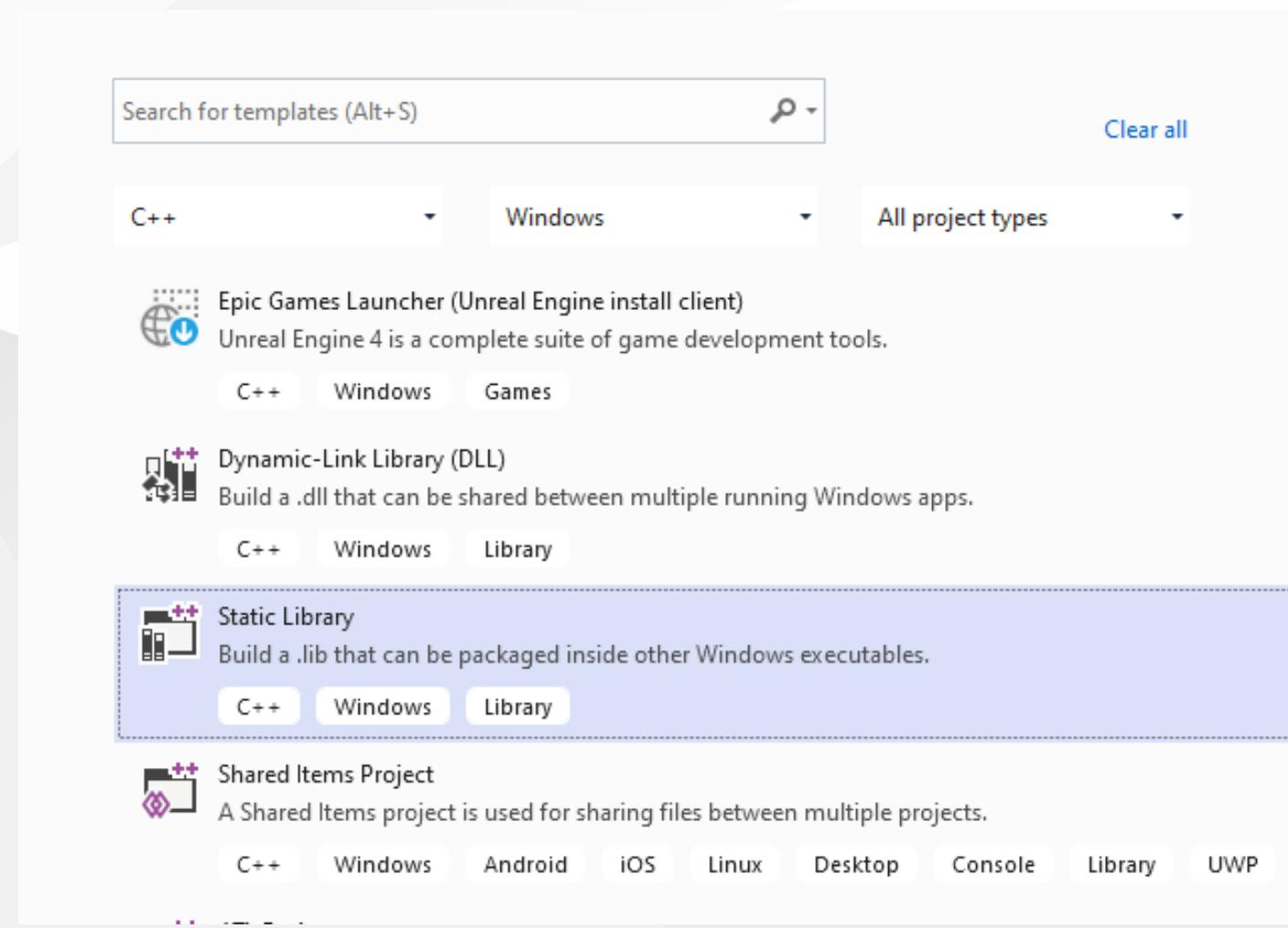


#### Create a new project

Choose a project template with code scaffolding to get started

[Continue without code →](#)

## Select C++ static library from project list



## Name static library project

### Configure your new project

Static Library   C++   Windows   Library

Project name

c-sample-lib

Location

E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming \Lectures\ce11

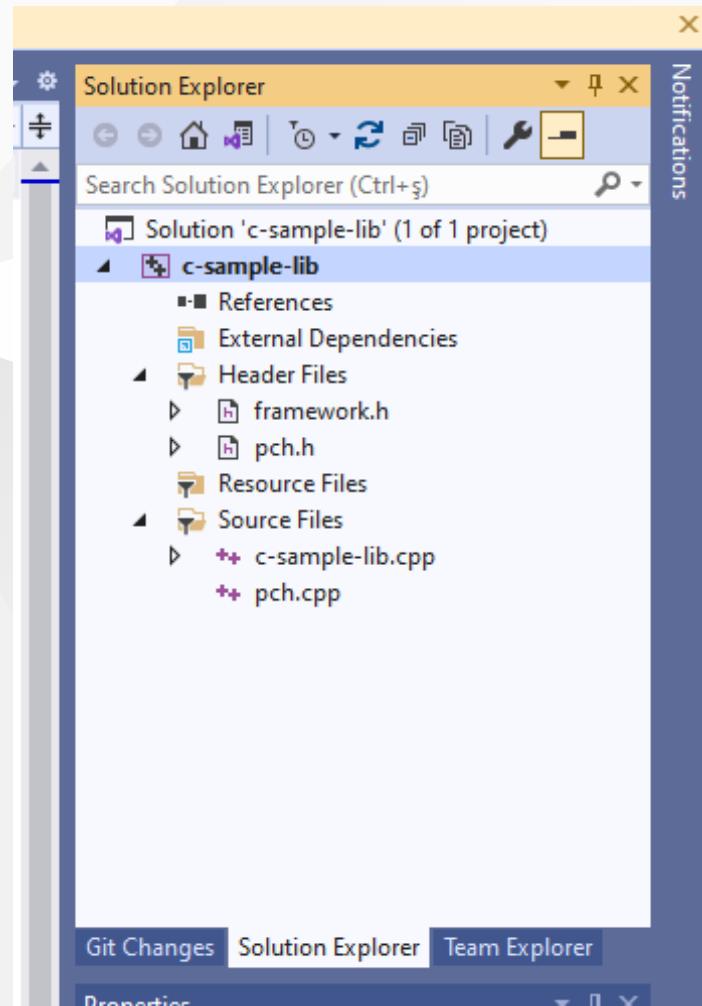
...

Solution name i

c-sample-lib

Place solution and project in the same directory

## Default configuration come with C++ project types and setting

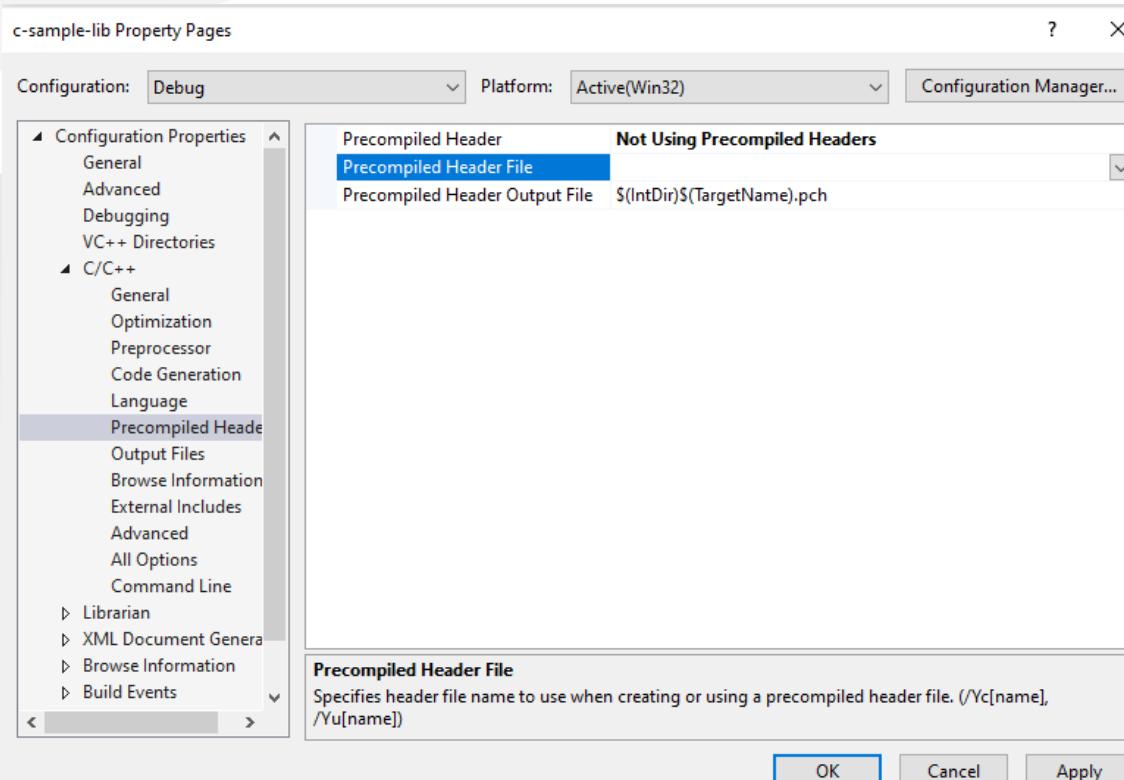


In the c-sample-lib.cpp you will sample function

```
void fnCSampleLib()  
{  
}  
}
```



Delete pch.h and pch.c files. Also disable use precompiled header settings from configurations and change to "Not Using Precomplied Headers", also you can delete precomplied Header File.



# Customize library header name and update "framework.h" to "samplelib.h"

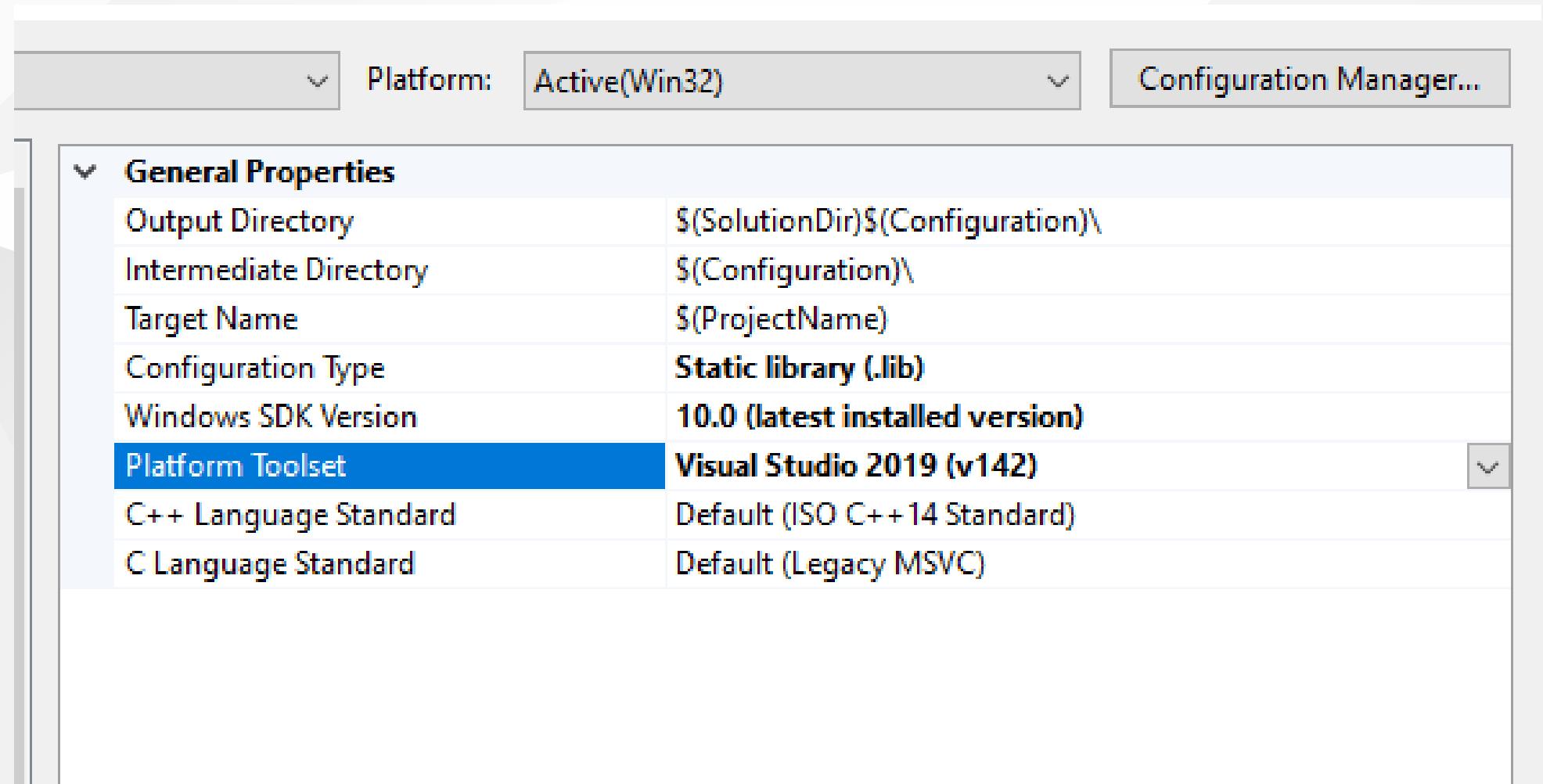
Insert your functions inside the c-sample-lib.c and update header files also.

```
// c-sample-lib.cpp : Defines the functions for the static library.  
//  
  
#include "samplelib.h"  
#include "stdio.h"  
  
/// <summary>  
///  
/// </summary>  
/// <param name="name"></param>  
void sayHelloTo(char* name){  
  
    if (name != NULL){  
        printf("Hello %s \n",name);  
    }  
    else {  
        printf("Hello There\n");  
    }  
}  
  
/// <summary>  
///  
/// </summary>  
/// <param name="a"></param>  
/// <param name="b"></param>  
/// <returns></returns>  
int sum(int a, int b){  
  
    int c = 0;  
    c = a + b;  
    return c;  
}
```

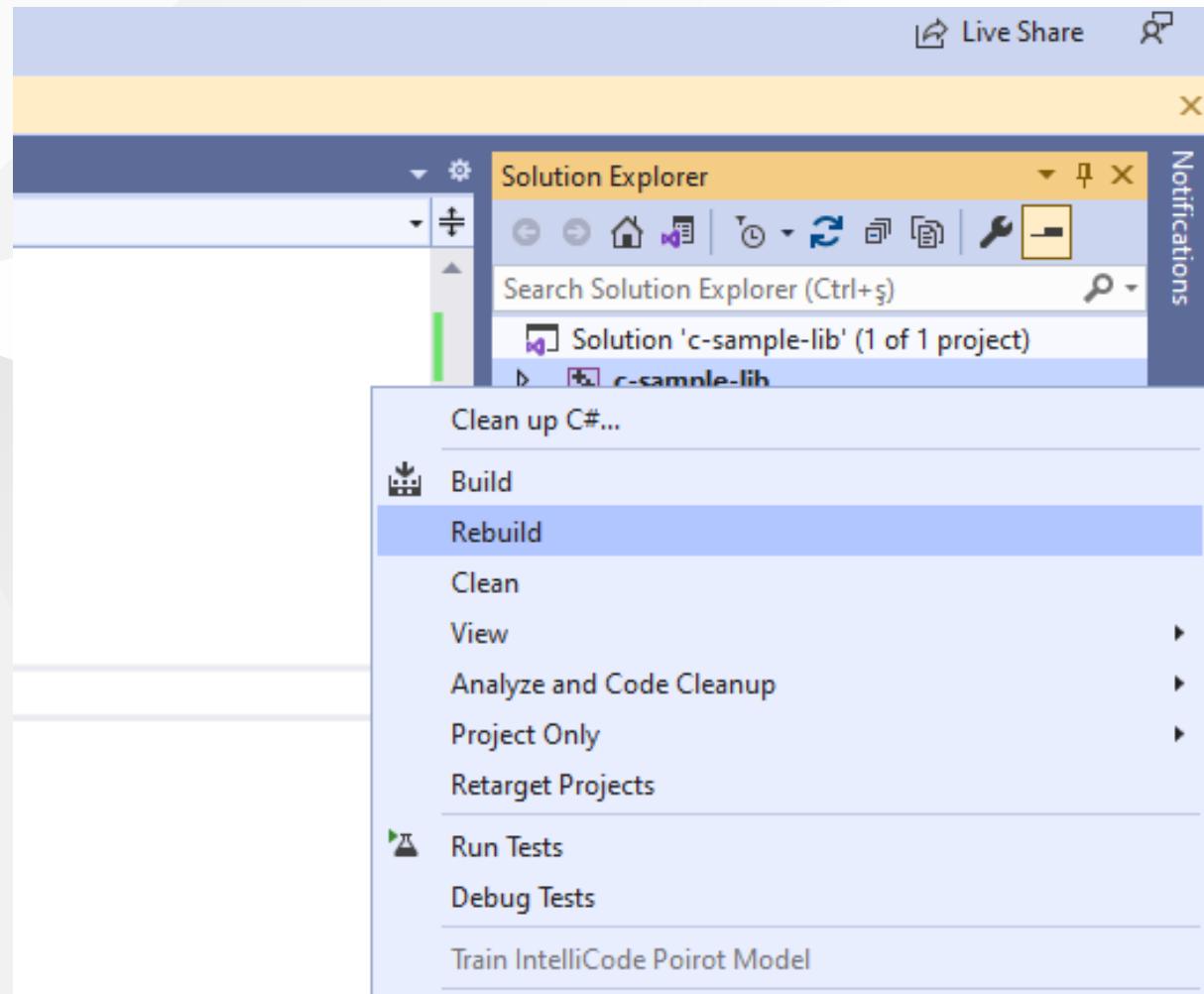
also update samplelib.h

```
#pragma once  
  
#define WIN32_LEAN_AND_MEAN // Exclude rarely-used stuff from Windows headers  
  
void sayHelloTo(char* name);  
int sum(int a, int b);
```

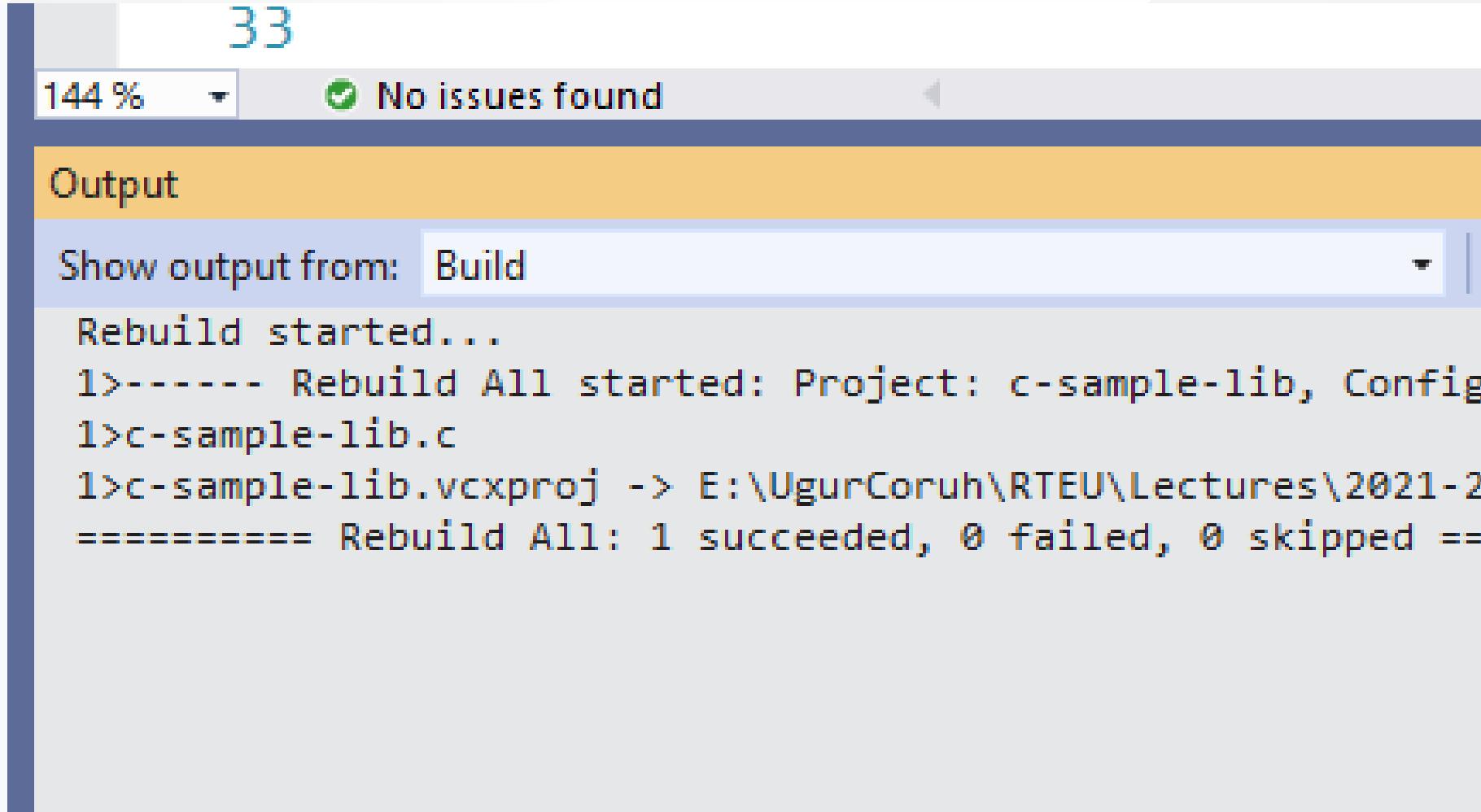
If you check configuration you will see that for C complier we are using Microsoft Environment and Toolkits



Now we can compile our library



You can follow operation from output window



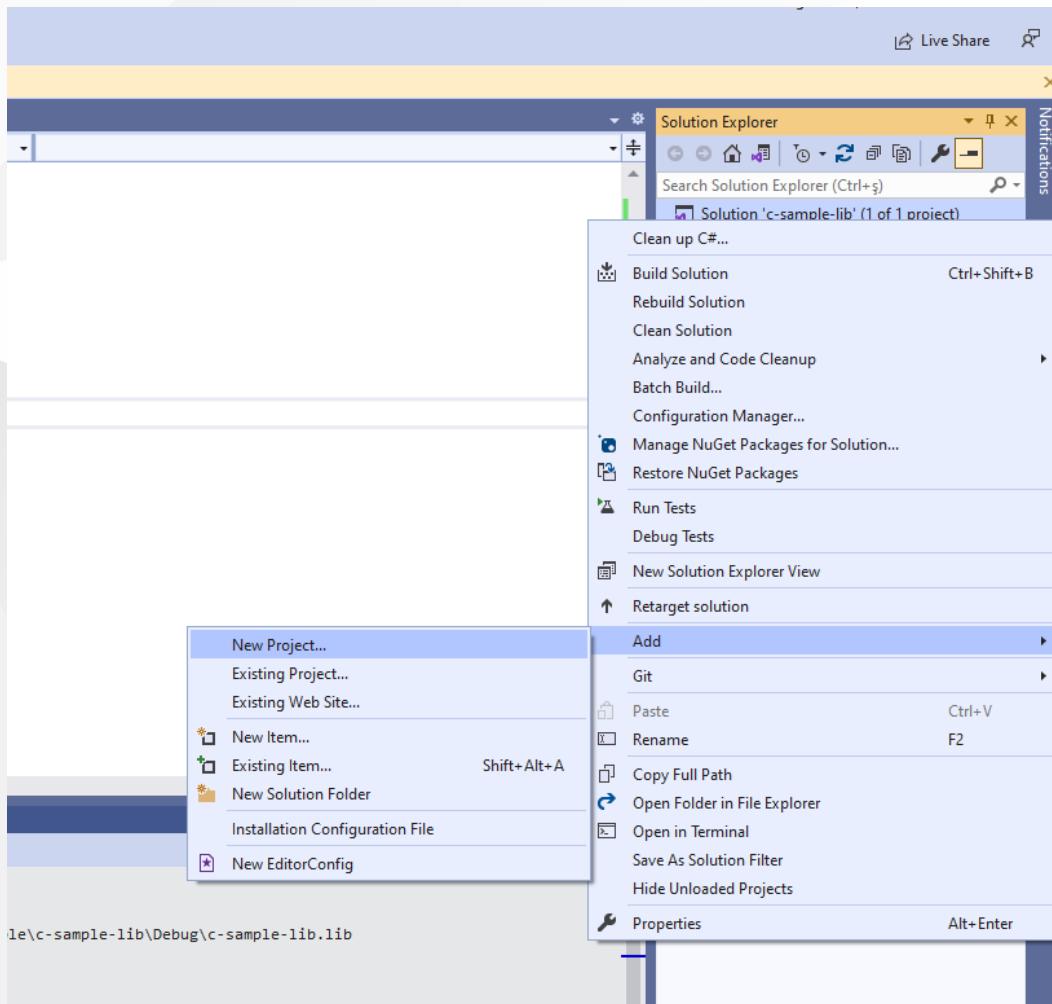
The screenshot shows a software interface with a dark blue header bar. In the top left corner of the header, the number '33' is displayed. To its right is a zoom control labeled '144 %' with a dropdown arrow. Next to it is a green circular icon with a checkmark and the text 'No issues found'. Below the header is a yellow rectangular area labeled 'Output' in black text. Underneath this is a white input field containing the text 'Show output from: Build' followed by a dropdown arrow. The main body of the window is a light gray area displaying a command-line style log:

```
Rebuild started...
1>----- Rebuild All started: Project: c-sample-lib, Configuration: Debug
1>c-sample-lib.c
1>c-sample-lib.vcxproj -> E:\UgurCoruh\RTEU\Lessons\2021-2
===== Rebuild All: 1 succeeded, 0 failed, 0 skipped ==
```

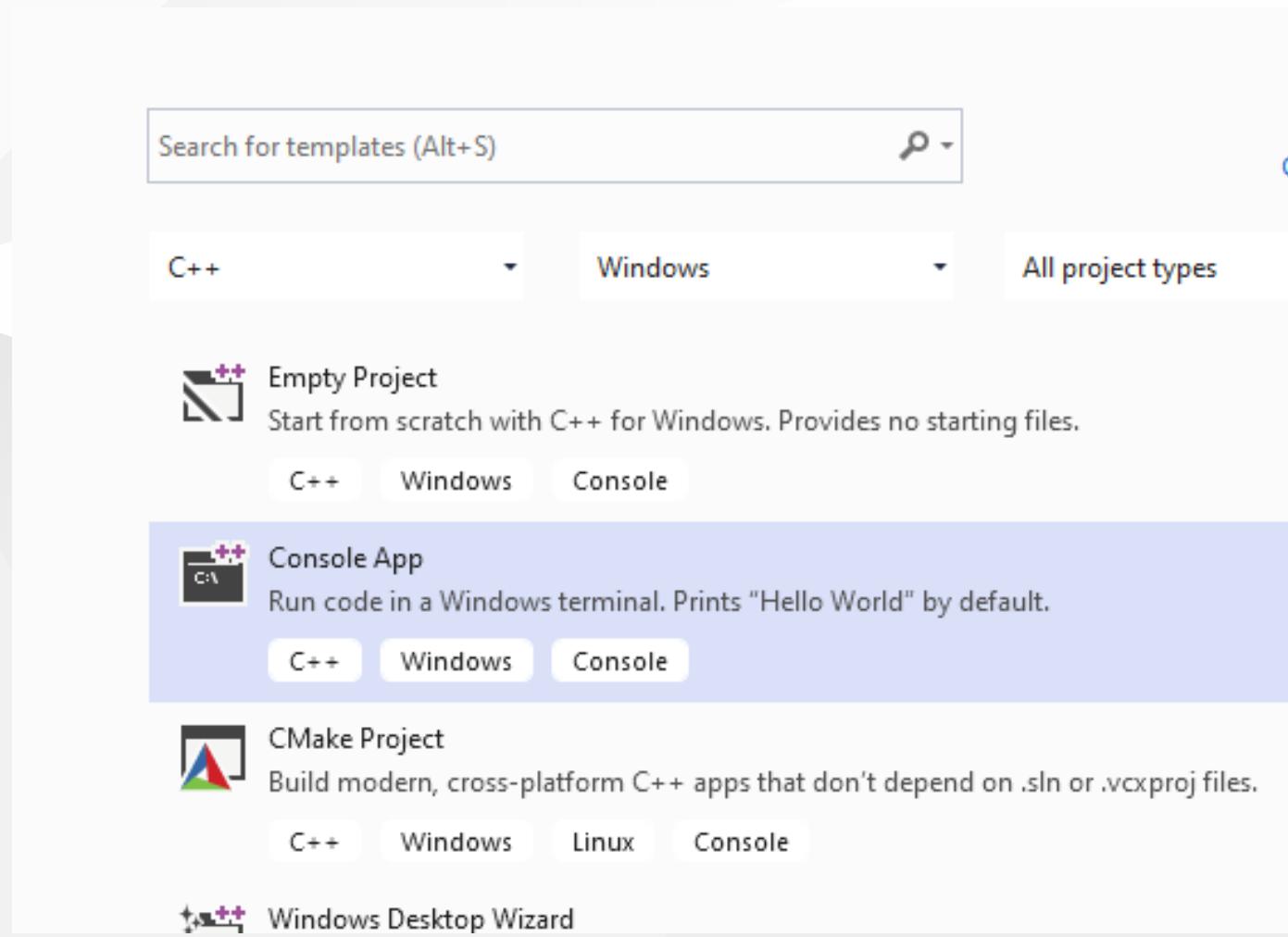
in debug folder we will see our output

Name
 <b>c-sample-lib.idb</b>
 <b>c-sample-lib.lib</b>
 <b>c-sample-lib.pdb</b>

now we will add a console application c-sample-app and use our library



## select C++ Windows Console Application from list



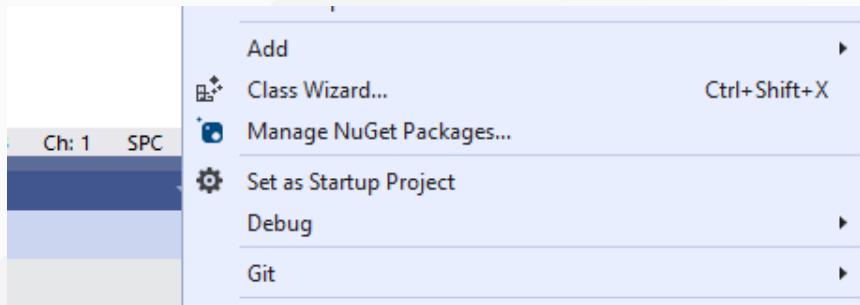
C++ Console Application Selection will generate a C++ console project we can change extension to C to compile our application as C application.

we will convert c-sample-app.c to following code

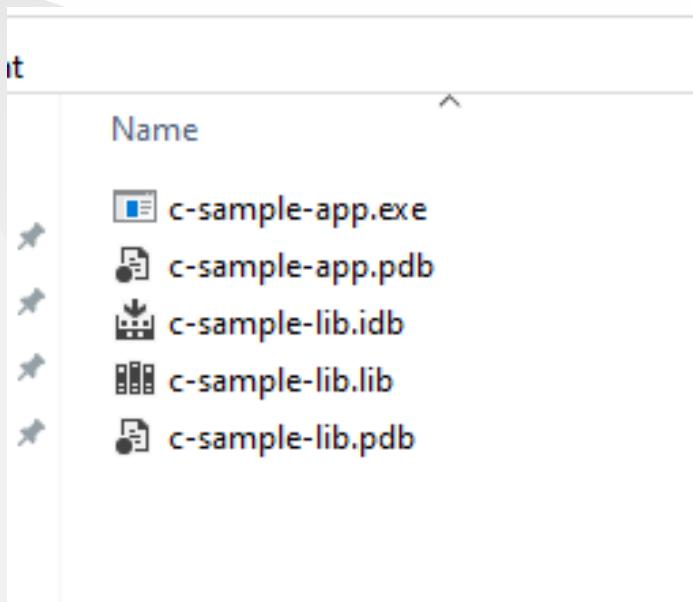
```
#include <stdio.h>

/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    printf("Hello World!\n");
}
```

after conversion set c-sample-app as startup project and build it



this will create c-sample-app.exe in the same folder with c-sample-lib.lib library

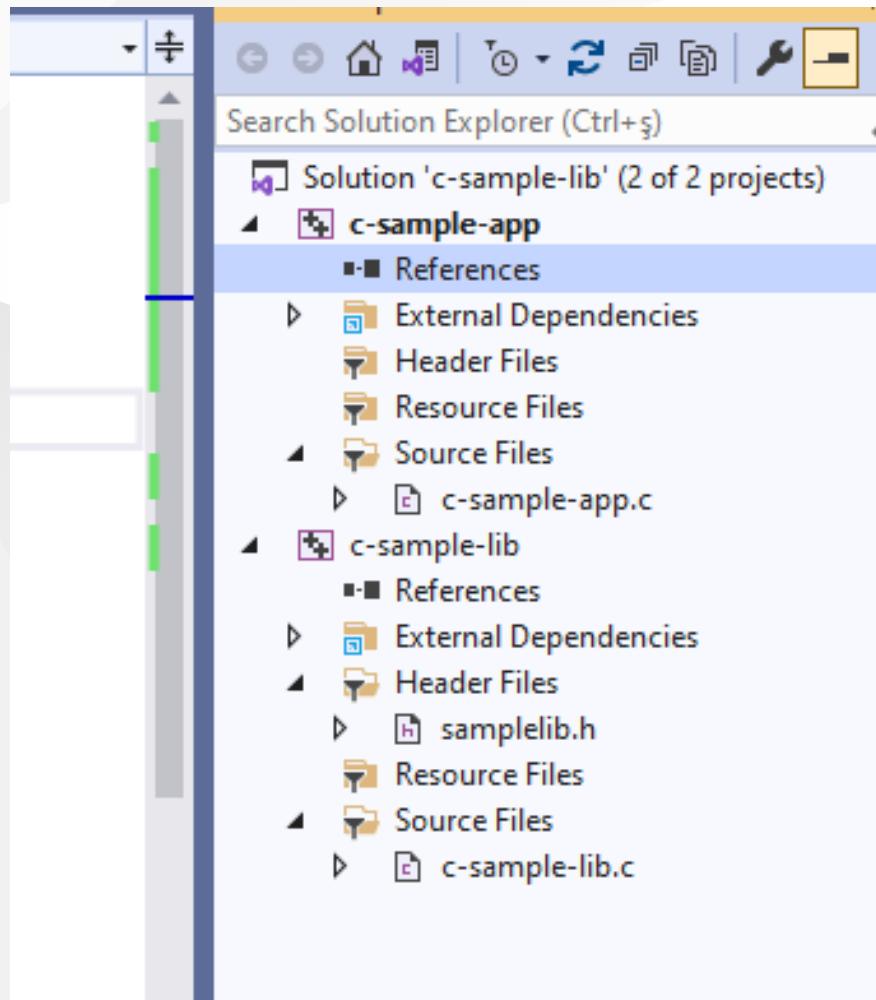


if we run application we will see only "Hello World"

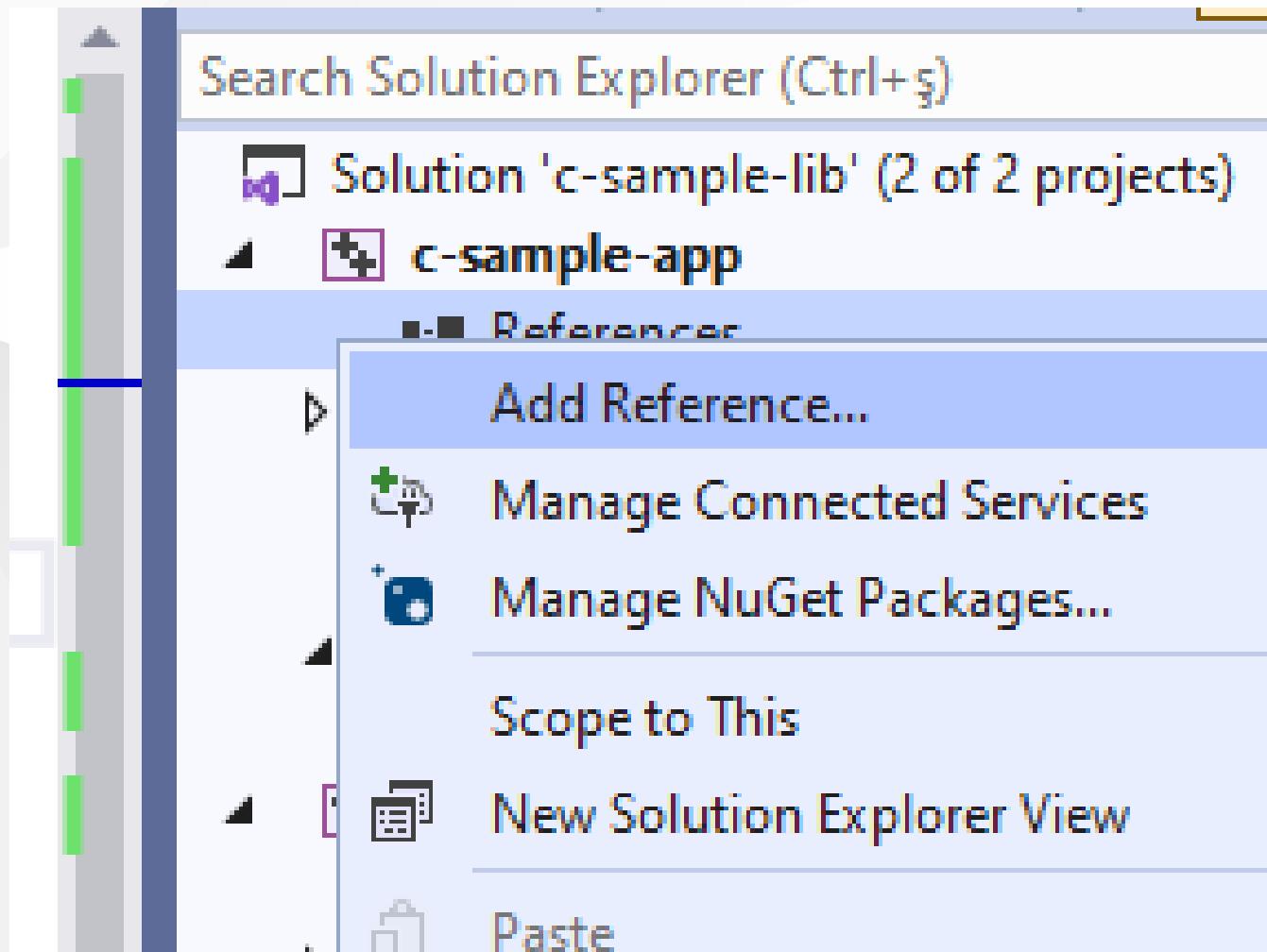
now we will see two options to add library as references in our application and use its functions.

## First option

right click references for c-sample-app and add current library as reference



## Select Add Reference



Browse for solution and select c-sample-lib

Add Reference

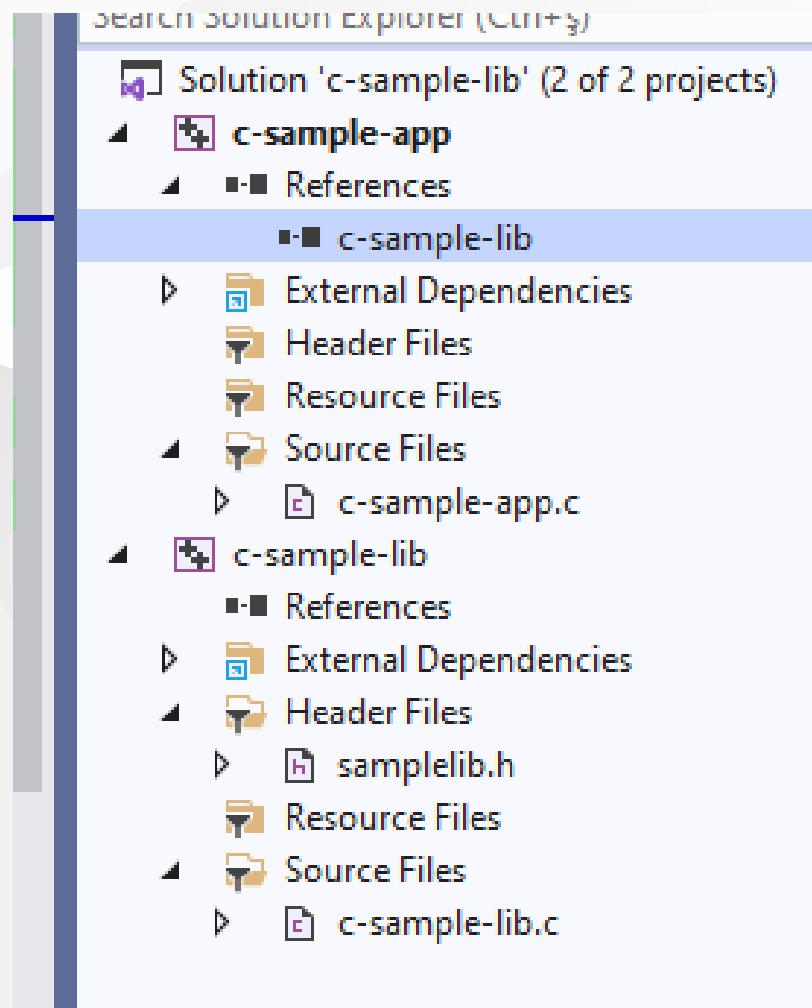
Projects

Solution

Shared Projects

	Name	Path
<input checked="" type="checkbox"/>	c-sample-lib	E:\UgurCoruh\

You can check added reference from references section



now we can include required headers from c-sample-lib folder and use it.

we can include required header with relative path as follow or with configuration

```
#include <stdio.h>
#include "..\c-sample-lib\samplelib.h"
/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    printf("Hello World!\n");
}
```

we can build our c-sample-app

The screenshot shows a terminal window with the following output:

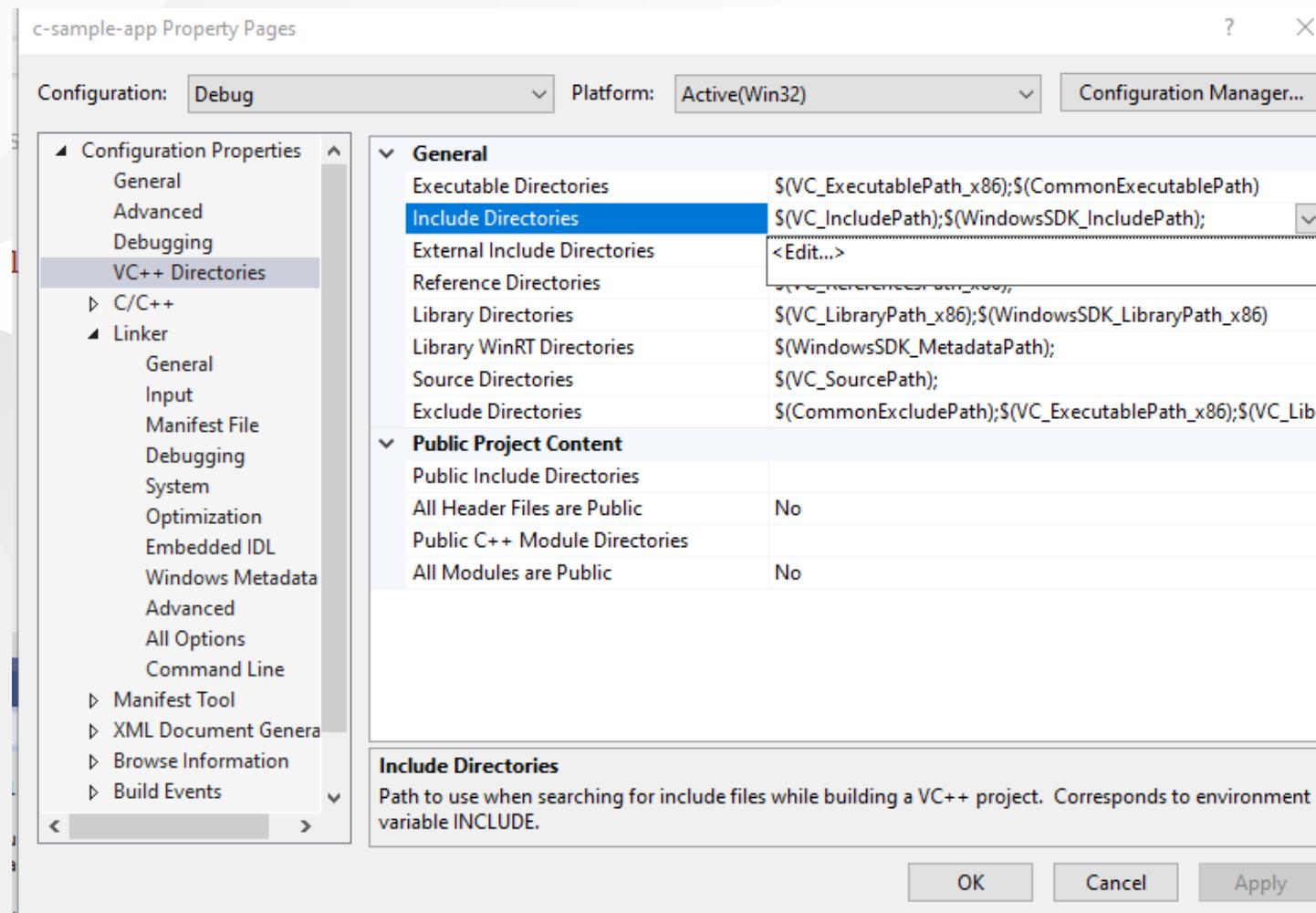
```
144 %  No issues found
Output
Show output from: Build
Rebuild started...
1>----- Rebuild All started: Project: c-sample-lib, Configuration: Debug Win32
1>c-sample-lib.c
1>c-sample-lib.vcxproj -> E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algor
2>----- Rebuild All started: Project: c-sample-app, Configuration: Debug Win32
2>c-sample-app.c
2>c-sample-app.vcxproj -> E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algor
===== Rebuild All: 2 succeeded, 0 failed, 0 skipped =====
```

also we can only write header name

```
#include <samplelib.h>
```



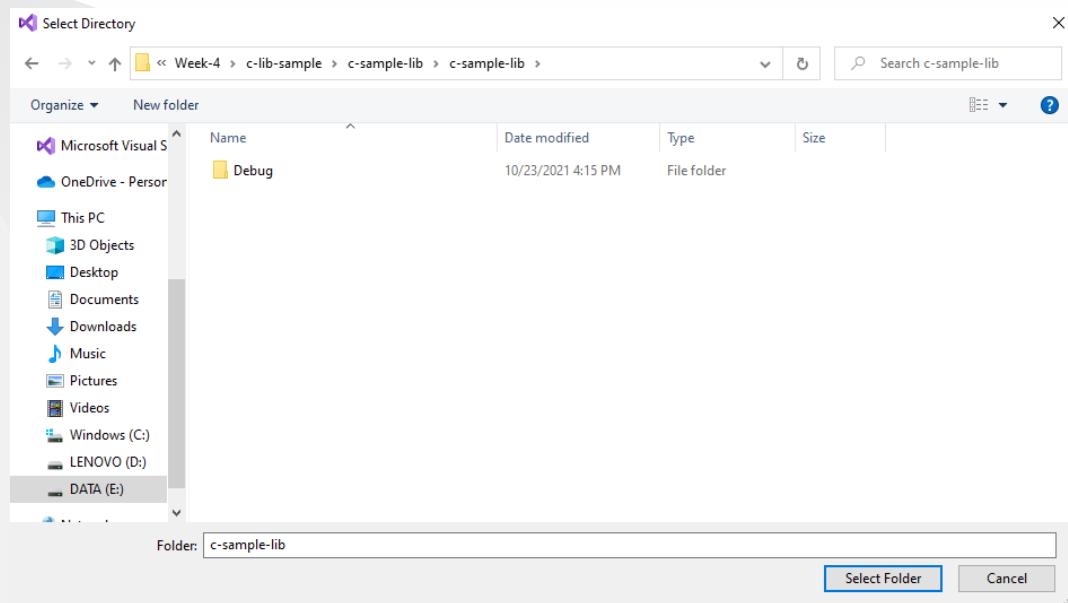
for this we need to configure include directories



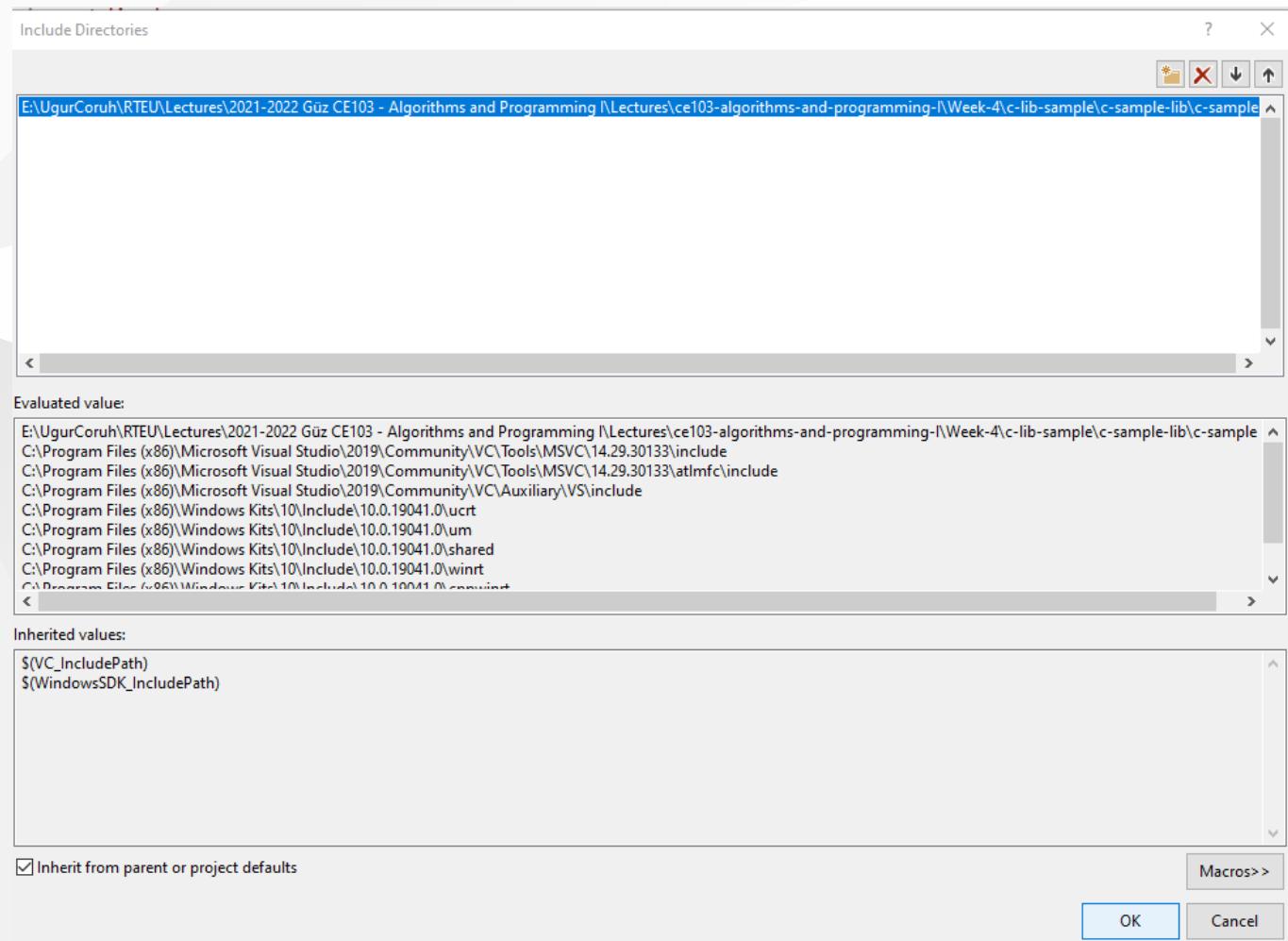
## select c-sample-lib header file location



browse for folder



your full path will be added to your configuration

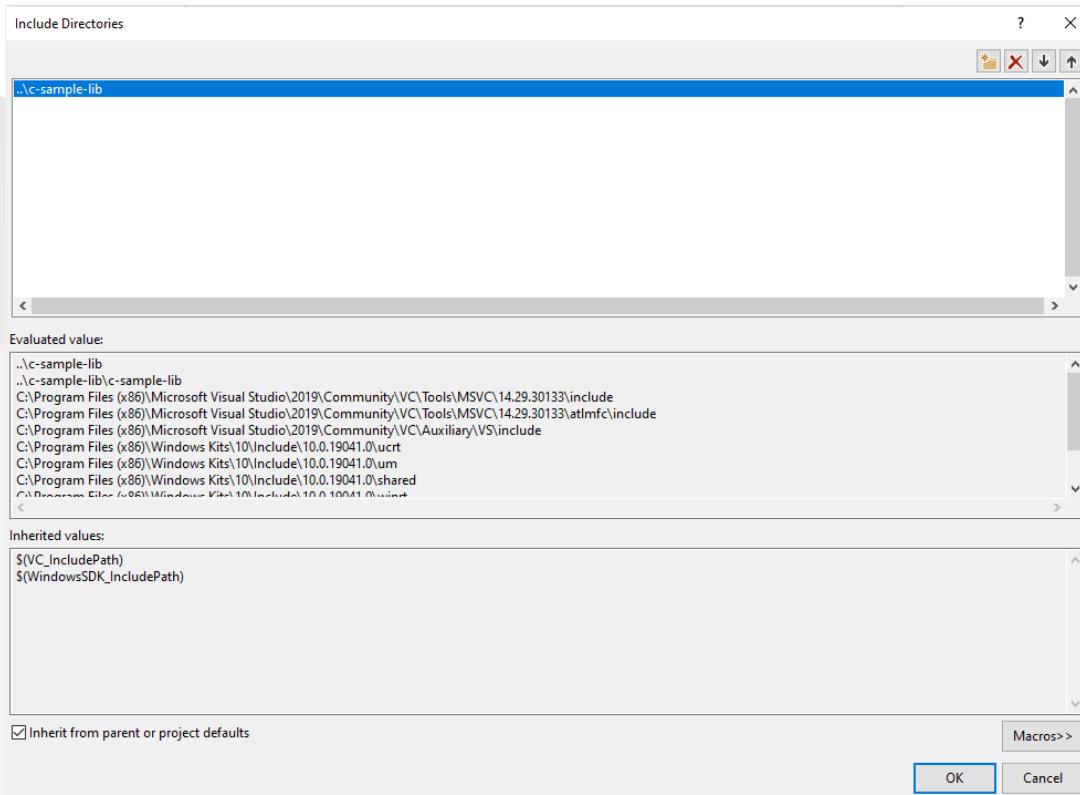


if you add header file paths to your configuration you can use header files by name in your source code

```
#include <stdio.h>
#include <samplelib.h>
/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    printf("Hello World!\n");
}
```

we can compile the following we don't have problems but here we need to configure relative paths for configuration open include library settings and update with relative path

.. \c-sample-lib



now we have portable source code configuration. we can call our functions and then we can update header and library folder configurations.

```
#include <stdio.h>
#include <samplelib.h>
/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    int result = 0;
    //printf("Hello World!\n");
    result = sum(5, 4);
    sayHelloTo("Computer");
    printf("Result is %d \n",result);
    printf("Press any key to continue...\n");
    getchar();
    return 0;
}
```



when you run you will see the following outputs, that mean we called library functions.



A screenshot of a terminal window titled "E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming \Lectures\ce103-algorithms-and-programming-I\Week-4\c-lib-sa...". The window contains the following text:  
Hello Computer  
Result is 9  
Press any key to continue...

static library is a code sharing approach if you want to share your source code with your customers then you can share static libraries and header files together. Another case you can use a precompiled static library with you or this library can be part of any installation then if there is a installed app and static libraries are placed on system folder or any different location then you can use configuration files to set library path and included header paths

Now we can remove project from c-sample-app references but we will set library file in configuration

Before this copy static library and header files to a folder like that

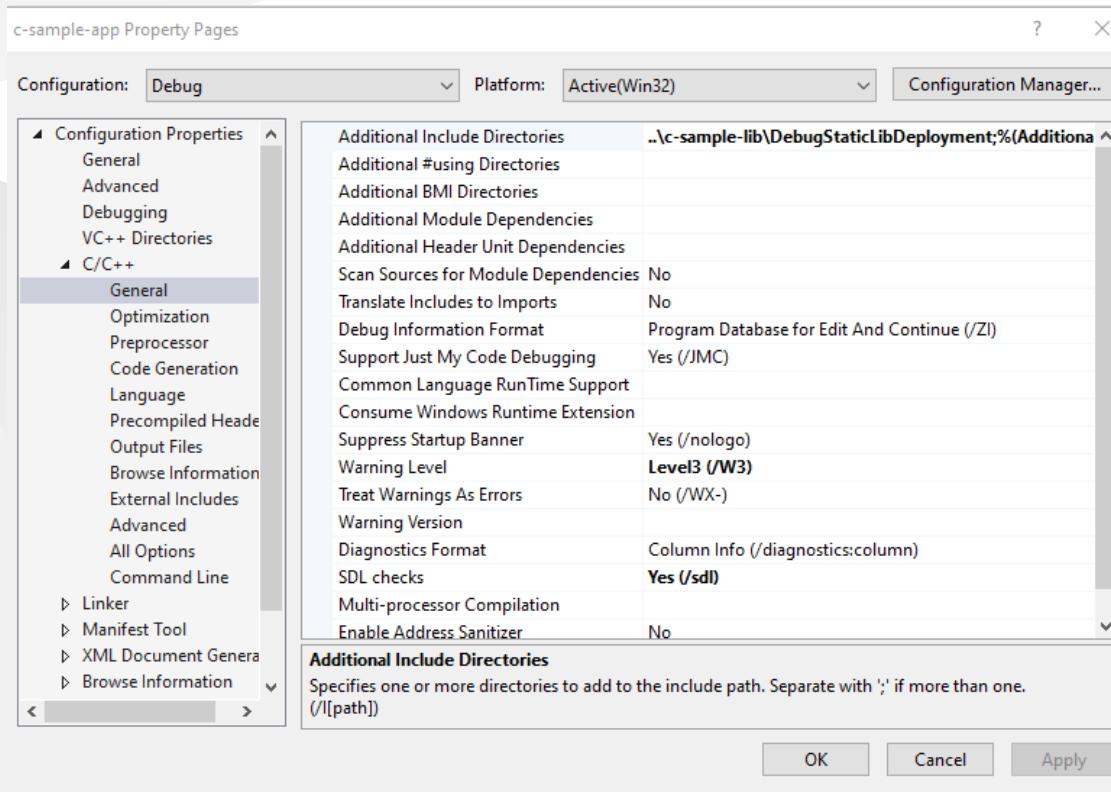
#### DebugStaticLibDeployment

- Set C/C++ -> General -> Additional Include Directories

There is a bug in configurations and relative path not finding headers so for this reason we will set full path but this is not a good practice for team working

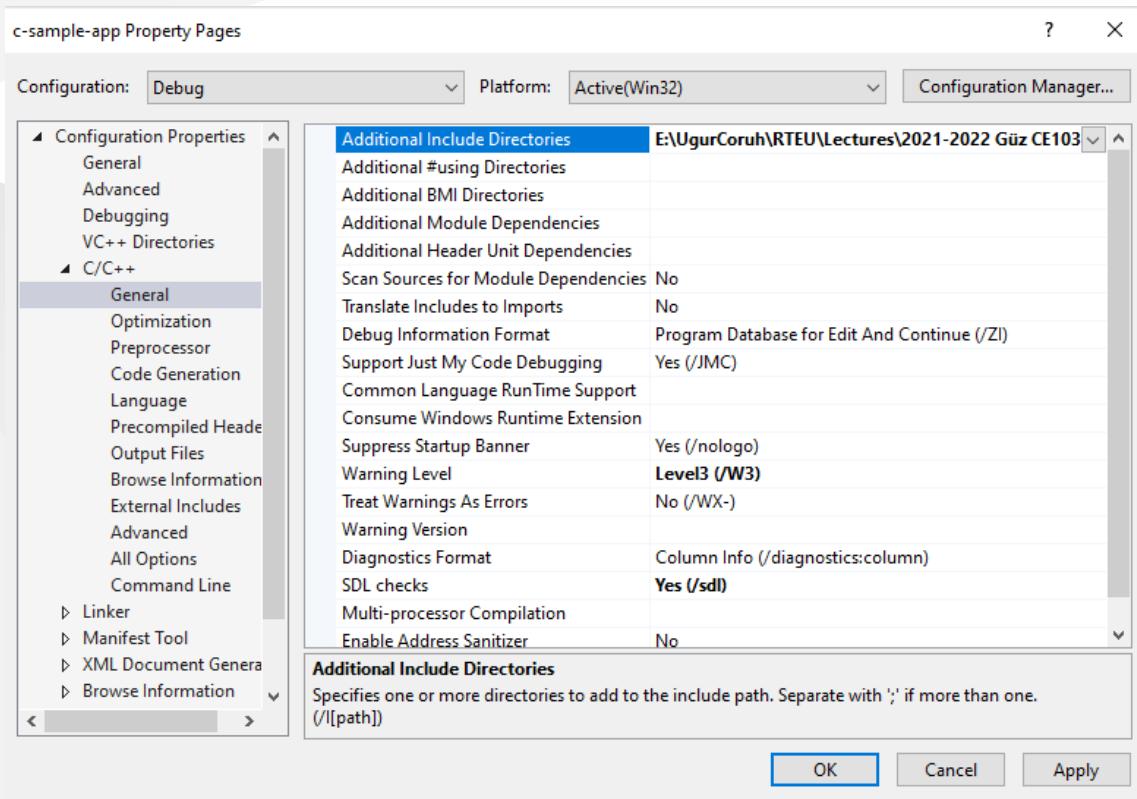
# Not Working

..\c-sample-lib\DebugStaticLibDeployment



# Working

E:\....\c-lib-sample\c-sample-lib\DebugStaticLibDeployment



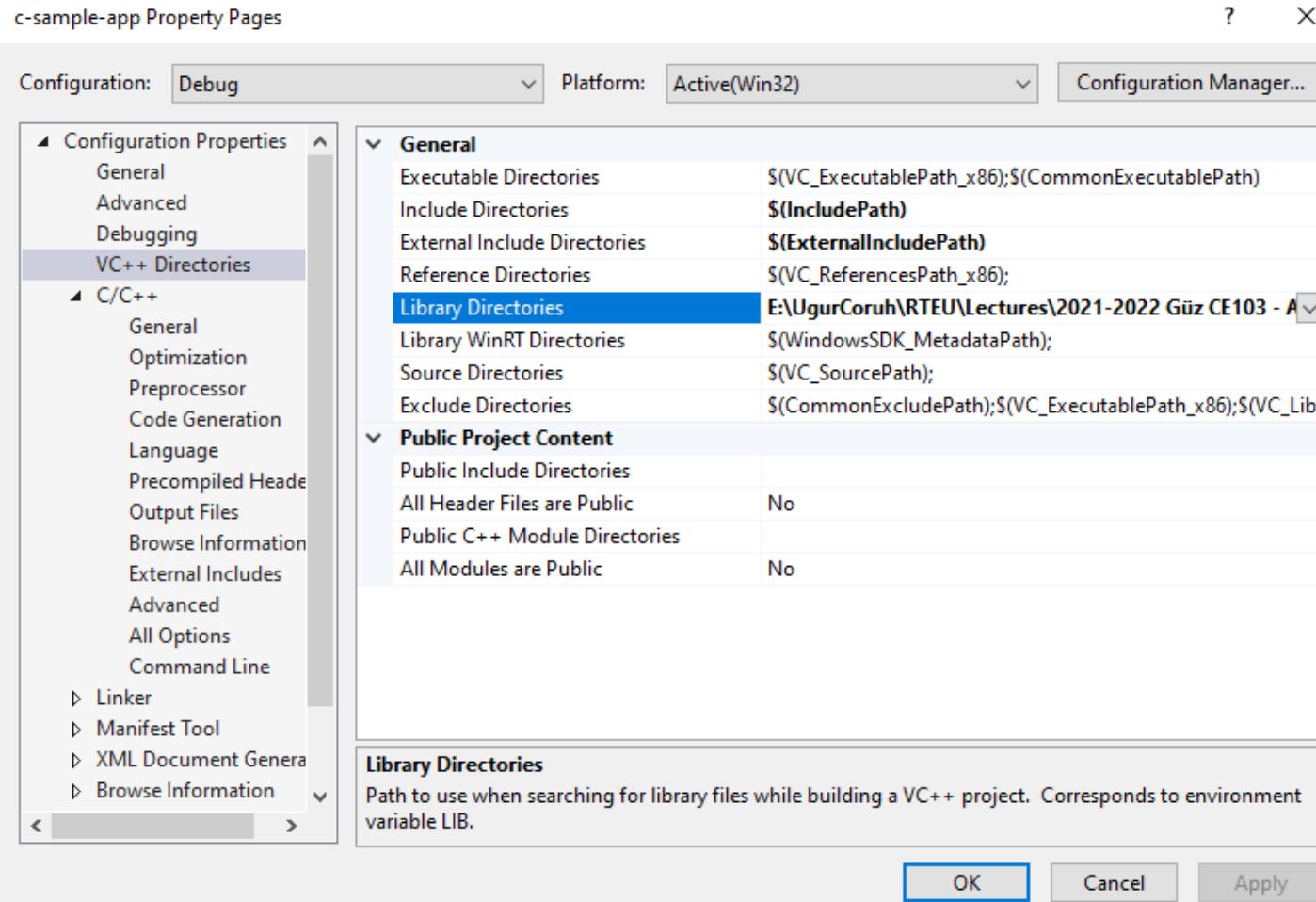
Now we will set library folder that our static library placed

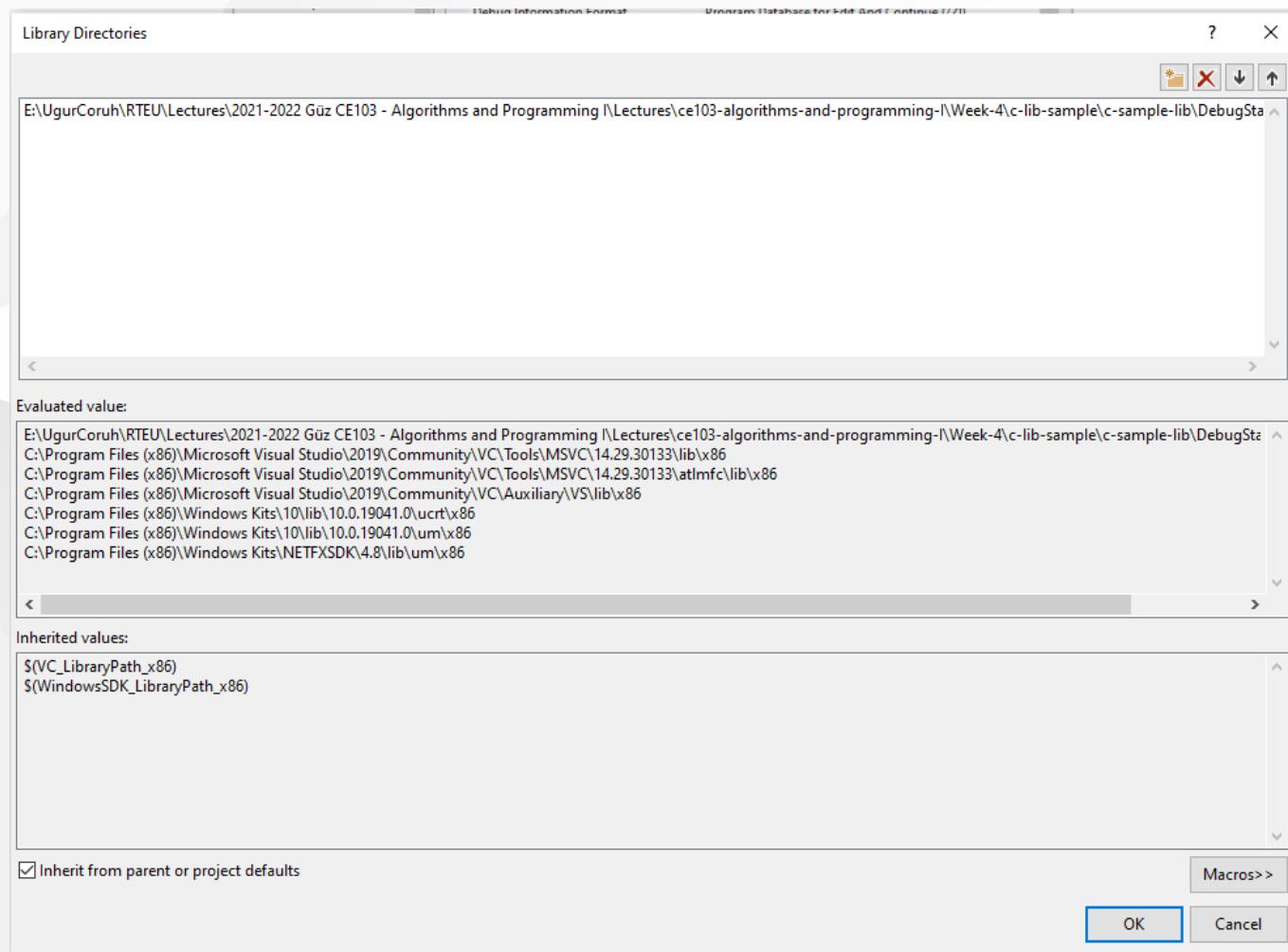
we will set VC++ Directories -> Library Directories

Here is the same issue if we use relative path it doesn't work we need to set full path for library folder

# Working

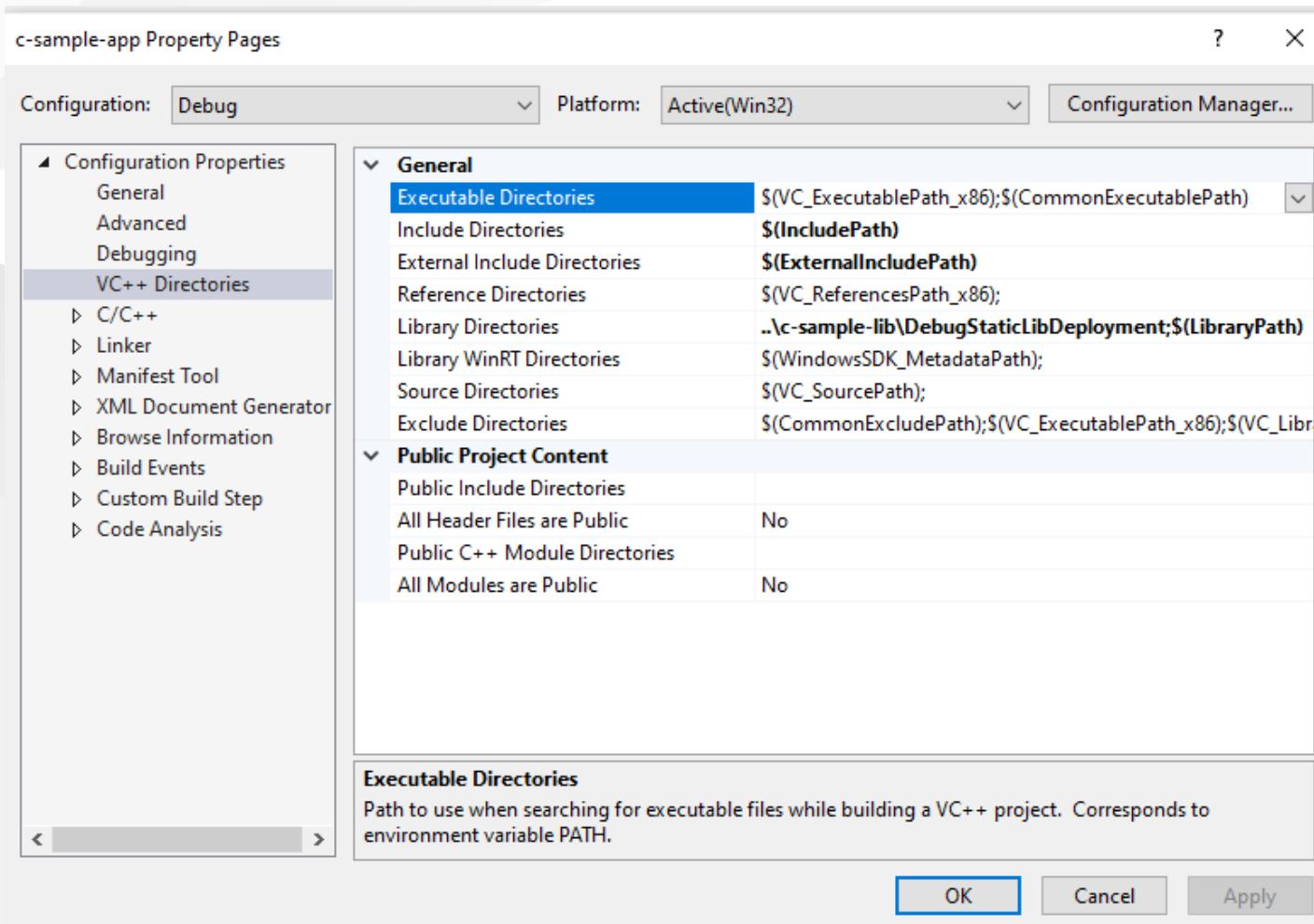
E:\...\c-lib-sample\c-sample-lib\DebugStaticLibDeployment

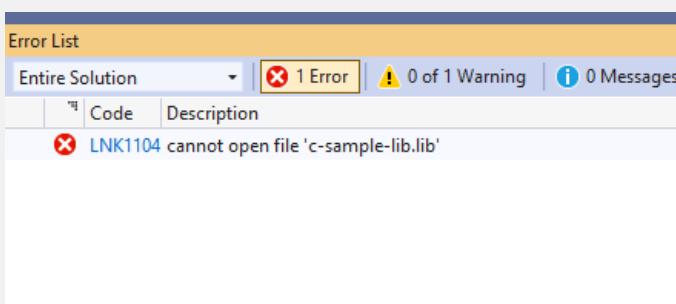
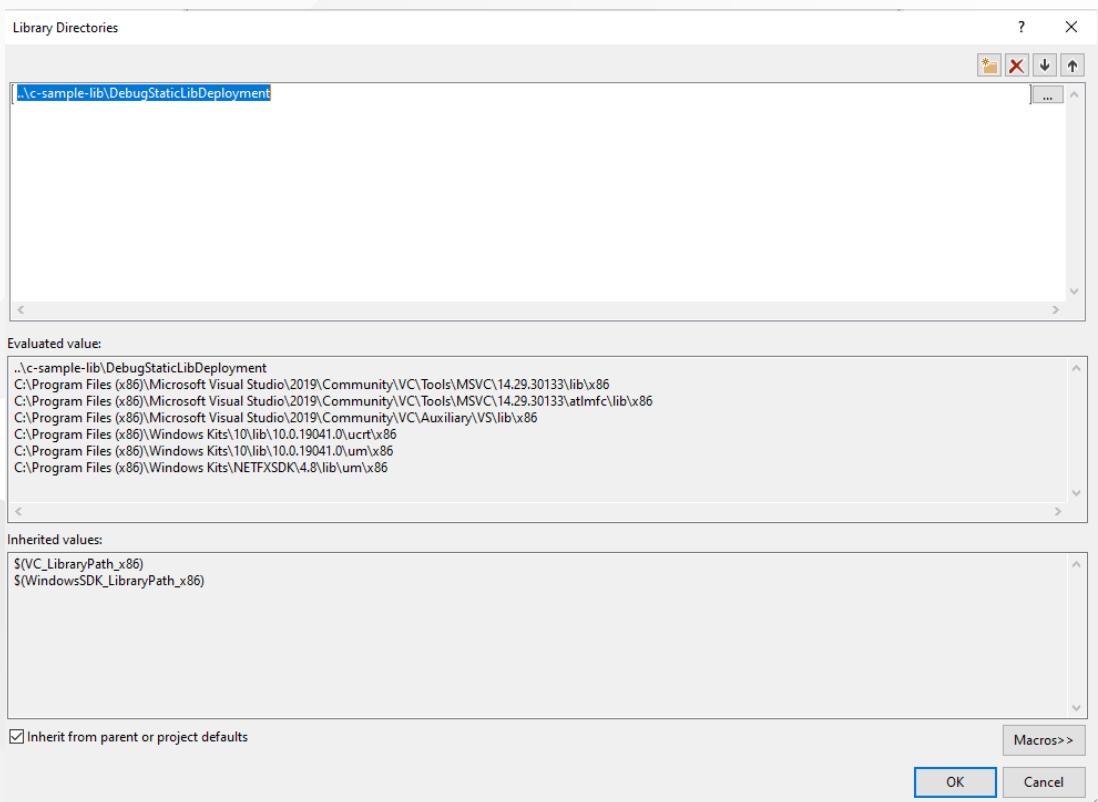




# Not Working

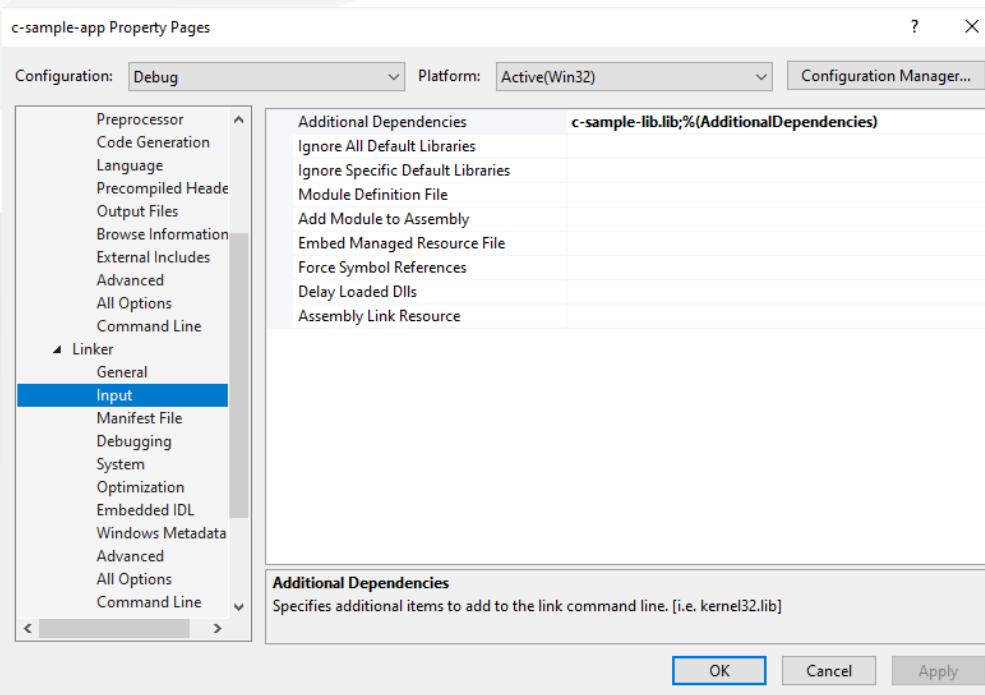
..\c-sample-lib\DebugStaticLibDeployment





If we set full path for both libraries and headers then we need to set library name for project

## Linker->Input->Additional Dependencies



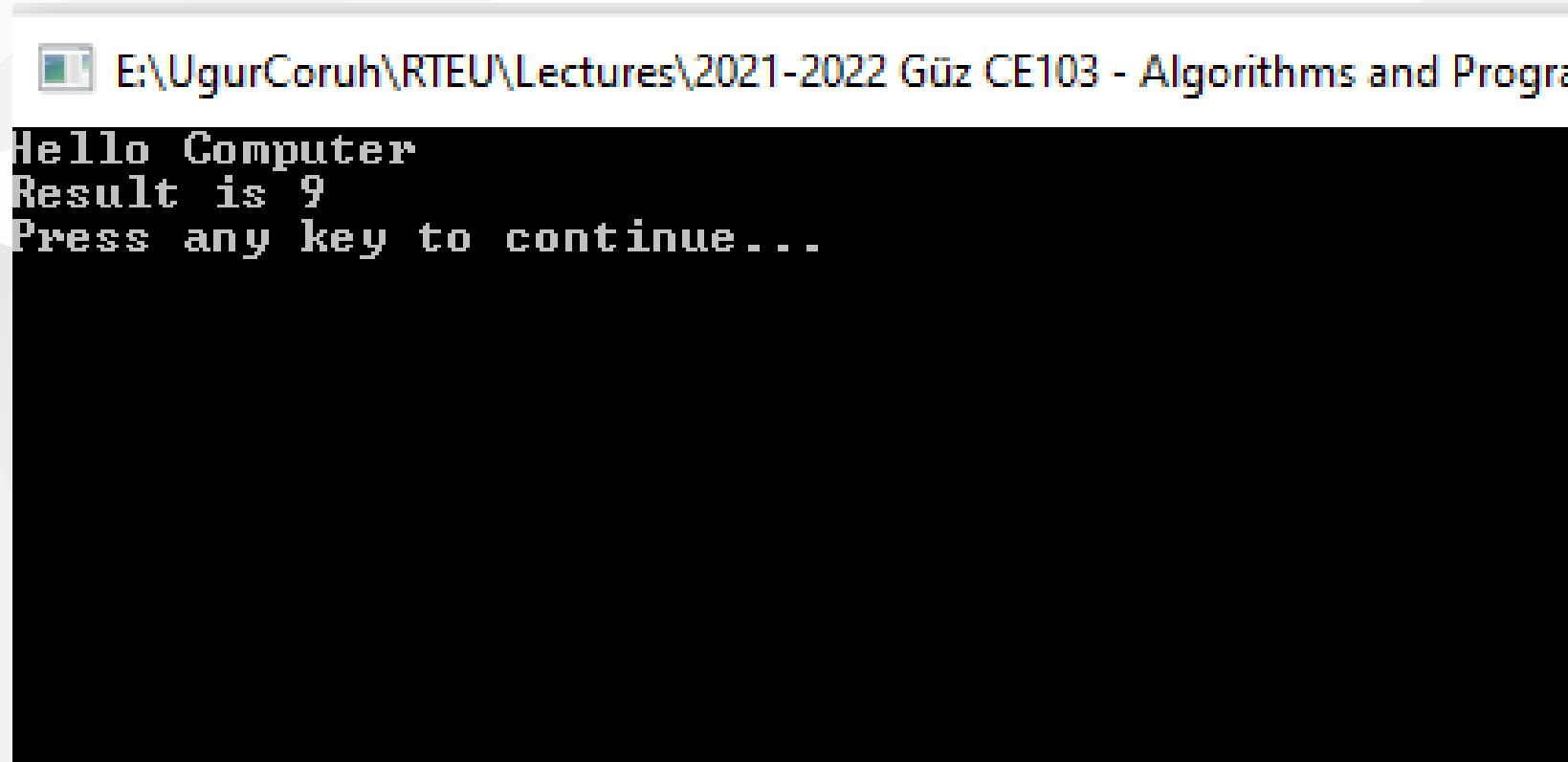
In this case we will compile c-sample-app and we do not need to compile c-sample-lib because we copied output files to different location and they are ready to use.

current source code will be like that nothing changed

```
#include <stdio.h>
#include <samplelib.h>

/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    int result = 0;
    //printf("Hello World!\n");
    result = sum(5, 4);
    sayHelloTo("Computer");
    printf("Result is %d \n",result);
    printf("Press any key to continue... \n");
    getchar();
    return 0;
}
```

and output



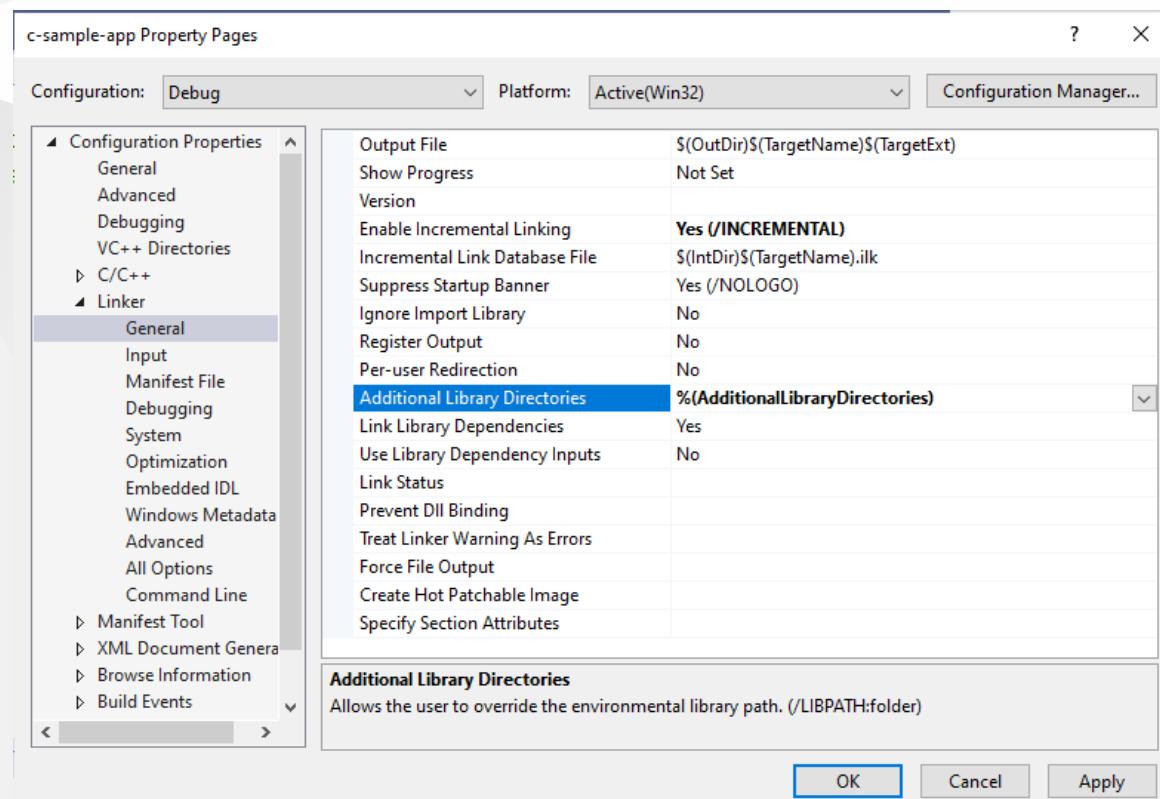
A screenshot of a terminal window with a black background and white text. The window title bar at the top left shows a small blue icon followed by the path "E:\UgurCoruh\RTEUN\Lectures\2021-2022 Güz CE103 - Algorithms and Program". The main content of the window displays three lines of text: "Hello Computer", "Result is 9", and "Press any key to continue...".

```
Hello Computer
Result is 9
Press any key to continue...
```

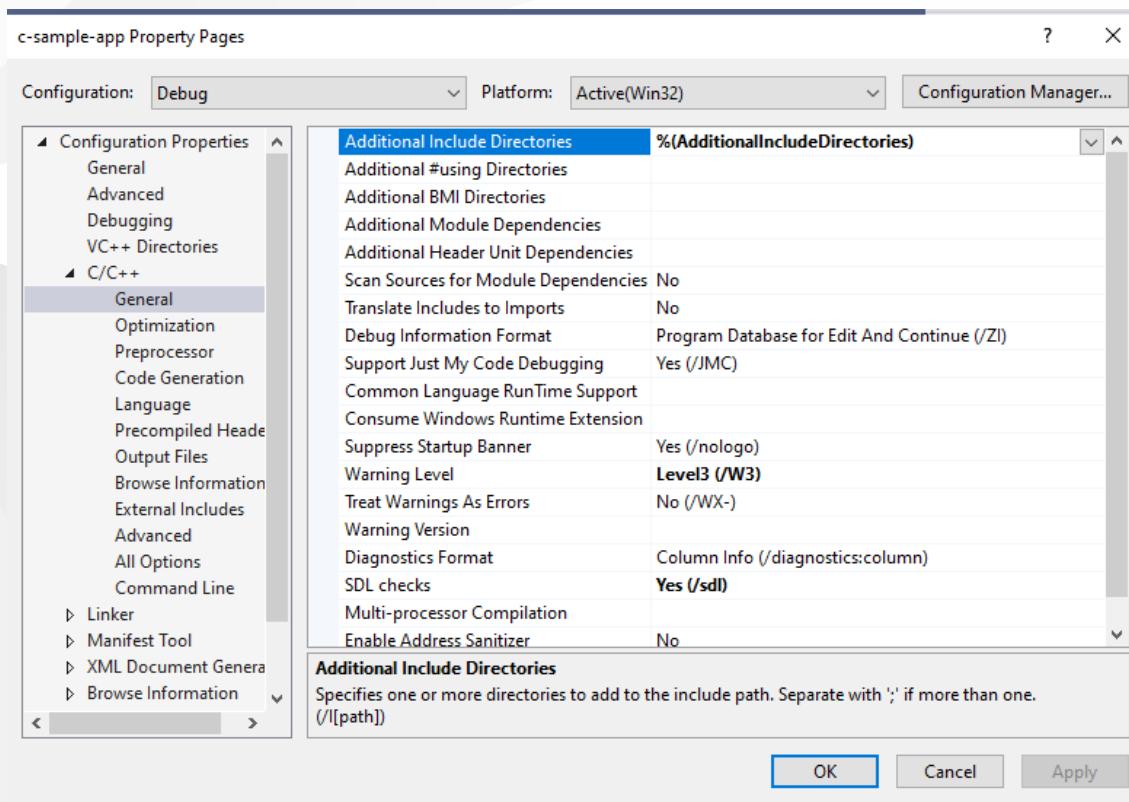
There is a option about portability that we can set for team works

We will remove all library related settings from configurations and we will write them in source code

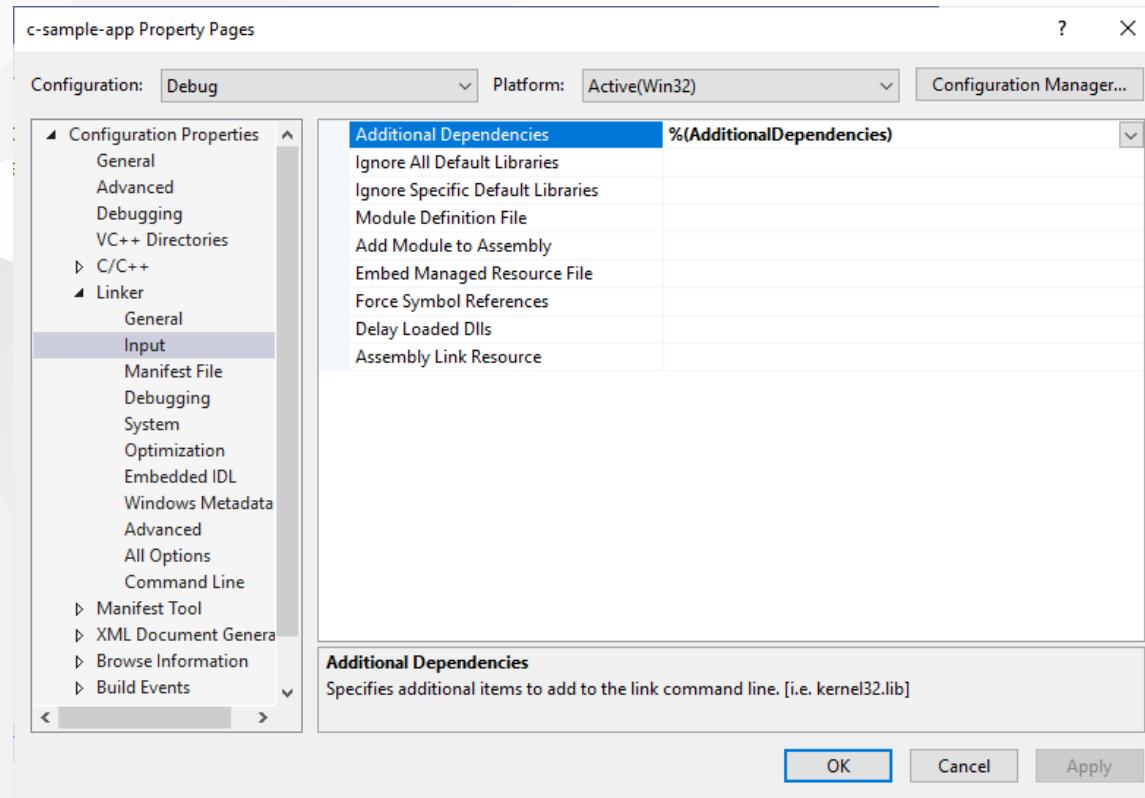
Clear linker->general->additional library directories



## Clear C/C++ -> General -> Additional Include Directories



# Clear Linker->Input->Additional Dependencies



# Now we can set this configurations in source code as follow

CE103 Algorithms and Programming I

```
#pragma comment(lib, "..\\DebugStaticLibDeployment\\c-sample-lib.lib")
#include "..\\DebugStaticLibDeployment\\samplelib.h"

#include <stdio.h>

/// <summary>
///
/// </summary>
/// <returns></returns>
int main()
{
    int result = 0;
    //printf("Hello World!\n");
    result = sum(5, 4);
    sayHelloTo("Computer");
    printf("Result is %d \n",result);
    printf("Press any key to continue... \n");
    getchar();
    return 0;
}
```



with this configuration if your friends download this code then they can run them with their environment without setting a path.

RTEU CE103 Week-4

## C++ Programming (Static Library)

## Visual Studio Community Edition

All steps are similar with C programming above, but you do not need to delete pch.h

You should take care about compiled source codes

for example if your code is compiled for x86 then your application also should use the x86 configuration else x64 then library should be x64 complied version.

## Source will look like the following

```
// cpp-sample-app.cpp : This file contains the 'main' function. Program execution begins and ends there.  
//  
  
#pragma comment(lib, "..\DebugStaticLibDeployment\cpp-sample-lib.lib")  
  
#include "..\DebugStaticLibDeployment\samplelib.h"  
  
#include <iostream>  
  
int main()  
{  
    std::cout << "Hello World!\n";  
  
    int result = 0;  
    //printf("Hello World!\n");  
    result = sum(5, 4);  
    sayHelloTo("Computer");  
    printf("Result is %d \n", result);  
    printf("Press any key to continue...\n");  
    getchar();  
    return 0;  
}
```

## C/C++ WSL Option

## Install WSL

[GitHub - ucoruh/ns3-wsl-win10-setup: ns3 windows 10 WSL2 setup and usage](#)

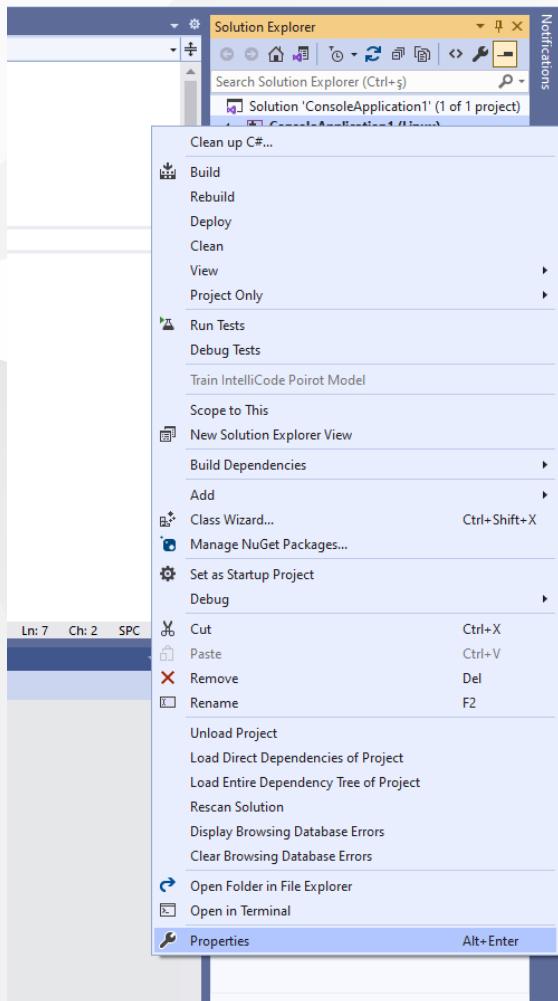
Create a Linux project

The screenshot shows the Microsoft Visual Studio Marketplace interface. At the top, there are three dropdown menus: 'C++' (selected), 'Linux' (selected), and 'All project types'. Below these, there are two main project categories displayed as cards:

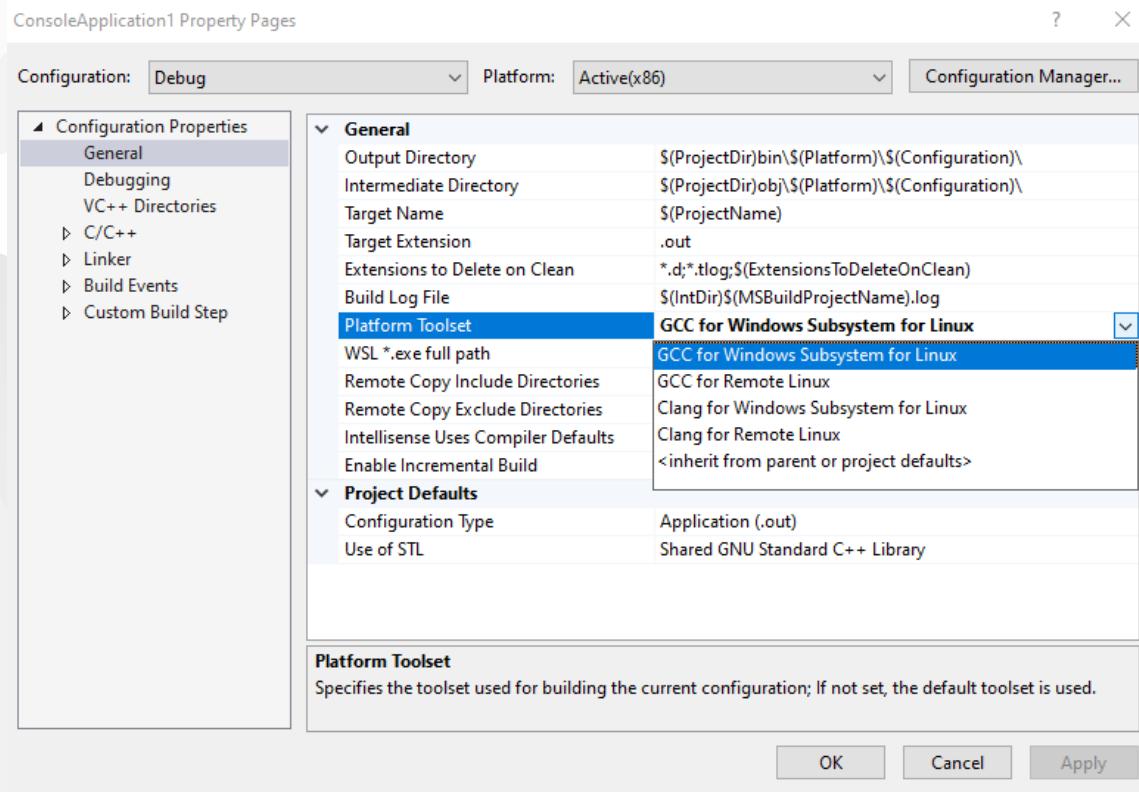
- CMake Project**:  
Icon: A CMake logo (a triangle composed of red, green, and blue segments).  
Description: Build modern, cross-platform C++ apps that don't depend on .sln or .vcxproj files.  
Buttons: C++, Windows, Linux, Console.
- Console Application**:  
Icon: A terminal window icon with a purple '+' sign.  
Description: Run code in a Linux terminal. Prints "hello" by default.  
Buttons: C++, Linux, Console.

At the bottom left is the RTEU logo, and at the bottom right is the page number '60'.

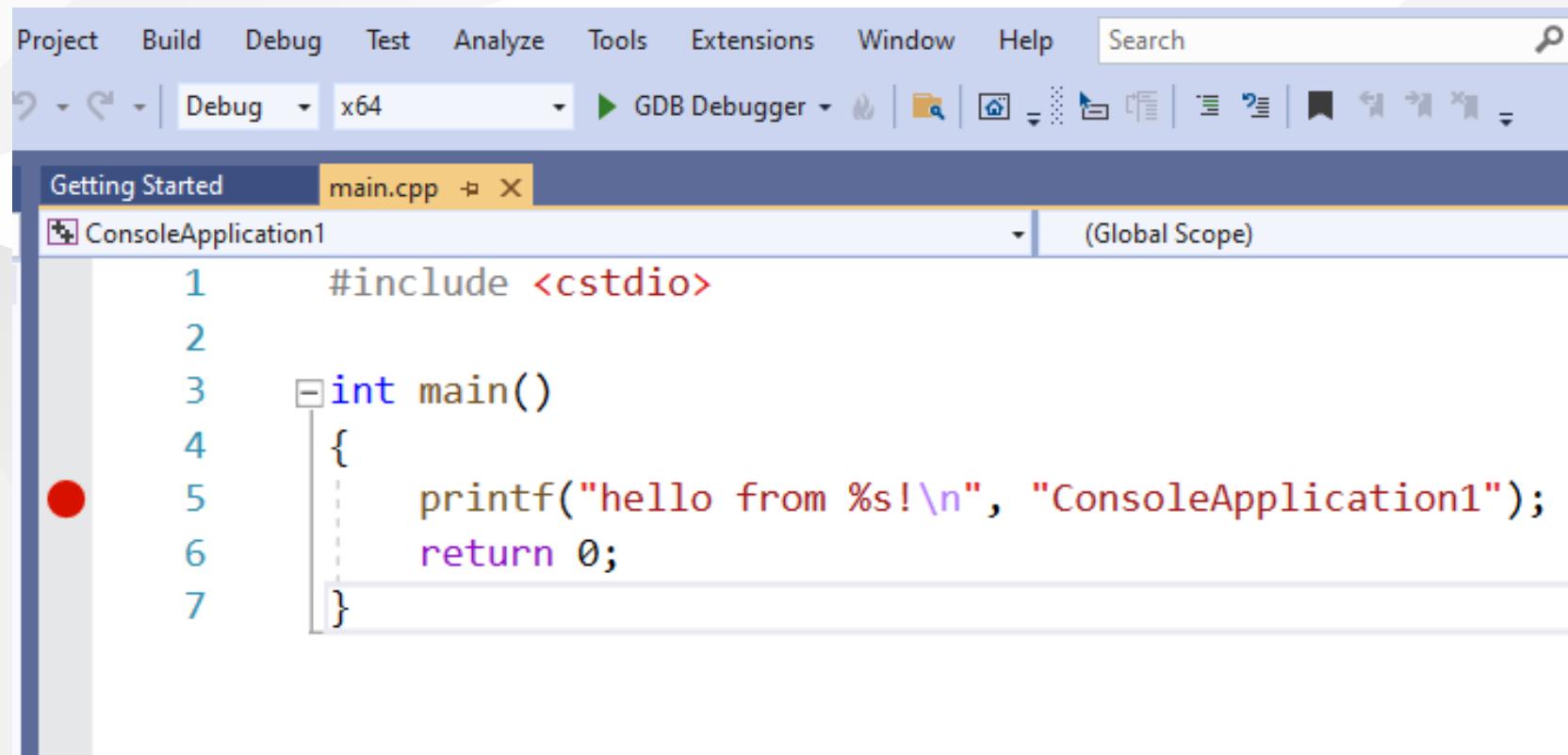
# Configure Platform Toolset to WSL



# Select GCC for Windows Subsystem for Linux



## Put a breakpoint and run debugger

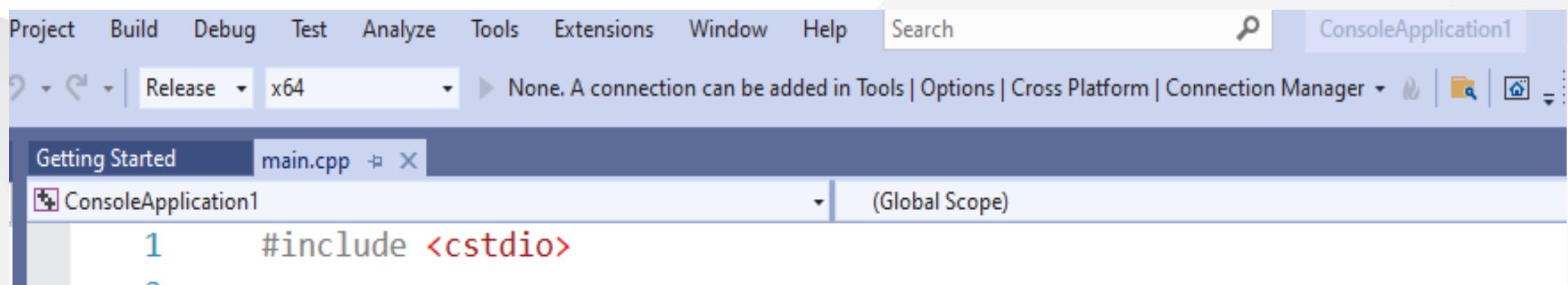


The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. The toolbar has various icons for file operations like Open, Save, and Build. The title bar shows "Getting Started" and "main.cpp". The code editor displays the following C++ code:

```
1 #include <stdio.h>
2
3 int main()
4 {
5     printf("Hello from %s!\n", "ConsoleApplication1");
6     return 0;
7 }
```

A red circular breakpoint marker is placed at the start of the `int main()` declaration. The status bar at the bottom indicates "(Global Scope)".

In the debugger for WSL you can use local WSL installation but if you want to run it on Release setting it require a SSH connection.



The screenshot shows the Microsoft Visual Studio IDE interface. The menu bar includes Project, Build, Debug, Test, Analyze, Tools, Extensions, Window, Help, and a Search bar. Below the menu is a toolbar with various icons. A dropdown menu shows 'Release' is selected. The main workspace shows a 'Getting Started' tab and a file 'main.cpp'. The code editor contains the following:

```
1 #include <cstdio>
```

## Configure SSH parameters

X

### Connect to Linux

This project uses remote builds, and a remote machine is required to host the builds and debug. Please enter the remote machine details below.

[Manage existing connections](#)

Host name:

Port:

User name:

Authentication type:

Password:

so you have to complete the following steps.

## C/C++ Remote Linux Option over SSH

Enable SSH

[SSH on Windows Subsystem for Linux \(WSL\) | Illuminia Studios](#)

Connect to Remote WSL Environment

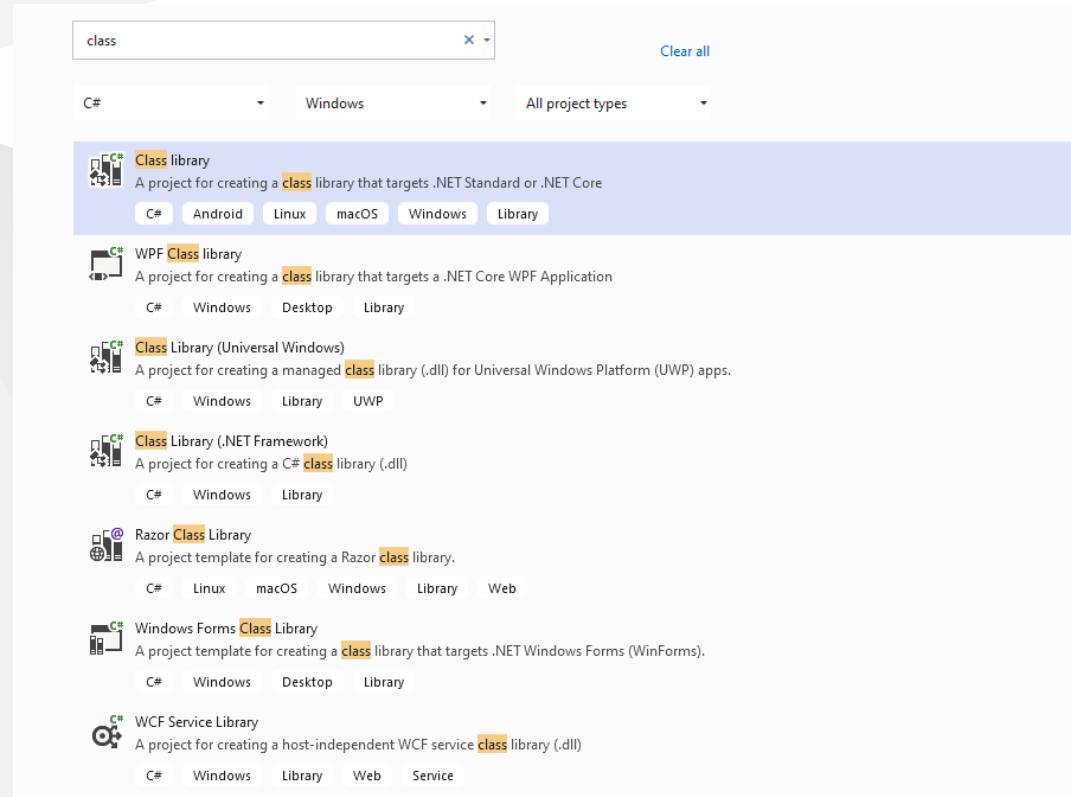
[Bağlan hedef Linux sisteminize Visual Studio | Microsoft Docs](#)

# C# Programming (Dinamik Library)

# Visual Studio Community Edition

In C# project we will create class library we have several options

for this sample we will select .NET core that we can build cross platform library



There is no static library option

## Configure your new project

Class library   C#   Android   Linux   macOS   Windows   Library

Project name

csharp-sample-lib

Location

E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce11

...

Solution name i

csharp-sample-lib

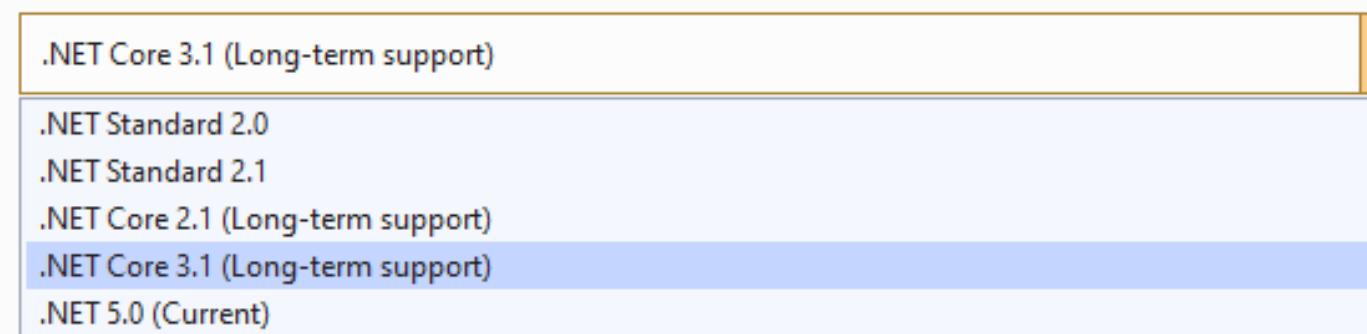
Place solution and project in the same directory

We will select .Net Core 3.1

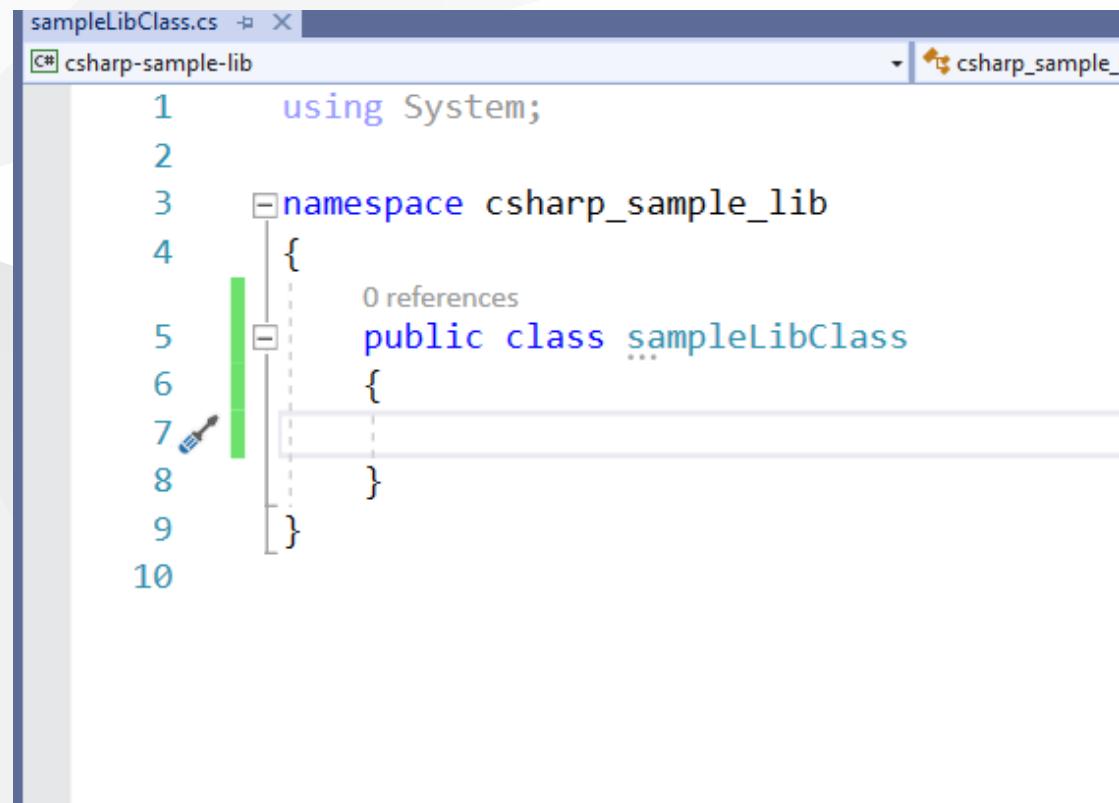
## Additional information

Class library    C#    Android    Linux    macOS    Windows    Library

Target Framework i



You will have default empty class library file

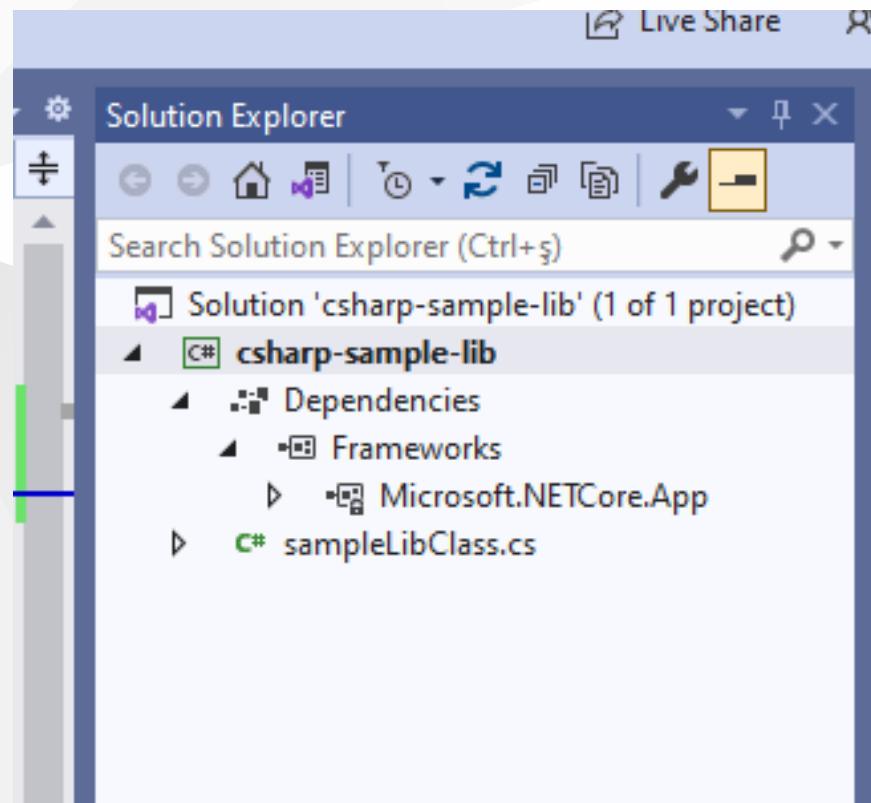


A screenshot of a Microsoft Visual Studio code editor window. The title bar shows the file name "sampleLibClass.cs" and the project name "csharp-sample-lib". The code editor displays the following C# code:

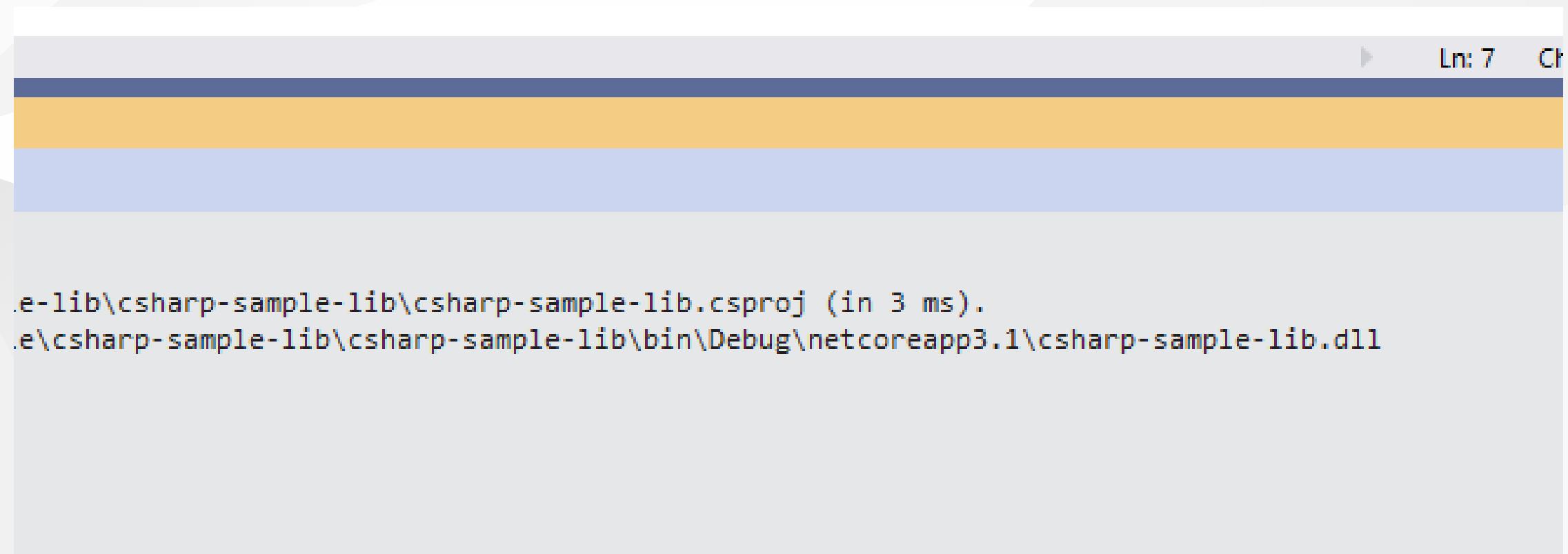
```
1  using System;
2
3  namespace csharp_sample_lib
4  {
5      public class sampleLibClass
6      {
7          ...
8      }
9  }
```

The code is numbered from 1 to 10 on the left. A green vertical bar with a small icon at the top is positioned to the left of the first few lines of code. The "sampleLibClass" class definition is highlighted with a light gray background. The "csharp\_sample\_lib" namespace is also highlighted.

In the project you can see .NETcore reference



We can build empty class library that generate dll for our application



The screenshot shows a terminal window with a dark blue header bar. In the header bar, there is a right-pointing arrow icon, the text "Ln: 7", and a "Ch" icon. The main area of the terminal is divided into three horizontal sections: a yellow section at the top, a light blue section in the middle, and a white section at the bottom where the command output is displayed. The output text is:  
.e-lib\csharp-sample-lib\csharp-sample-lib.csproj (in 3 ms).  
.e\csharp-sample-lib\csharp-sample-lib\bin\Debug\netcoreapp3.1\csharp-sample-lib.dll

Now we will add Console Application but this will also use .NETCore



## Select New Project

Search for templates (Alt+S)

C# Windows All project types

**Console Application**  
A project for creating a command-line application that can run on .NET Core on Windows, Linux and macOS  
C# Linux macOS Windows Console

**ASP.NET Core Web App**  
A project template for creating an ASP.NET Core application with example ASP.NET Razor Pages content.  
C# Linux macOS Windows Cloud Service Web

Name the project

## Configure your new project

Console Application    C#    Linux    macOS    Windows    Console

Project name

csharp-sample-app

Location

C:\Users\ugur.coruh\Desktop\csharp-lib-sample\csharp-sample-lib\

...

## Select .NETCore framework

### Additional information

Console Application    C#    Linux    macOS    Windows    Console

Target Framework 

.NET Core 3.1 (Long-term support)

.NET Core 2.1 (Long-term support)

.NET Core 3.1 (Long-term support)

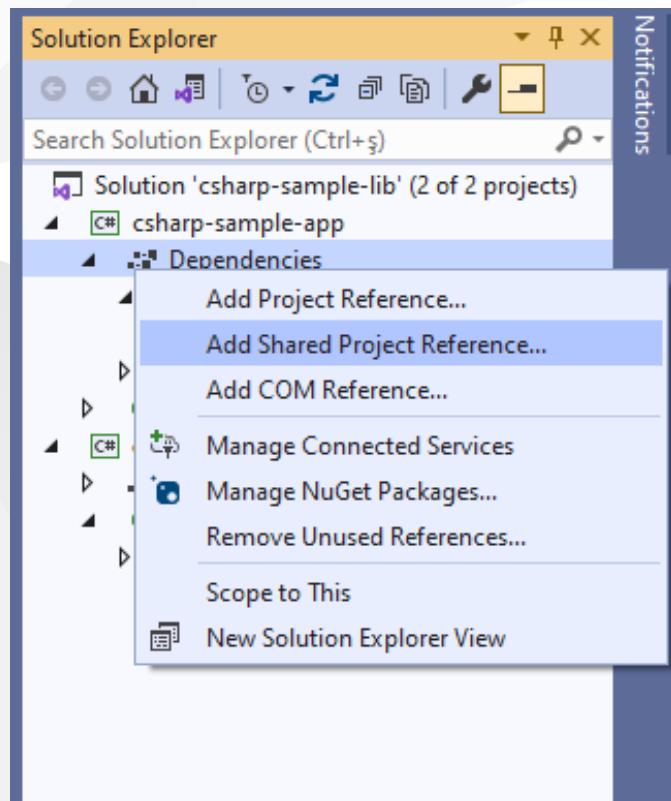
.NET 5.0 (Current)

You will have the following sample main.cs file

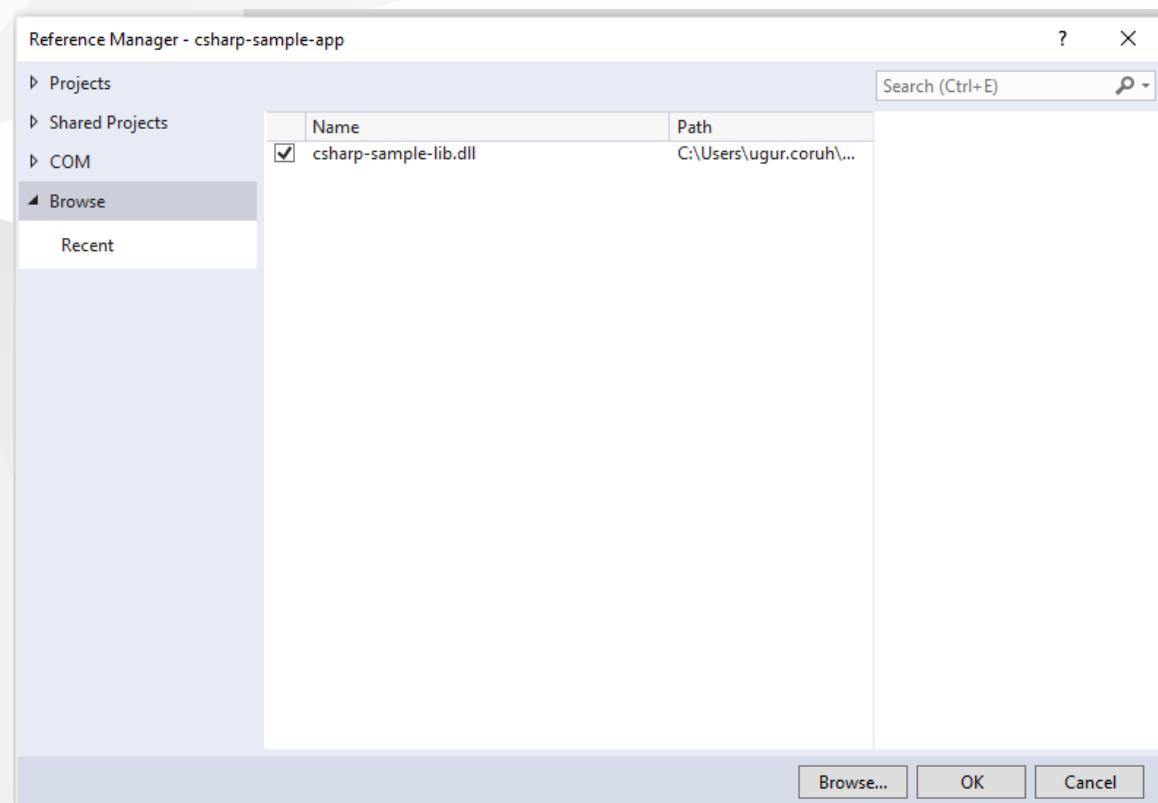
```
using System;

namespace csharp_sample_app
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");
        }
    }
}
```

Now we can link projects with adding references open reference section



browse for class library project output folder and select output dll file for console application



now we can update our library code and use it in console application

copy following sample to sampleLibClass file in the library

```
using System;

namespace csharp_sample_lib
{
    public class sampleLibClass
    {
        public static void sayHelloTo(string name)
        {
            if (!String.IsNullOrEmpty(name))
            {
                Console.WriteLine("Hello " + name);
            }
            else
            {
                Console.WriteLine("Hello There");
            }
        }

        public static int sum(int a, int b)
        {
            int c = 0;
            c = a + b;
            return c;
        }
    }
}
```



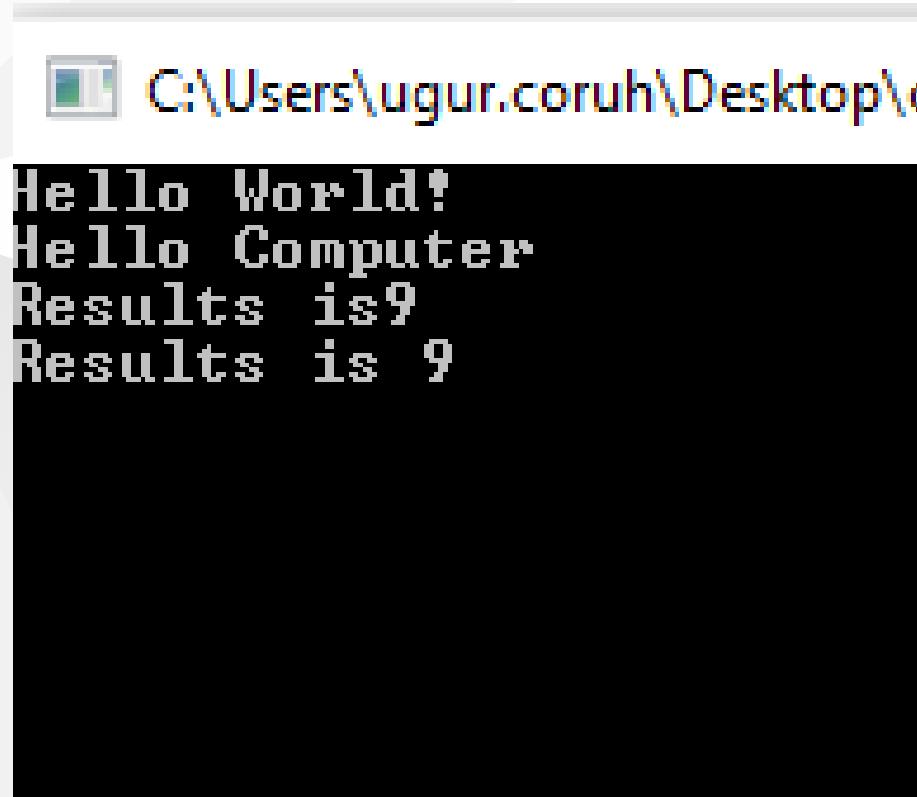
after this operation copy following sample to console application and build app then you can run

```
using csharp_sample_lib;
using System;

namespace csharp_sample_app
{
    class Program
    {
        static void Main(string[] args)
        {
            Console.WriteLine("Hello World!");

            sampleLibClass.sayHelloTo("Computer");
            int result = sampleLibClass.sum(5, 4);
            Console.WriteLine("Results is " + result);
            Console.WriteLine("Results is {0}", result);
            Console.Read();
        }
    }
}
```

You will see following output that mean we called DLL functions



```
C:\Users\ugur.coruh\Desktop\c
Hello World!
Hello Computer
Results is 9
Results is 9
```

Also we can publish this console application with dll for linux environment or others  
for linux environment we should install .NETCore

follow the link below or commands that I shared with you as below for deployment

## [How to Install Dotnet Core on Ubuntu 20.04 – TecAdmin](#)

Step 1 – Enable Microsoft PPA

```
wget https://packages.microsoft.com/config/ubuntu/20.04/packages-microsoft-prod.deb  
sudo dpkg -i packages-microsoft-prod.deb
```

## Step 2 – Installing Dotnet Core SDK

```
sudo apt update  
sudo apt install apt-transport-https  
sudo apt install dotnet-sdk-3.1
```

### Step 3 – Install Dotnet Core Runtime Only

To install .NET Core Runtime on Ubuntu 20.04 LTS system, execute the commands:

```
sudo apt update
```

To install the previous version of .Net core runtime 2.1, type:

```
sudo apt install dotnet-runtime-2.1
```

Press "y" for any input prompted by the installer.

## Step 4 – (Optional) Check .NET Core Version

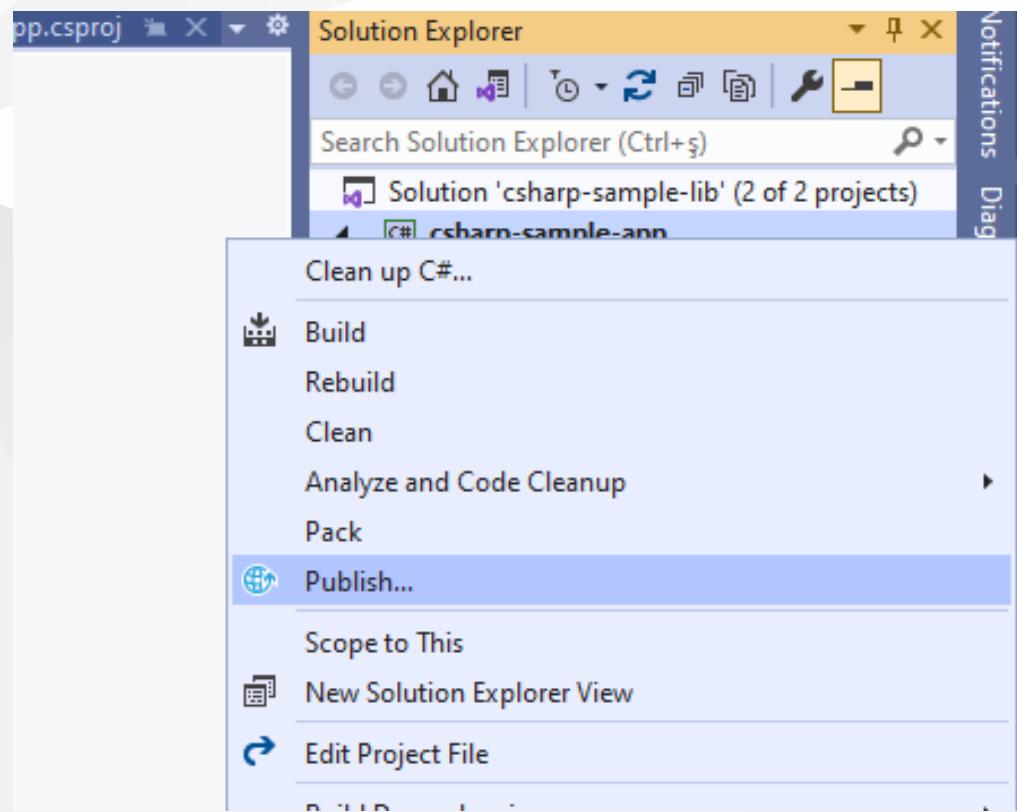
You can use dotnet command line utility to check installed version of .NET Core on your system. To check dotnet version, type:

```
dotnet --version
```

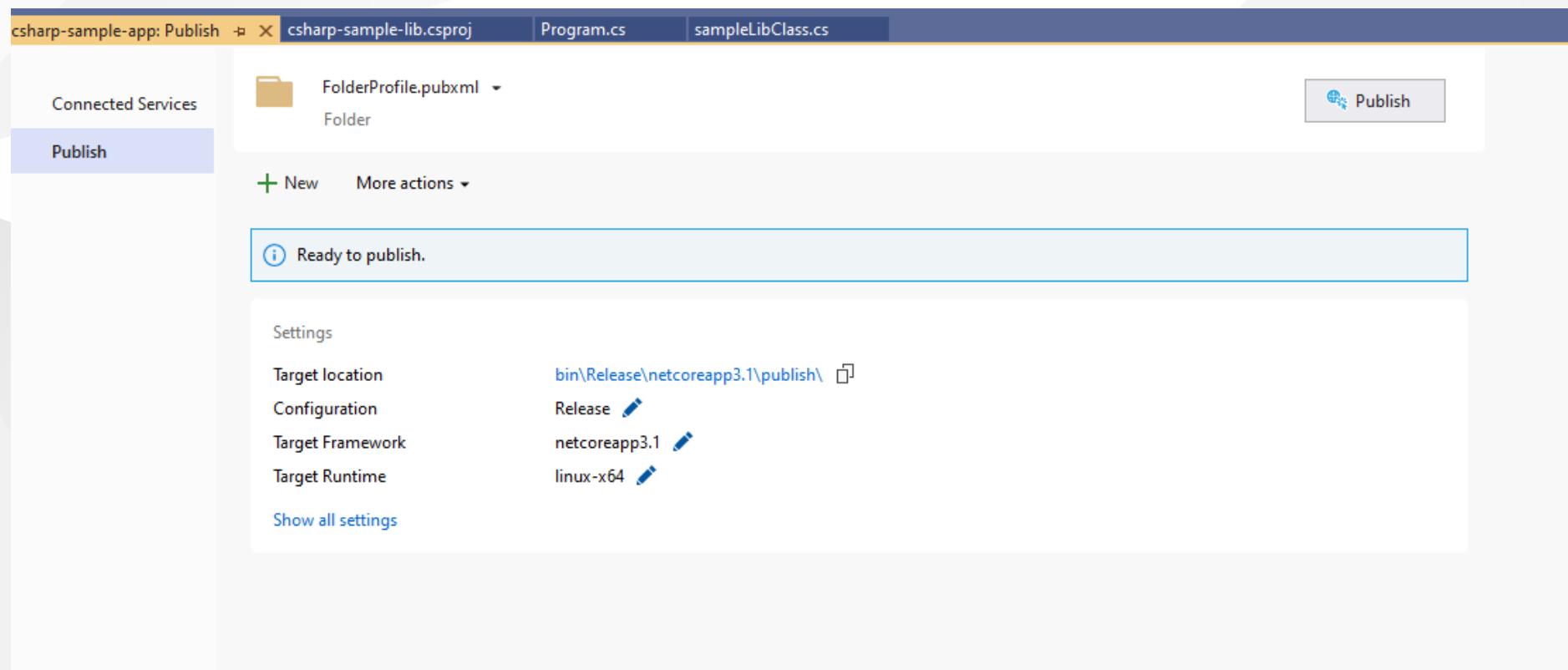
```
ucoruh@LAPTOP-RQNNNS:~$ dotnet --version  
3.1.414  
ucoruh@LAPTOP-RQNNNS:~$
```

Now we will publish our application as single executable

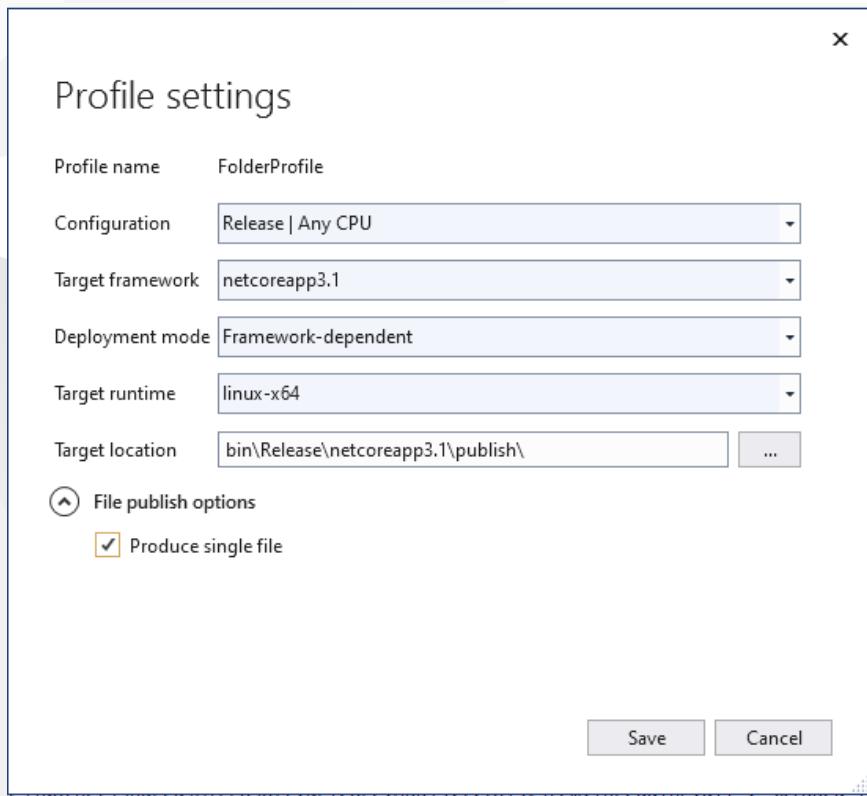
Open publish menu



## Select netcoreapp3.1 and Release for linux-x64



## Select produce single file



After successfull publish you will have linux binary that you can run with WSL

```
esktop > csharp-lib-sample > csharp-sample-lib > csharp-sample-app > bin > Release > netcoreapp3.1 > publish
```

Name	Date modified	Type	Size
csharp-sample-app	10/24/2021 1:36 AM	File	97 KB
csharp-sample-app.pdb	10/24/2021 1:36 AM	Program Debug D...	10 KB
csharp-sample-lib.pdb	10/24/2021 1:30 AM	Program Debug D...	10 KB
packages-microsoft-prod.deb	4/23/2020 10:02 PM	DEB File	4 KB

Open WSL and enter the path where this folder located  
and run application as follow

```
Processing triggers for man-db (2.9.1-1) ...
ucoruh@LAPTOP-RQNN9IG:/mnt/c/Users/ugur.coruh/Desktop/csharp-lib-sample/csharp-sample-lib/csharp-sample-app/bin/Release/netcoreapp3.1/publish$ dotnet --version
3.1.414
ucoruh@LAPTOP-RQNN9IG:/mnt/c/Users/ugur.coruh/Desktop/csharp-lib-sample/csharp-sample-lib/csharp-sample-app/bin/Release/netcoreapp3.1/publish$ ./
csharp-sample-app      csharp-sample-app.pdb      csharp-sample-lib.pdb      packages-microsoft-prod.deb
ucoruh@LAPTOP-RQNN9IG:/mnt/c/Users/ugur.coruh/Desktop/csharp-lib-sample/csharp-sample-lib/csharp-sample-app/bin/Release/netcoreapp3.1/publish$ ./csharp-sample-app
Hello World!
Hello Computer
Results is9
Results is 9
```

check dotnet --version and then run application

```
ublish$ dotnet --version  
ublish$ ./  
ublish$ ./csharp-sample-app
```

you will see similar output

```
ucoruh@LAPTOP-RQNN9IG:/mnt/c/  
csharp-sample-app  
ucoruh@LAPTOP-RQNN9IG:/mnt/c/  
Hello World!  
Hello Computer  
Results is9  
Results is 9
```

In this sample we created single application from settings lets try with shared library located option uncheck the "produce single file" option and publish again.

Then you will have the following outputs

top > csharp-lib-sample > csharp-sample-lib > csharp-sample-app > bin > Release > netcoreapp3.1 > publish				
Name	Date modified	Type	Size	
csharp-sample-app	10/24/2021 1:36 AM	File	88 KB	
csharp-sample-app.deps.json	10/24/2021 1:36 AM	JSON File	1 KB	
csharp-sample-app.dll	10/24/2021 1:36 AM	Application exten...	4 KB	
csharp-sample-app.pdb	10/24/2021 1:36 AM	Program Debug D...	10 KB	
csharp-sample-app.runtimeconfig.json	10/24/2021 1:36 AM	JSON File	1 KB	
csharp-sample-lib.dll	10/24/2021 1:30 AM	Application exten...	4 KB	
csharp-sample-lib.pdb	10/24/2021 1:30 AM	Program Debug D...	10 KB	

If you run csharp-sample-app  
you will have the same output

```
ucoruh@LAPTOP-RQNN5I: ~
Hello World!
Hello Computer
Results is 9
Results is 9
```

# Java Programming

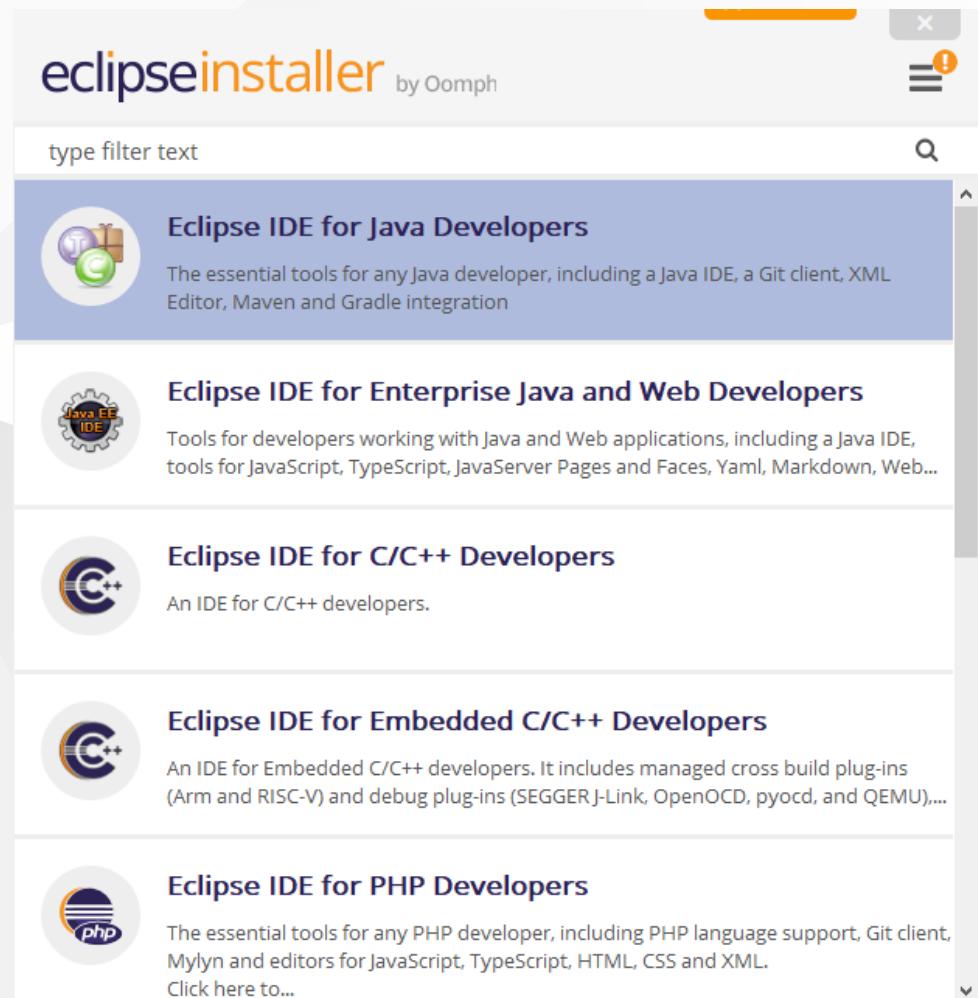


## Eclipse IDE

You should download and install eclipse installer and then you should select Eclipse IDE for Java Developers

[Eclipse Installer 2021-09 R | Eclipse Packages](#)





The screenshot shows the Eclipse Installer by Oomph interface. At the top, there's a header with the title "eclipseinstaller" in orange and "by Oomph". Below the header is a search bar with the placeholder "type filter text" and a magnifying glass icon. To the right of the search bar is a settings icon with an exclamation mark.

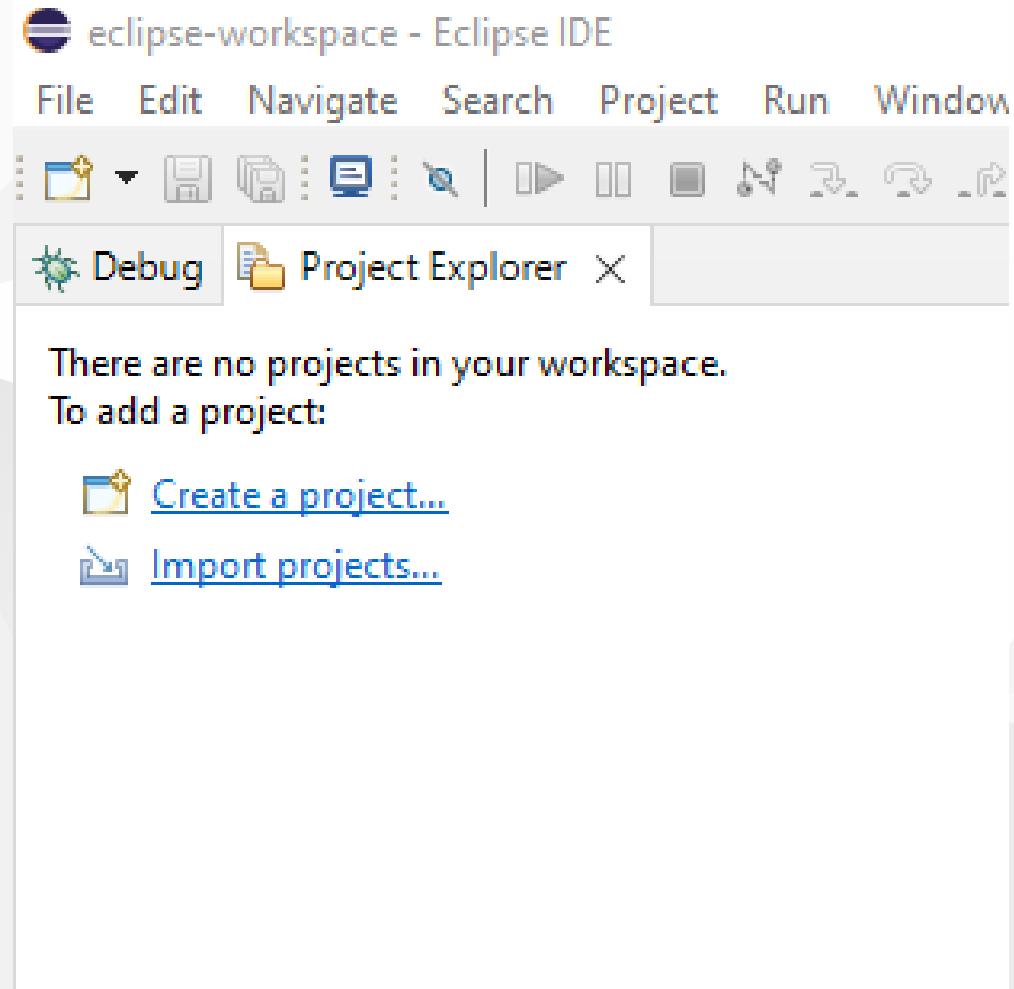
The main area displays five listed options, each with an icon and a brief description:

- Eclipse IDE for Java Developers**  
The essential tools for any Java developer, including a Java IDE, a Git client, XML Editor, Maven and Gradle integration.
- Eclipse IDE for Enterprise Java and Web Developers**  
Tools for developers working with Java and Web applications, including a Java IDE, tools for JavaScript, TypeScript, JavaServer Pages and Faces, Yaml, Markdown, Web...
- Eclipse IDE for C/C++ Developers**  
An IDE for C/C++ developers.
- Eclipse IDE for Embedded C/C++ Developers**  
An IDE for Embedded C/C++ developers. It includes managed cross build plug-ins (Arm and RISC-V) and debug plug-ins (SEGGER J-Link, OpenOCD, pyocd, and QEMU),...
- Eclipse IDE for PHP Developers**  
The essential tools for any PHP developer, including PHP language support, Git client, Mylyn and editors for JavaScript, TypeScript, HTML, CSS and XML.  
[Click here to...](#)

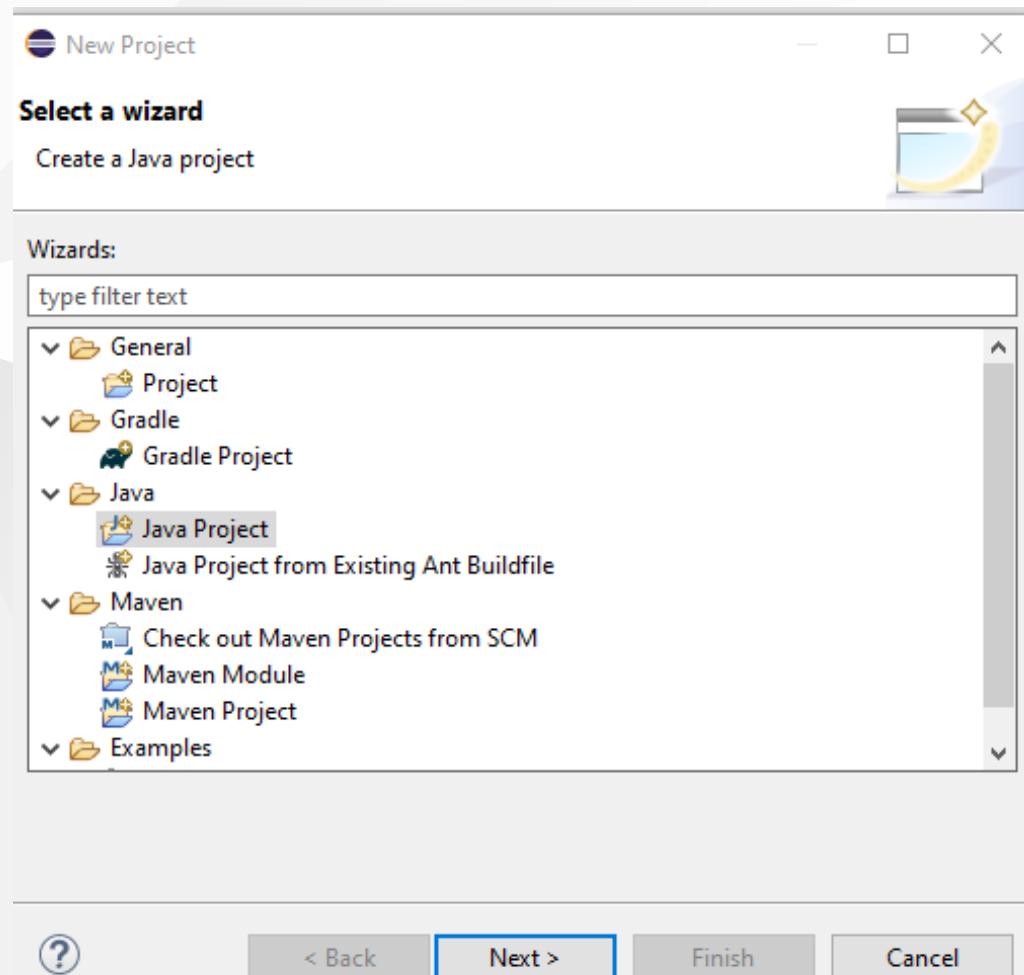




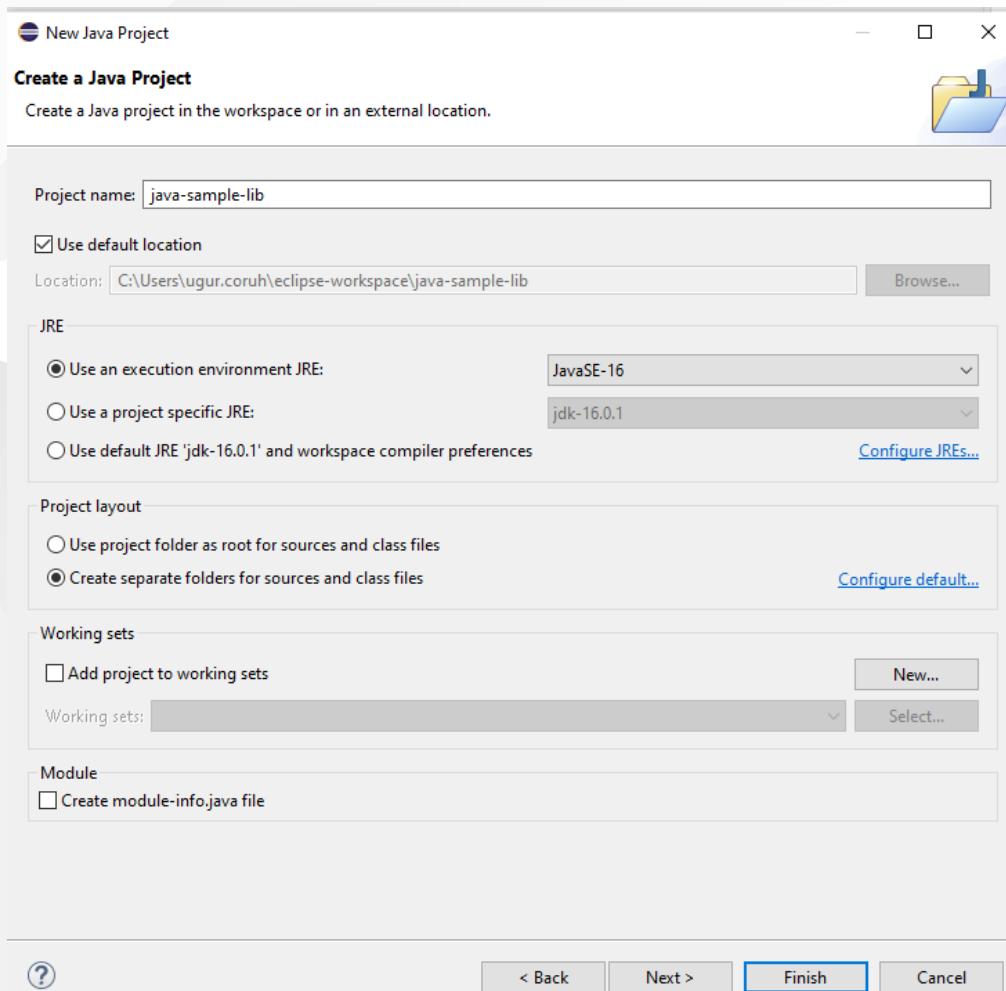
## select create a project



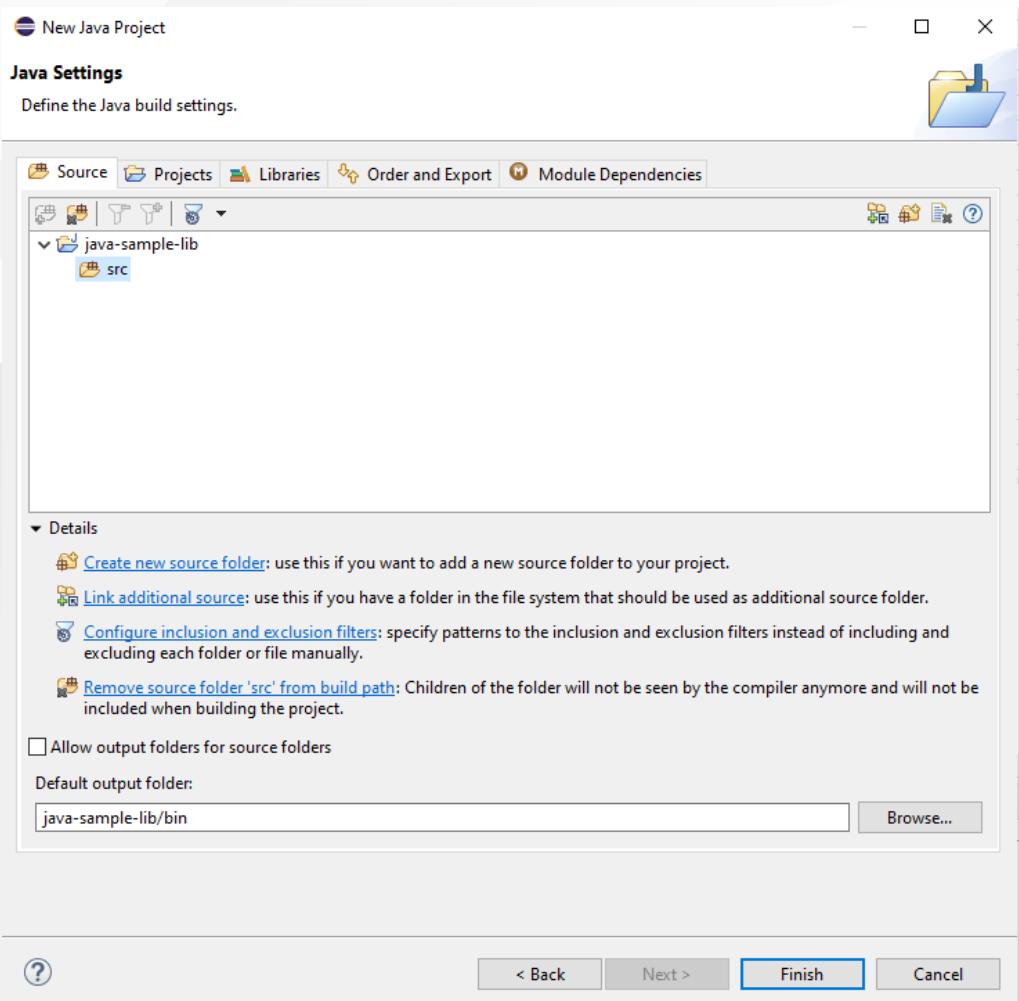
## select java project



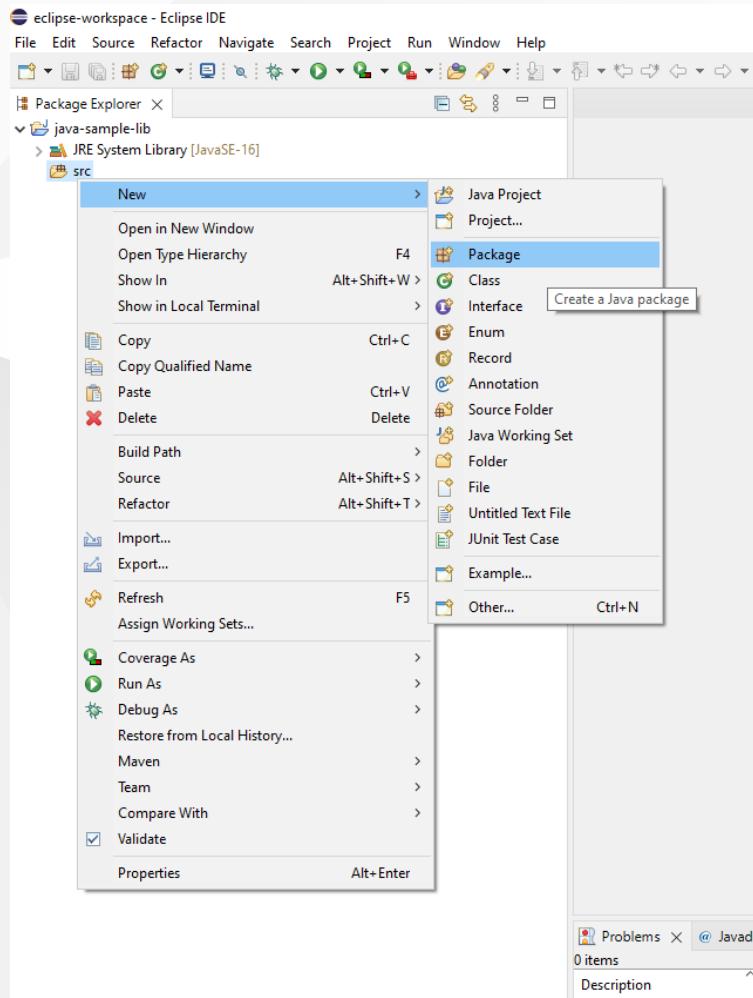
## give project name



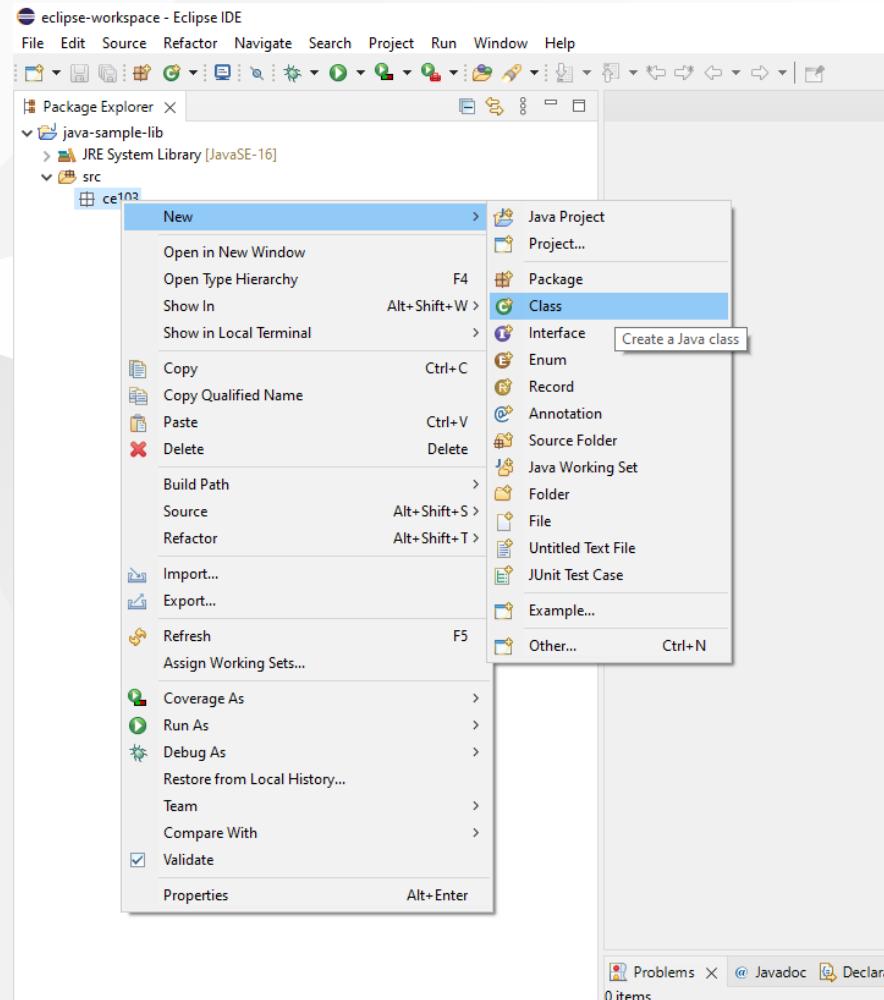
## select finish



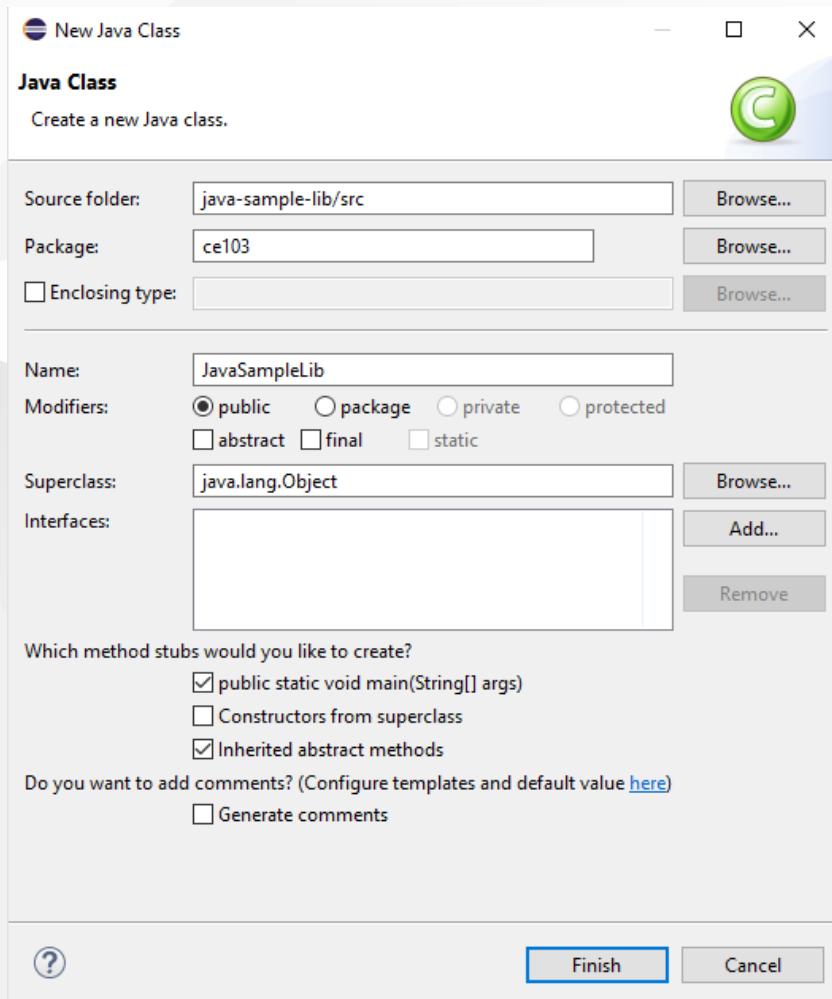
first we need to add a default package to keep everything organized



then we can create our class that includes our functions



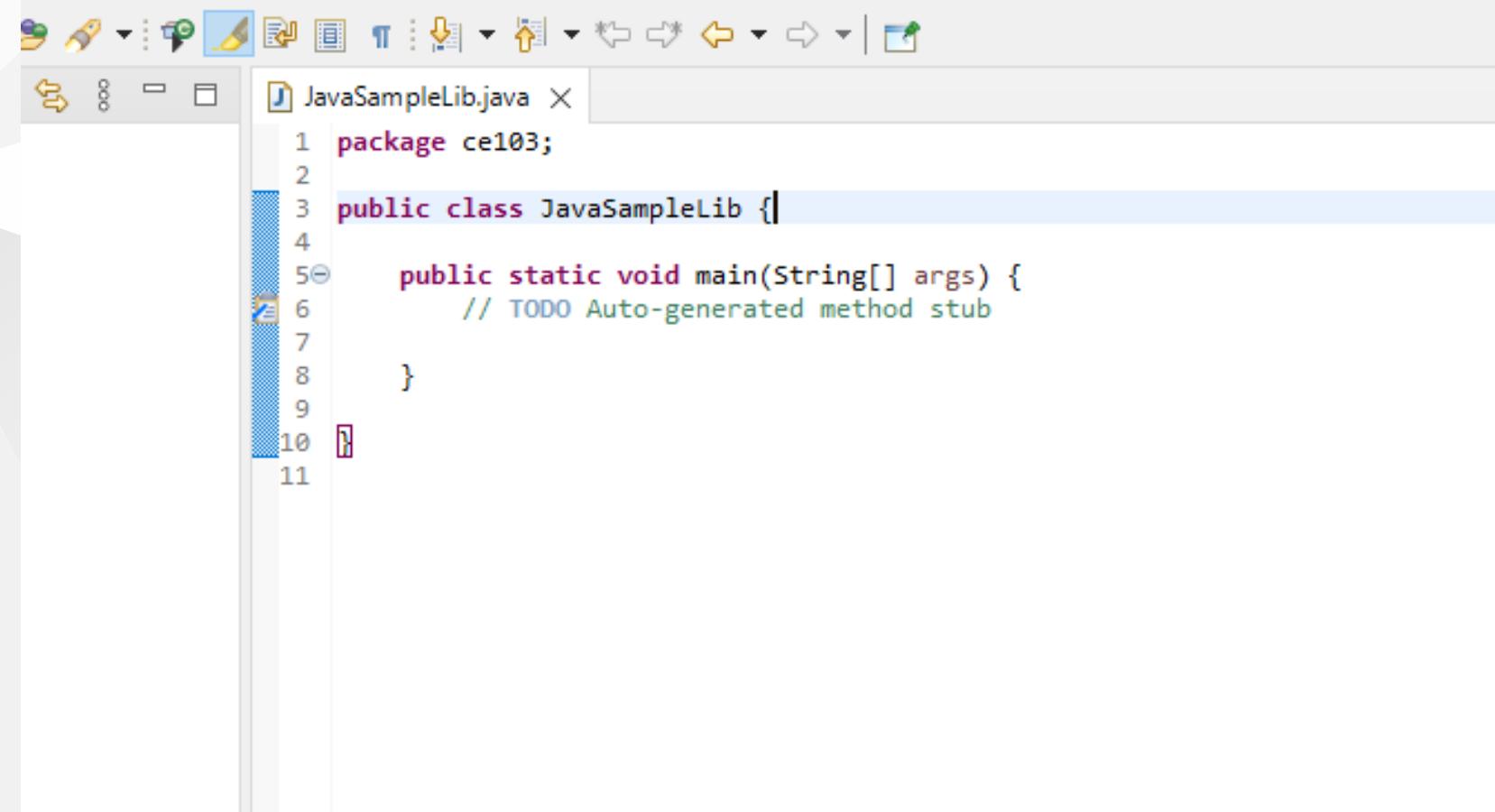
# give class a name



you will have following class with main

Eclipse IDE

Window Help



A screenshot of the Eclipse IDE interface. The title bar shows "JavaSampleLib.java". The code editor displays the following Java code:

```
1 package ce103;
2
3 public class JavaSampleLib {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8     }
9
10}
11
```

# We will create sample java library with static functions as below.

```
package ce103;

import java.io.IOException;

public class JavaSampleLib {

    public static void sayHelloTo(String name) {
        if(name.isBlank() || name.isEmpty())
        {
            System.out.println("Hello "+name);
        }else {
            System.out.println("Hello There");
        }
    }

    public static int sum(int a,int b)
    {
        int c = 0;
        c = a+b;
        return c;
    }

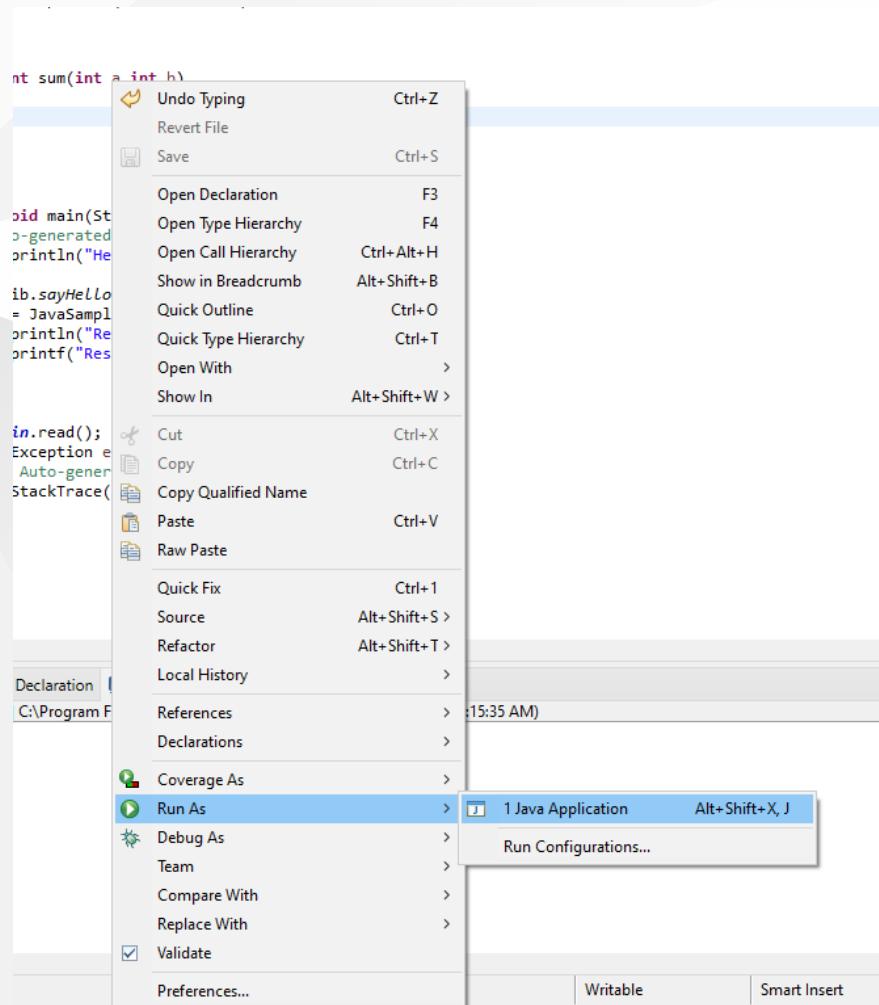
    public static void main(String[] args) {
        // TODO Auto-generated method stub
        System.out.println("Hello World!");

        JavaSampleLib.sayHelloTo("Computer");
        int result = JavaSampleLib.sum(5, 4);
        System.out.println("Results is" + result);
        System.out.printf("Results is %d \n", result);

        try {
            System.in.read();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```



also we can add main method to run our library functions. If we run this file its process main function



we can see output from console as below

The screenshot shows the Eclipse IDE interface with the following details:

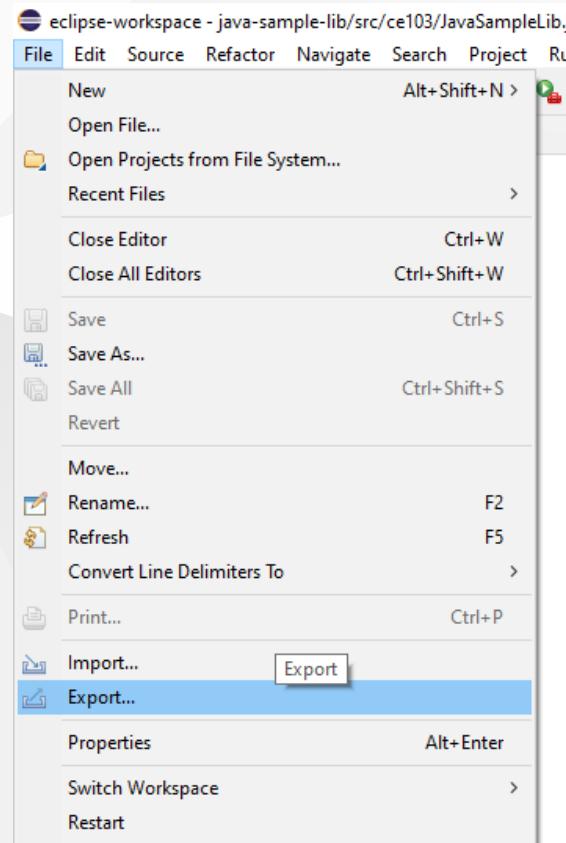
- File Menu:** File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Package Explorer:** Shows a project named "java-sample-lib" containing a package "ce103" which has a file "JavaSampleLib.java".
- Java Sample Library Code:** The code is as follows:

```
1 package ce103;
2
3 import java.io.IOException;
4
5 public class JavaSampleLib {
6
7     public static void sayHelloTo(String name) {
8         if(name.isBlank() || name.isEmpty())
9             {
10                 System.out.println("Hello "+name);
11             }else {
12                 System.out.println("Hello There");
13             }
14     }
15
16     public static int sum(int a,int b)
17     {
18         int c = 0;
19         c = a+b;
20         return c;
21     }
22
23     public static void main(String[] args) {
24         // TODO Auto-generated method stub
25         System.out.println("Hello World!");
26
27         JavaSampleLib.sayHelloTo("Computer");
28         int result = JavaSampleLib.sum(5, 4);
29         System.out.println("Results is" + result);
30         System.out.printf("Results is %d \n", result);
31
32         try {
33             System.in.read();
34         } catch (IOException e) {
35             // TODO Auto-generated catch block
36             e.printStackTrace();
37         }
38     }
39
40 }
41
42 }
```

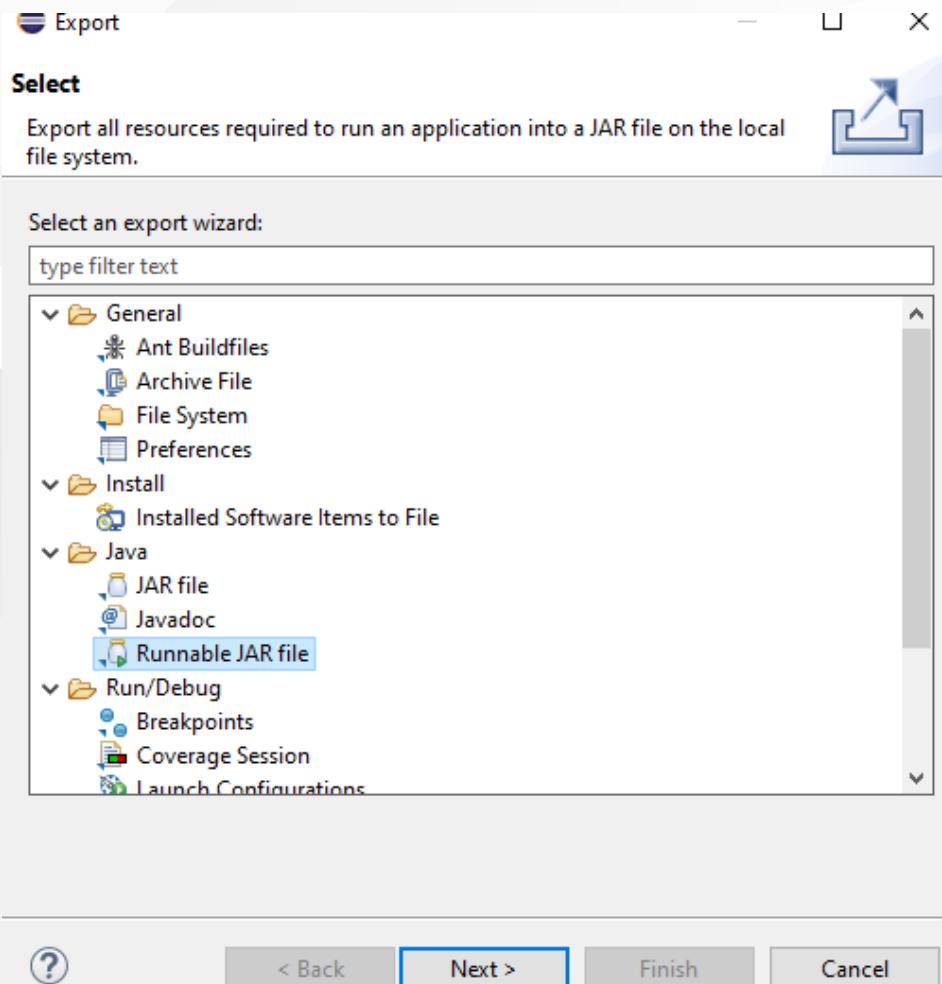
**Console Output:**

```
Hello World!
Hello There
Results is9
Results is 9
```

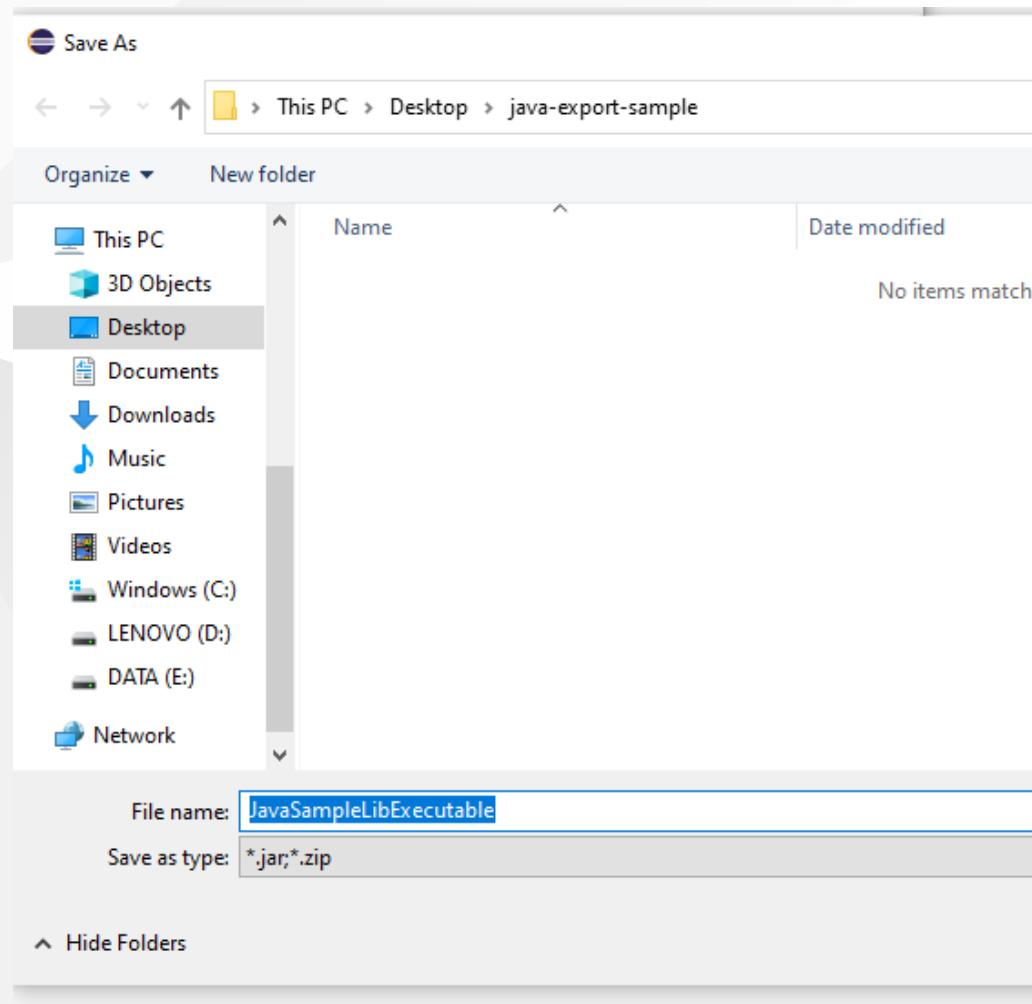
There is no exe files java runtime environment run class files but we can export this as an executable.



## Select Java->Runnable JAR File

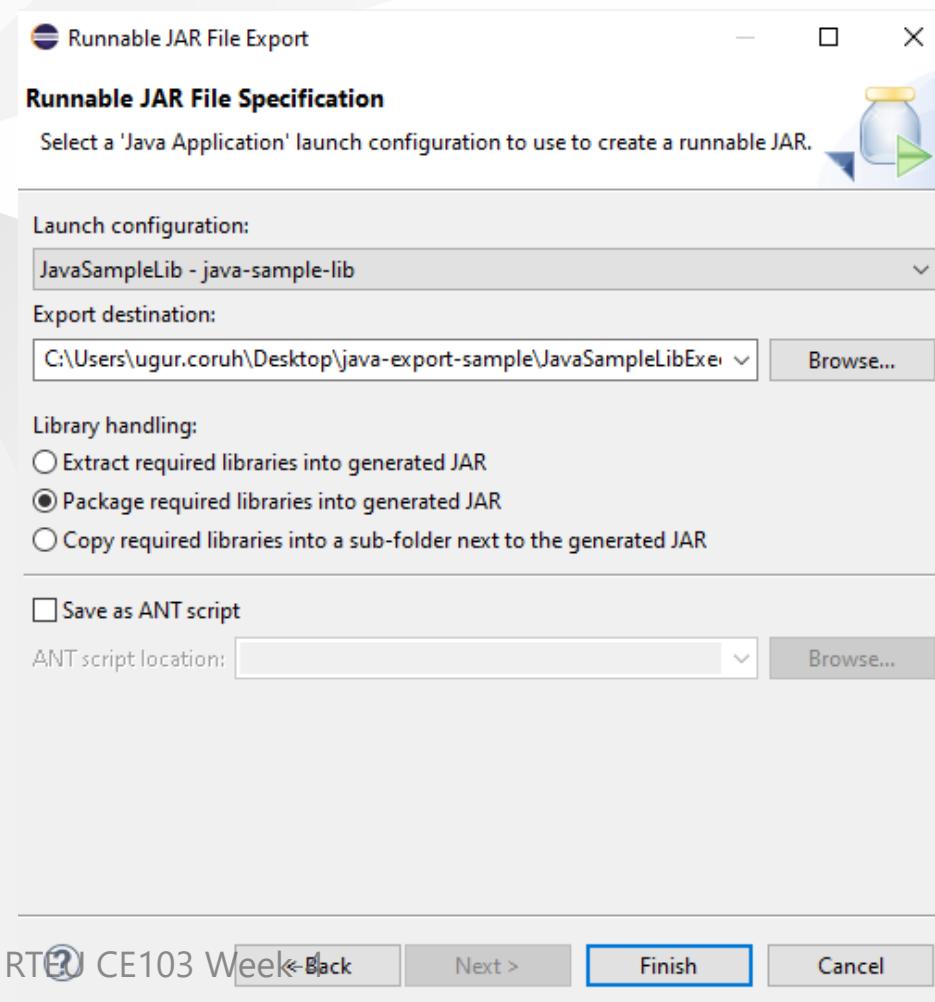


click next and set output path for jar file

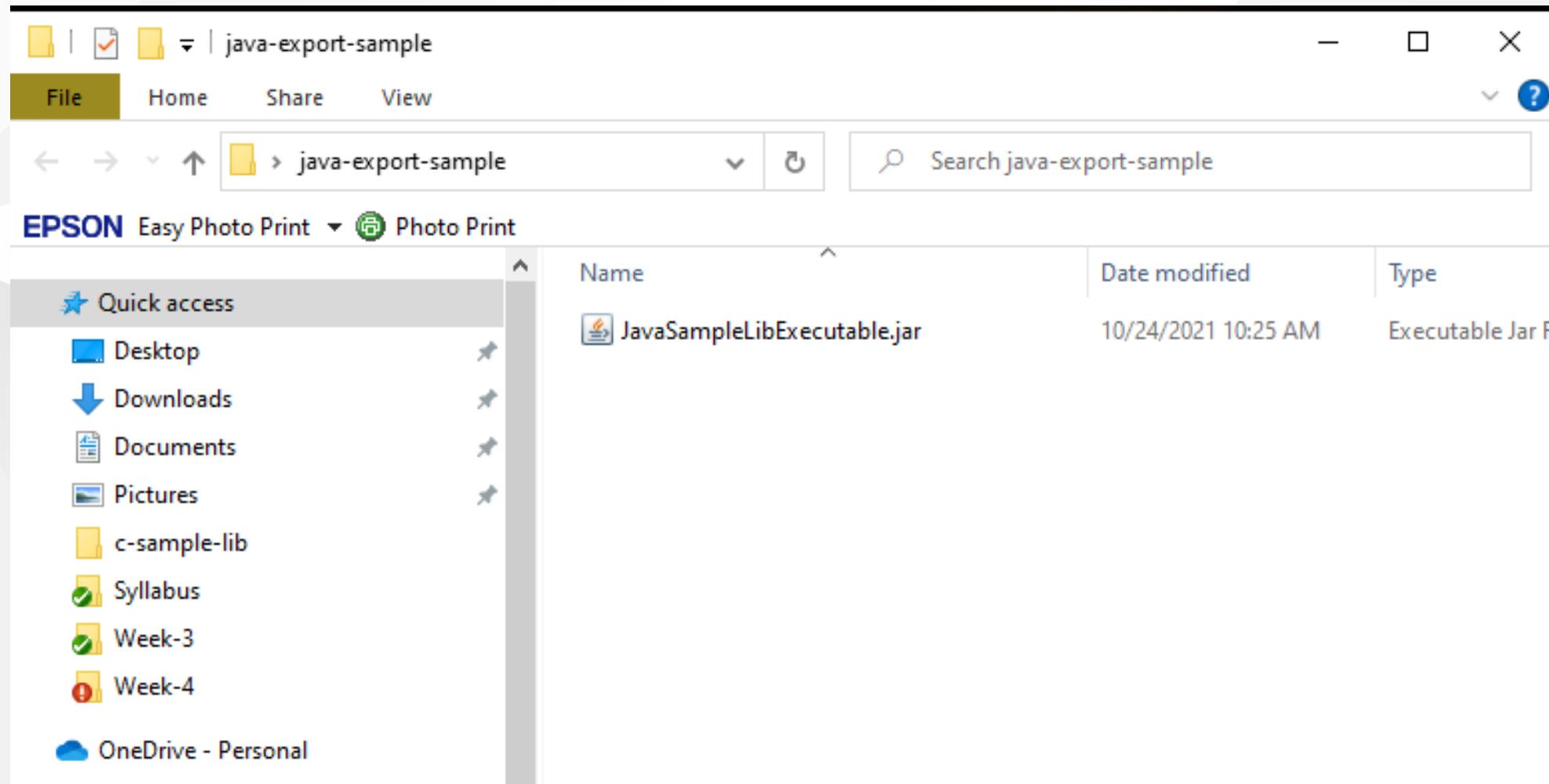


If our project has several external dependency then we can extract this required files (jar, so, dll) in seperated folder or we can combine them and generate a single executable jar

Lets pack everthing together, Select launch configuration that has main function



end of this operation we will have the following jar that we can by click



When you click application if cannot run then try command line to see problem

enter jar folder and run the following command

```
java -jar JavaSampleLibExecutable.jar
```

```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleLibExecutable.jar
Exception in thread "main" java.lang.UnsupportedClassVersionError: ce103/JavaSampleLib has been compiled by a more recent
version of the Java Runtime (class file version 60.0), this version of the Java Runtime only recognizes class file ver-
sions up to 52.0
        at java.lang.ClassLoader.defineClass1(Native Method)
        at java.lang.ClassLoader.defineClass(Unknown Source)
        at java.security.SecureClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.defineClass(Unknown Source)
        at java.net.URLClassLoader.access$100(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.net.URLClassLoader$1.run(Unknown Source)
        at java.security.AccessController.doPrivileged(Native Method)
        at java.net.URLClassLoader.findClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.ClassLoader.loadClass(Unknown Source)
        at java.lang.Class.forName0(Native Method)
        at java.lang.Class.forName(Unknown Source)
        at org.eclipse.jdt.internal.jarinjarloader.JarRsrcLoader.main(JarRsrcLoader.java:59)

C:\Users\ugur.coruh\Desktop\java-export-sample>
```

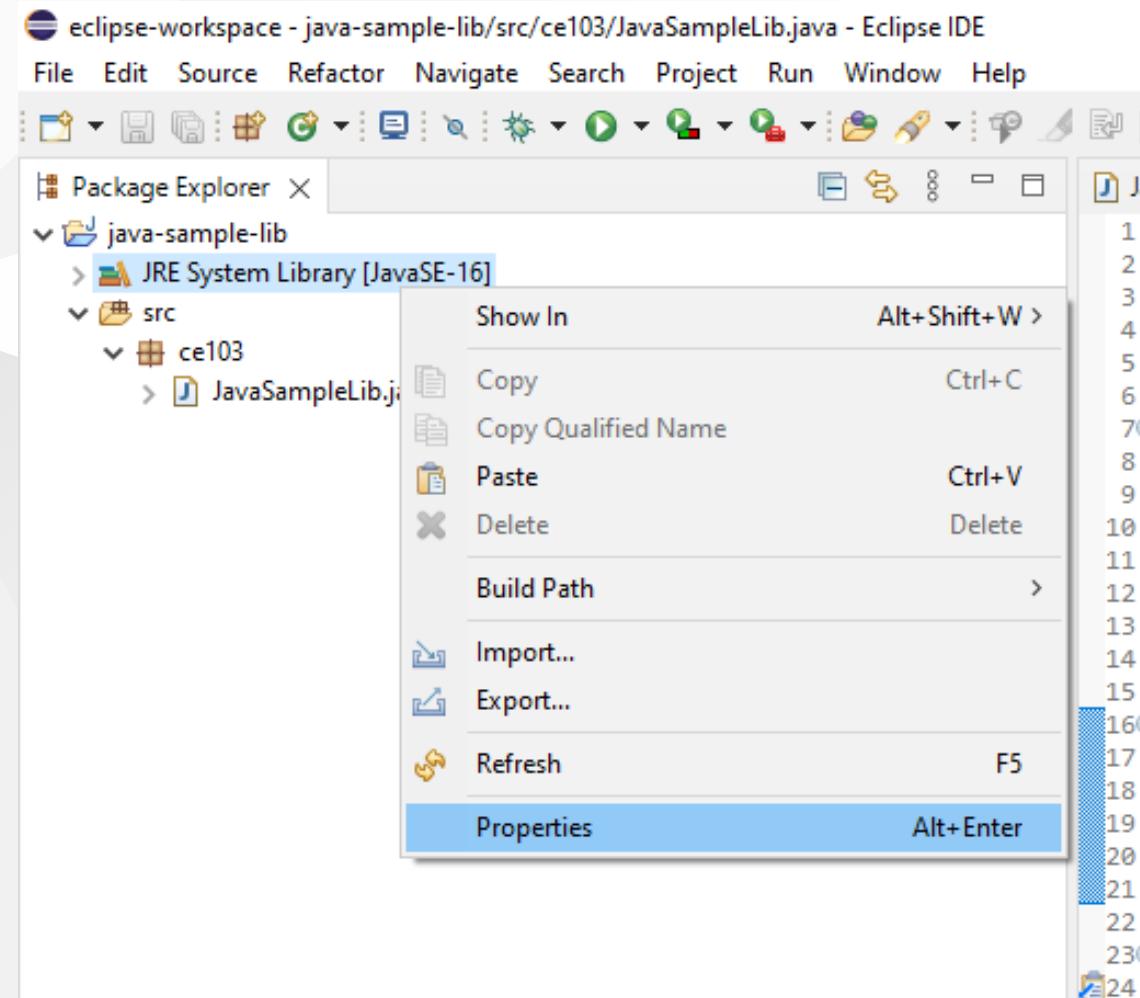
In my case eclipse build JDK is newer than that I installed and set for my OS

If we check version we can see problem Java version 1.8.0\_231

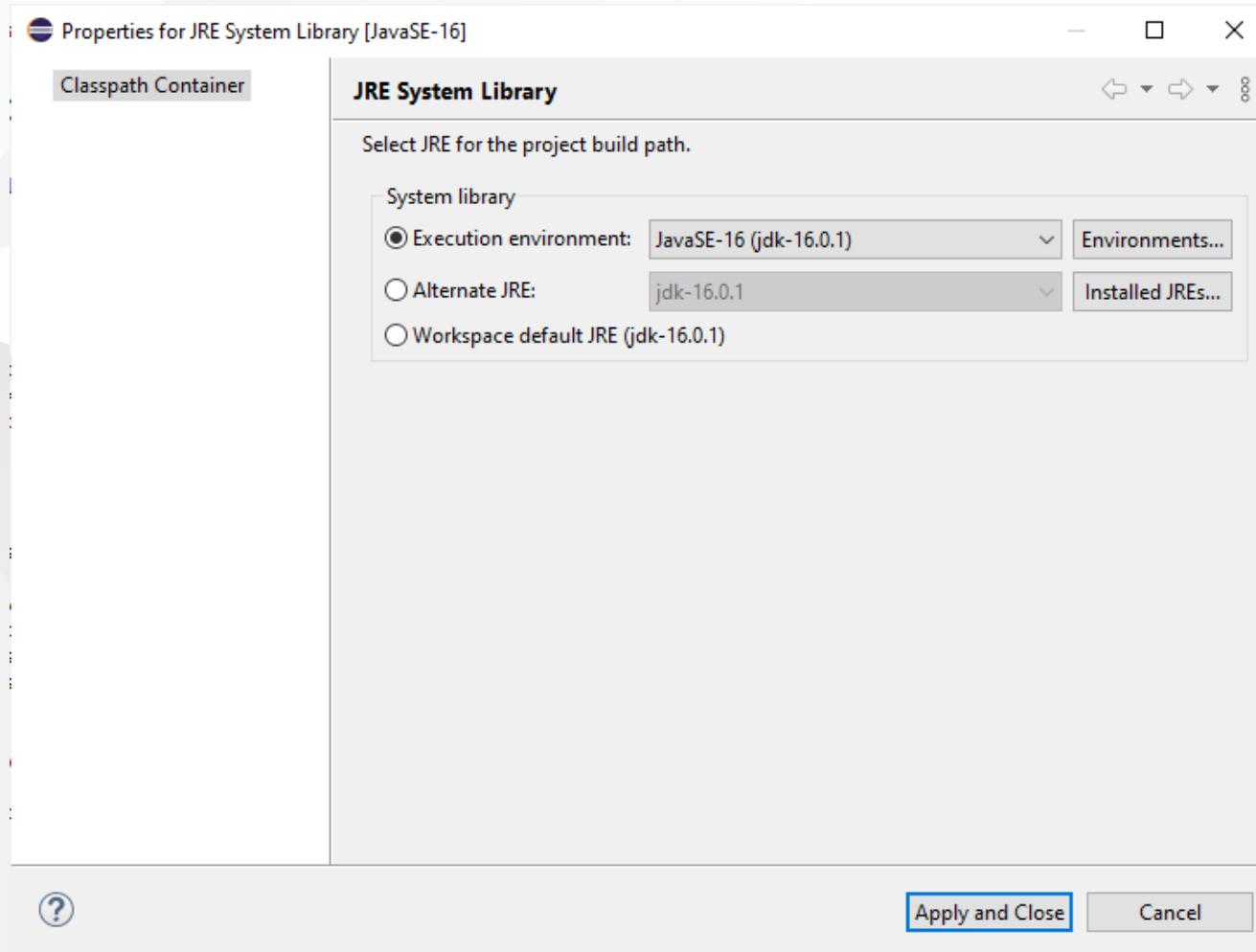
```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -showversion
java version "1.8.0_231"
Java(TM) SE Runtime Environment (build 1.8.0_231-b11)
Java HotSpot(TM) 64-Bit Server VM (build 25.231-b11, mixed mode)

Usage: java [-options] class [args...]
```

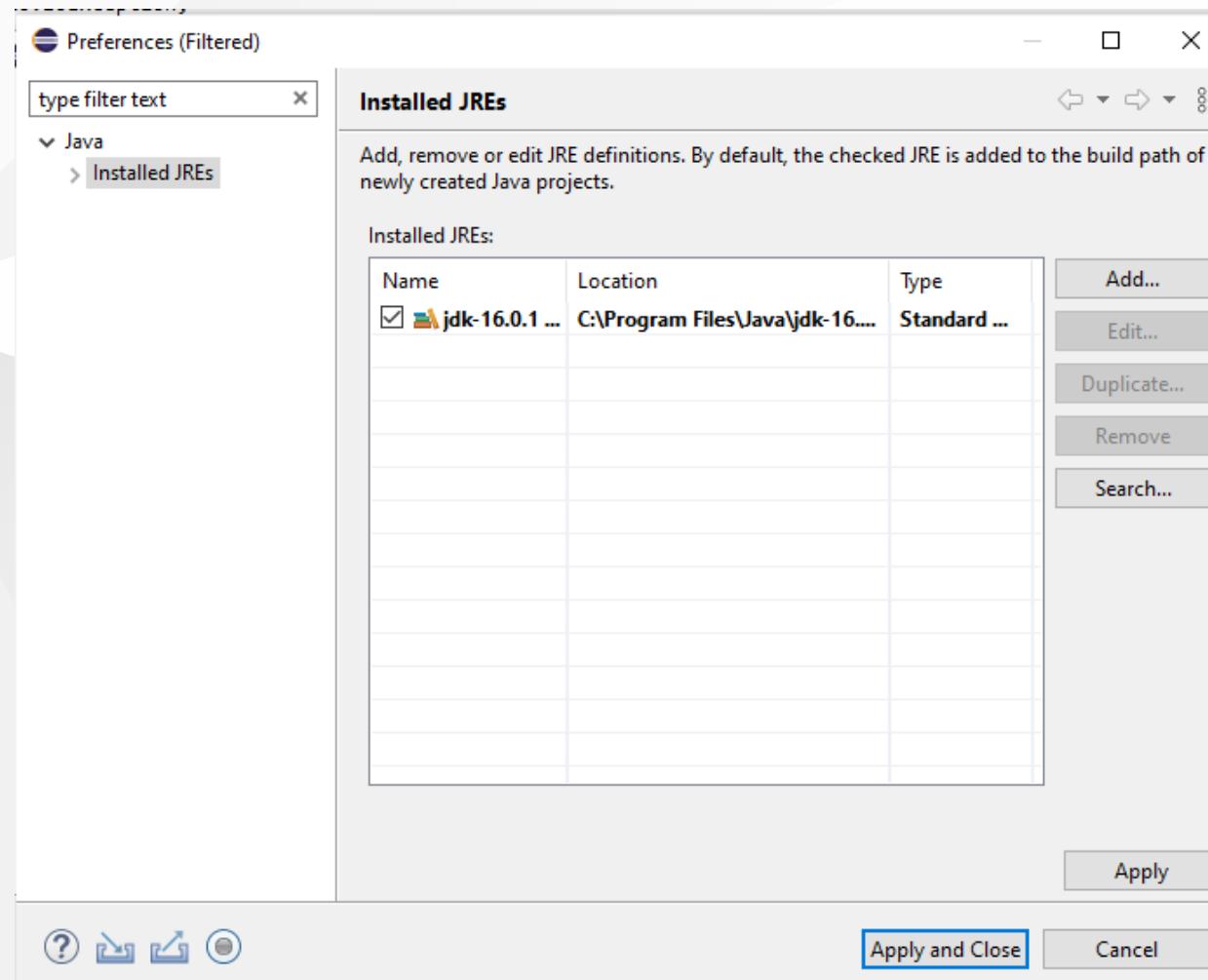
We can found installed and builded JDK for our application from Eclipse setting



# select environments

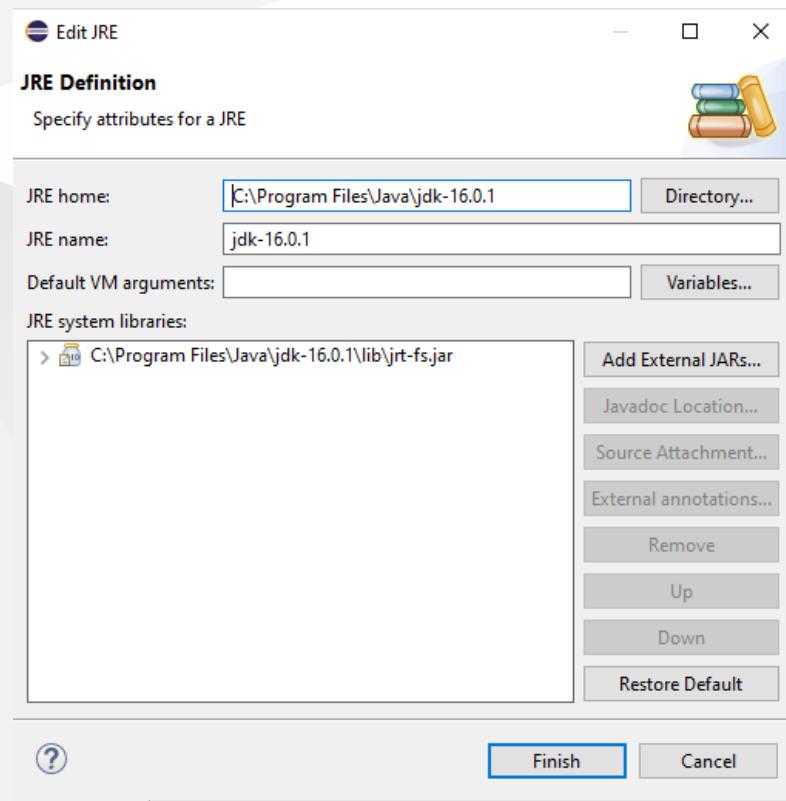


## select installed JRE or JDK

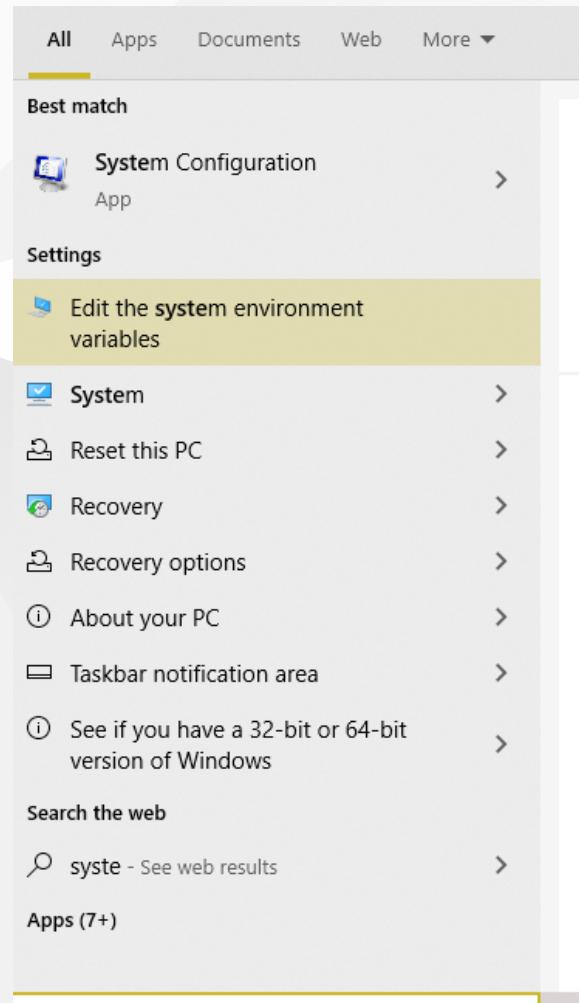


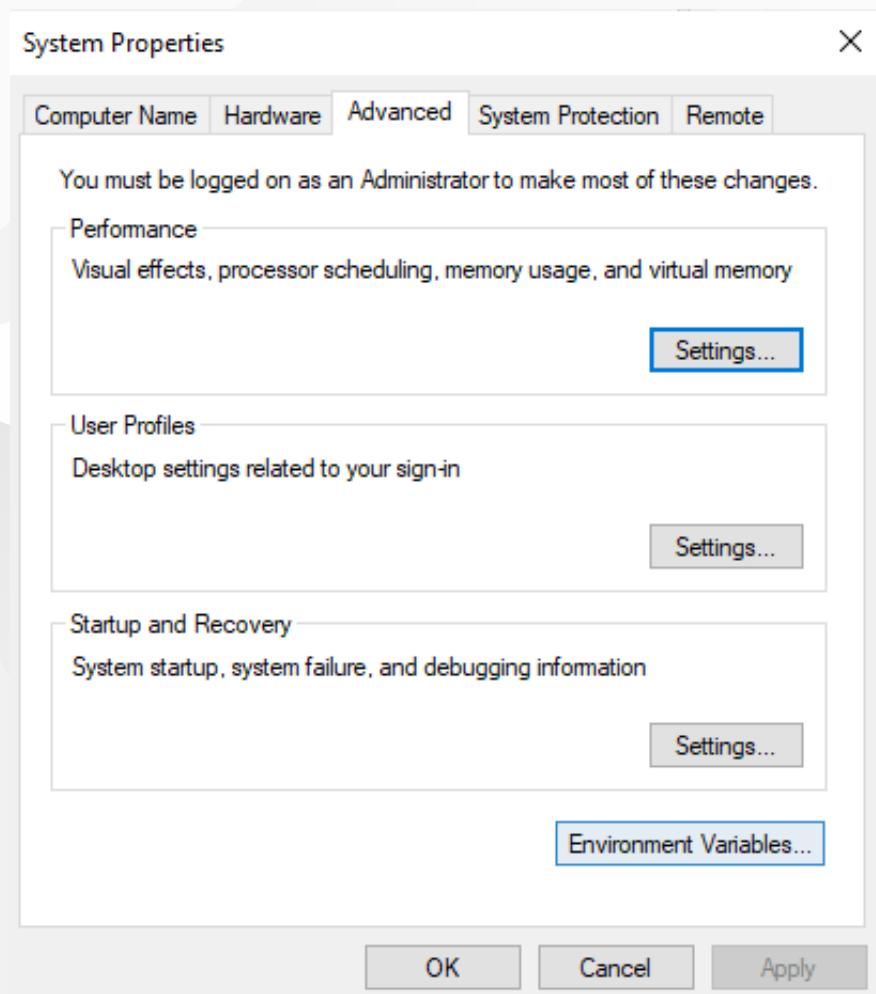
you can see installed JRE or JDK home

C:\Program Files\Java\jdk-16.0.1

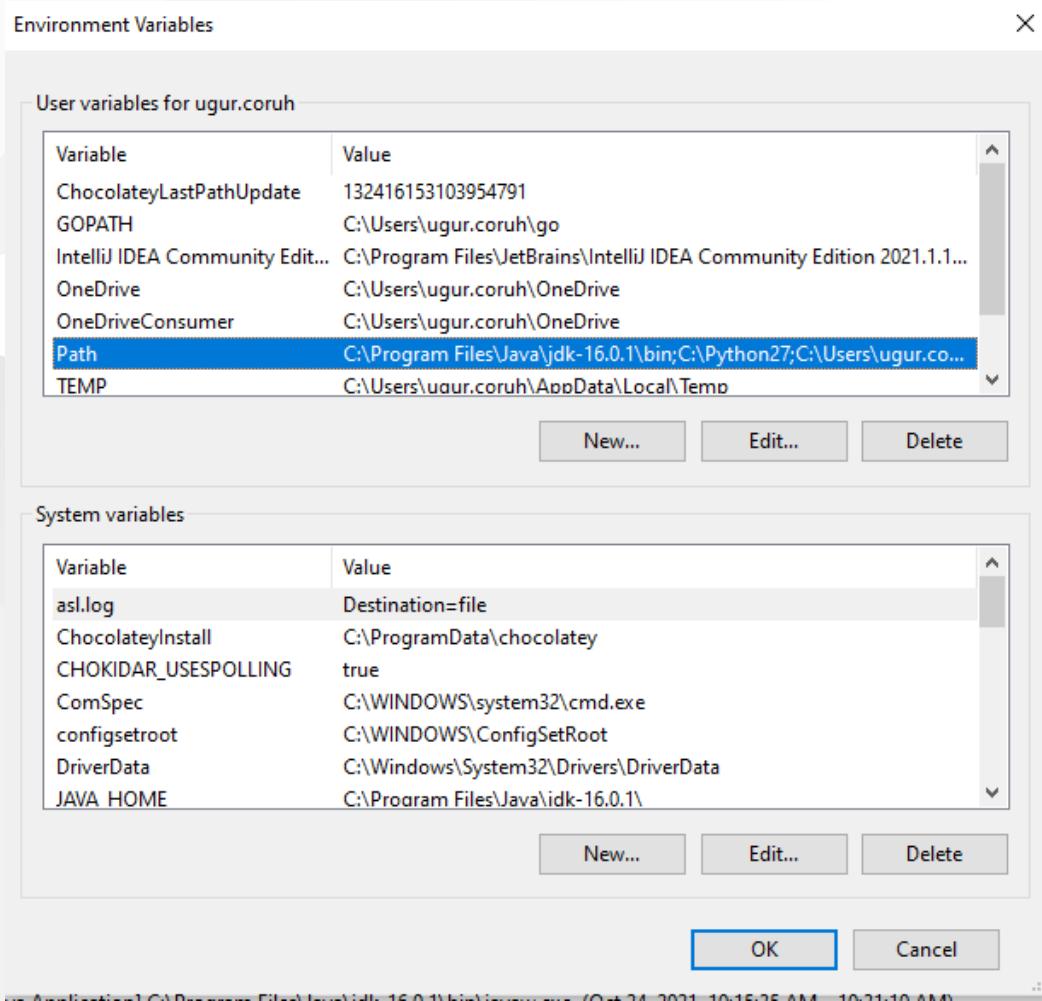


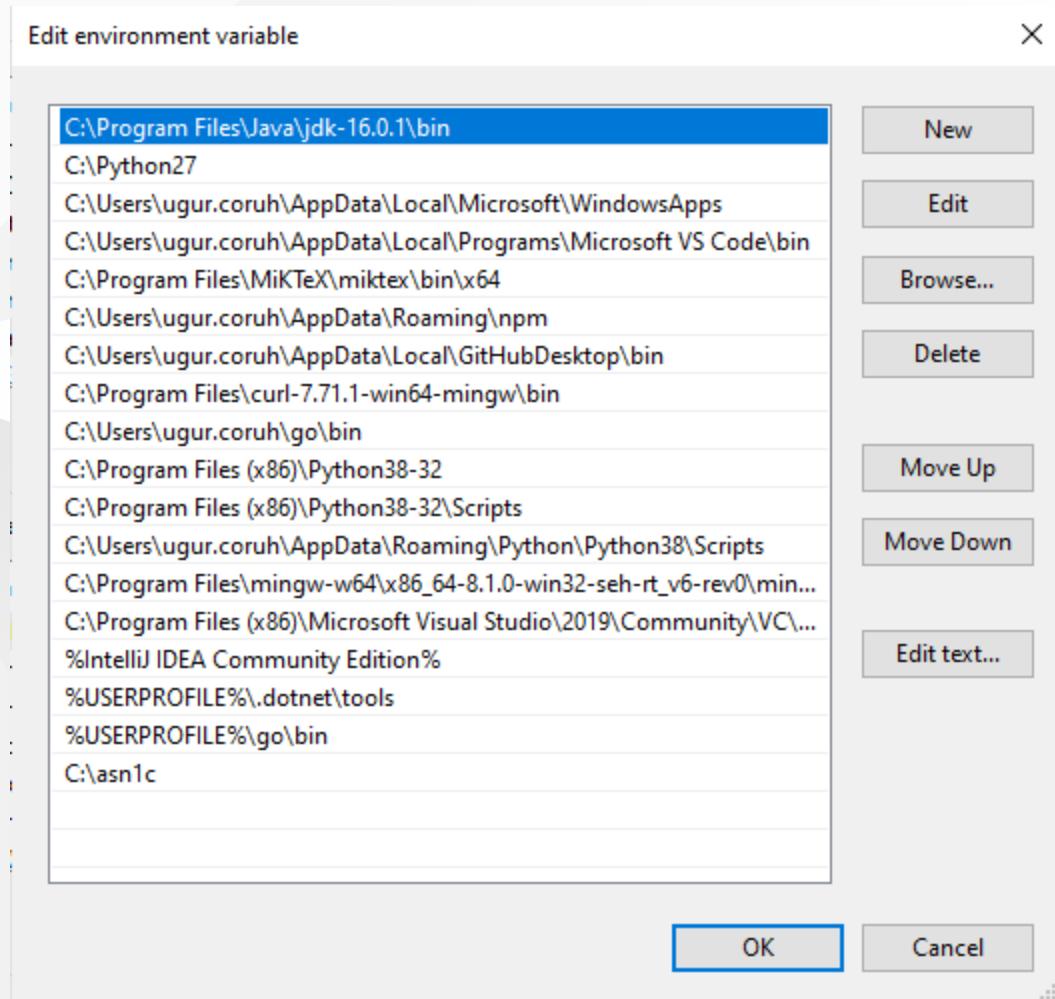
## Open system environment to fix this problem



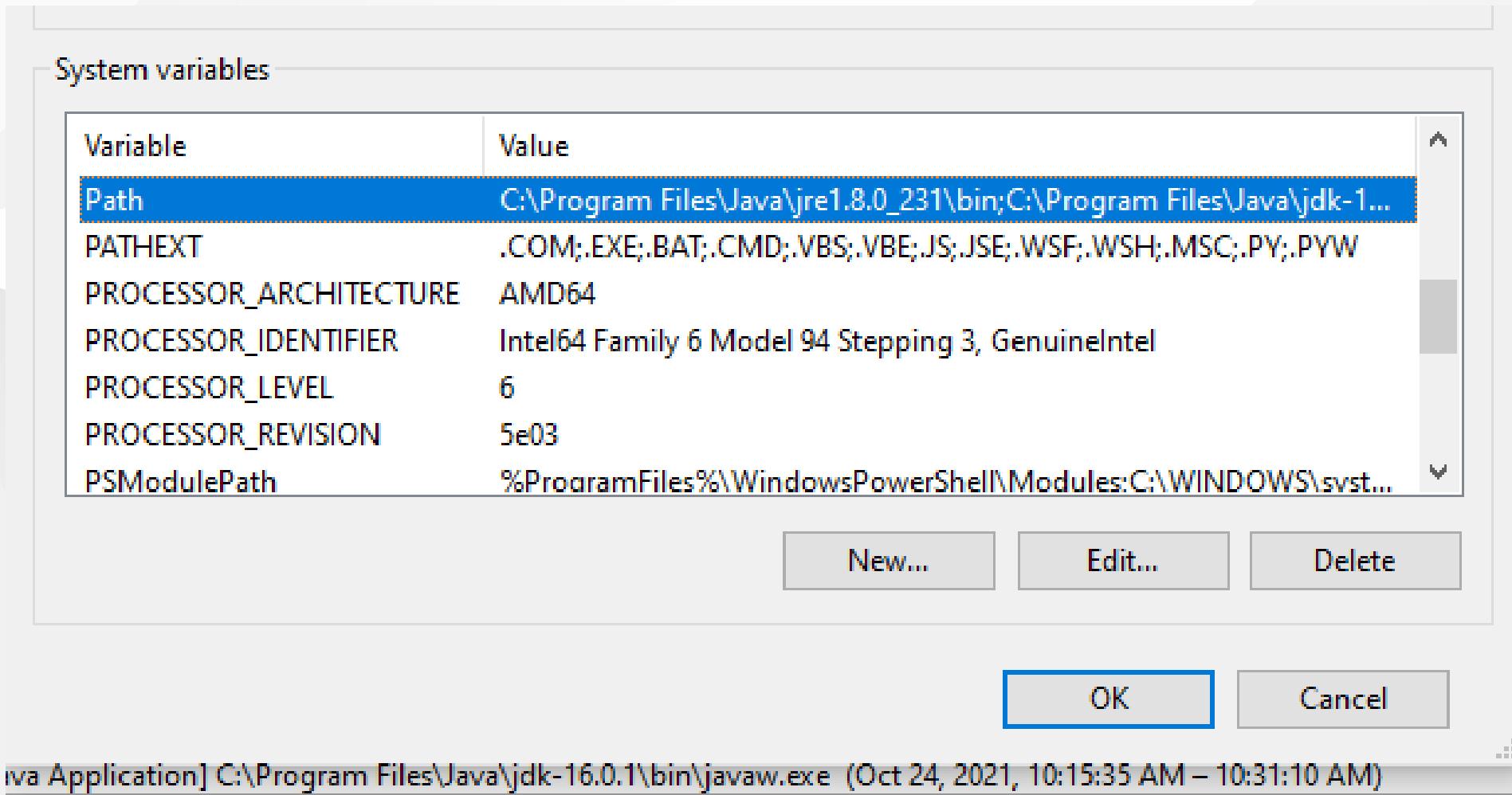


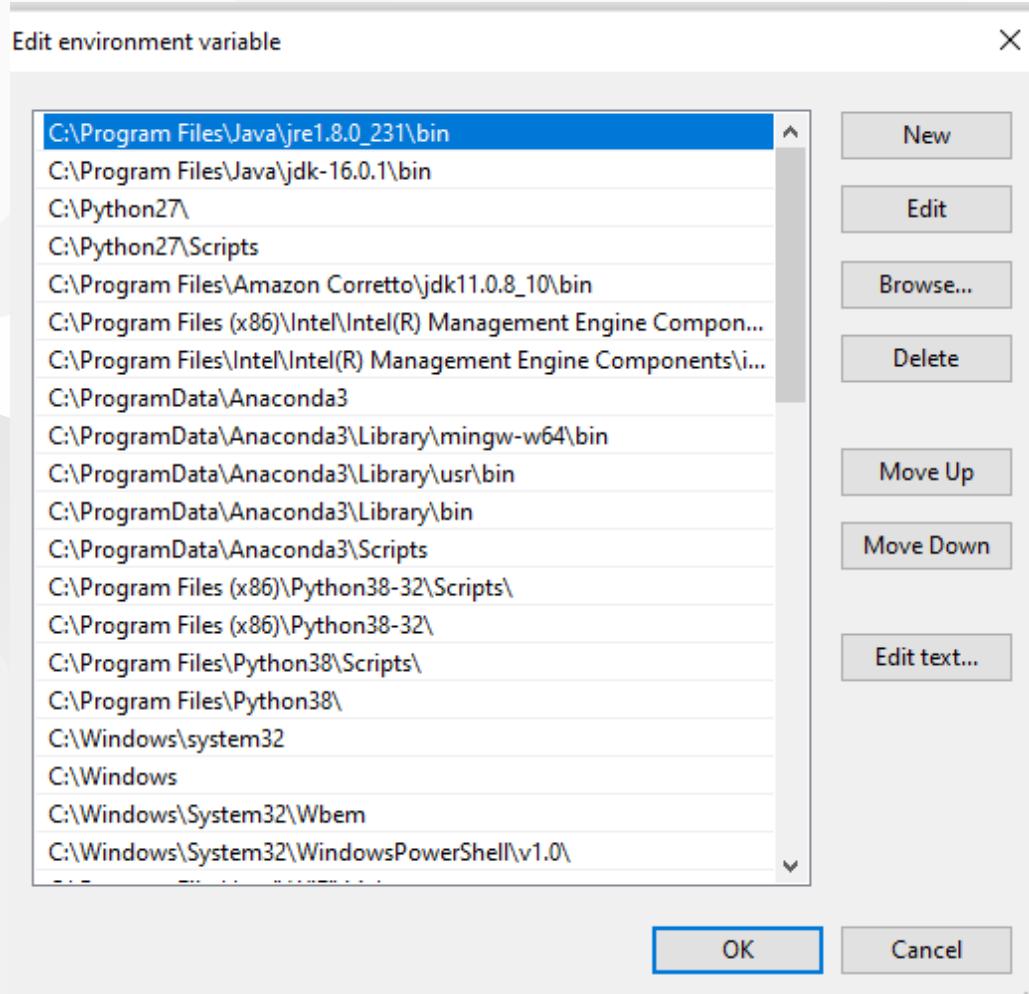
# Check user settings first



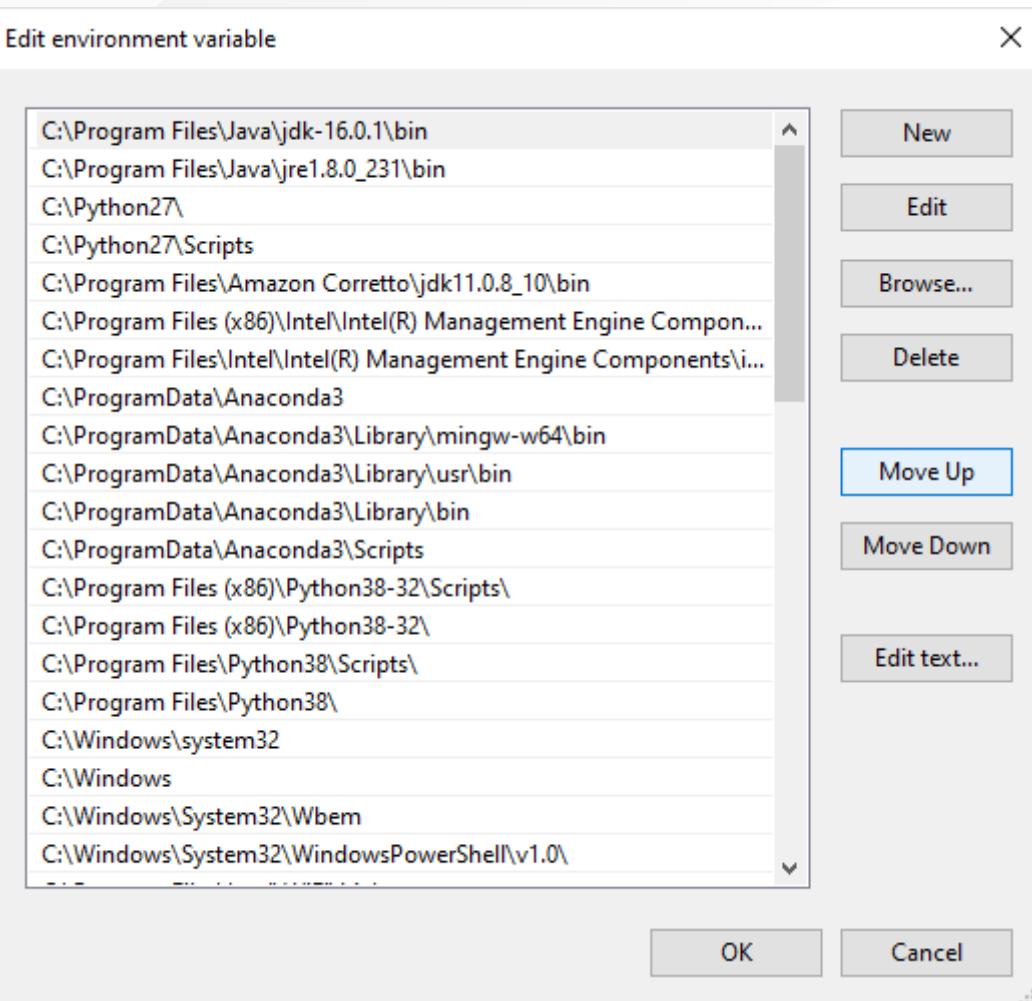


## Check system settings





we will move up the JDK 16 configuration then command line will run first java



Also in system setting check JAVA\_HOME

## System variables

Variable	Value
JAVA_HOME	C:\Program Files\Java\jdk-16.0.1\
MOSQUITTO_DIR	C:\Program Files\mosquitto
NUMBER_OF_PROCESSORS	8
OS	Windows NT

After this settings close current command line and open new one

CE103 Algorithms and Programming I

write

```
java --version
```

if you see java version updated and 16.0.1 then settings are correct

```
C:\WINDOWS\system32\cmd.exe
Microsoft Windows [Version 10.0.19043.1288]
(c) Microsoft Corporation. All rights reserved.

C:\Users\ugur.coruh>java --version
java 16.0.1 2021-04-20
Java(TM) SE Runtime Environment (build 16.0.1+9-24)
Java HotSpot(TM) 64-Bit Server VM (build 16.0.1+9-24, mixed mode, sharing)

C:\Users\ugur.coruh>
```

and now if we enter and run application as follow we will see output

```
C:\Users\ugur.coruh>cd Desktop  
  
C:\Users\ugur.coruh\Desktop>cd java-export-sample  
  
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleLibExecutable.jar  
Hello World!  
Hello There  
Results is 9  
Results is 9
```

CE103 Algorithms and Programming I  
But when you click this jar its not running as you see so we have options to provide a clickable application there

Launch4j is an option here

## Launch4j - Cross-platform Java executable wrapper

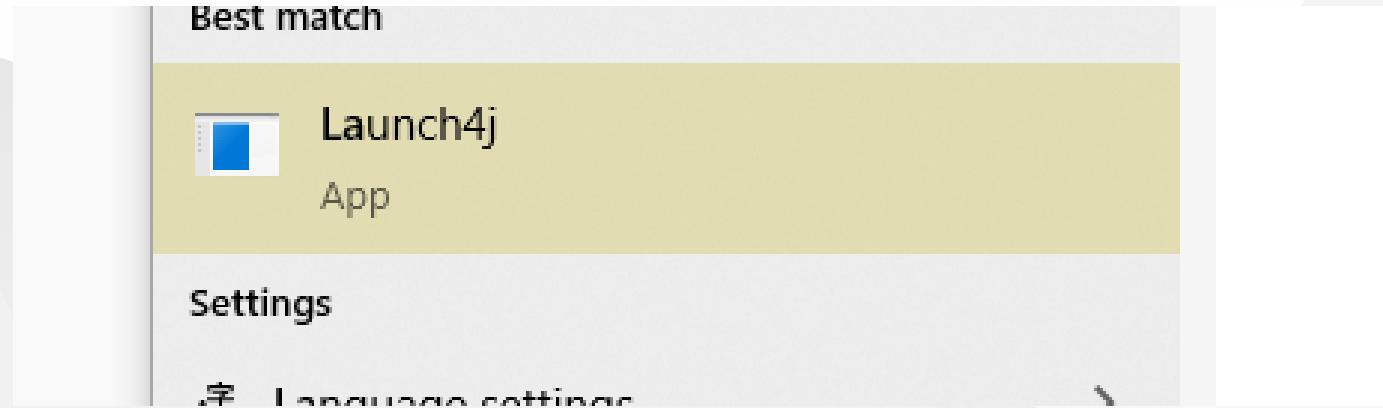


you can watch this tutorial also

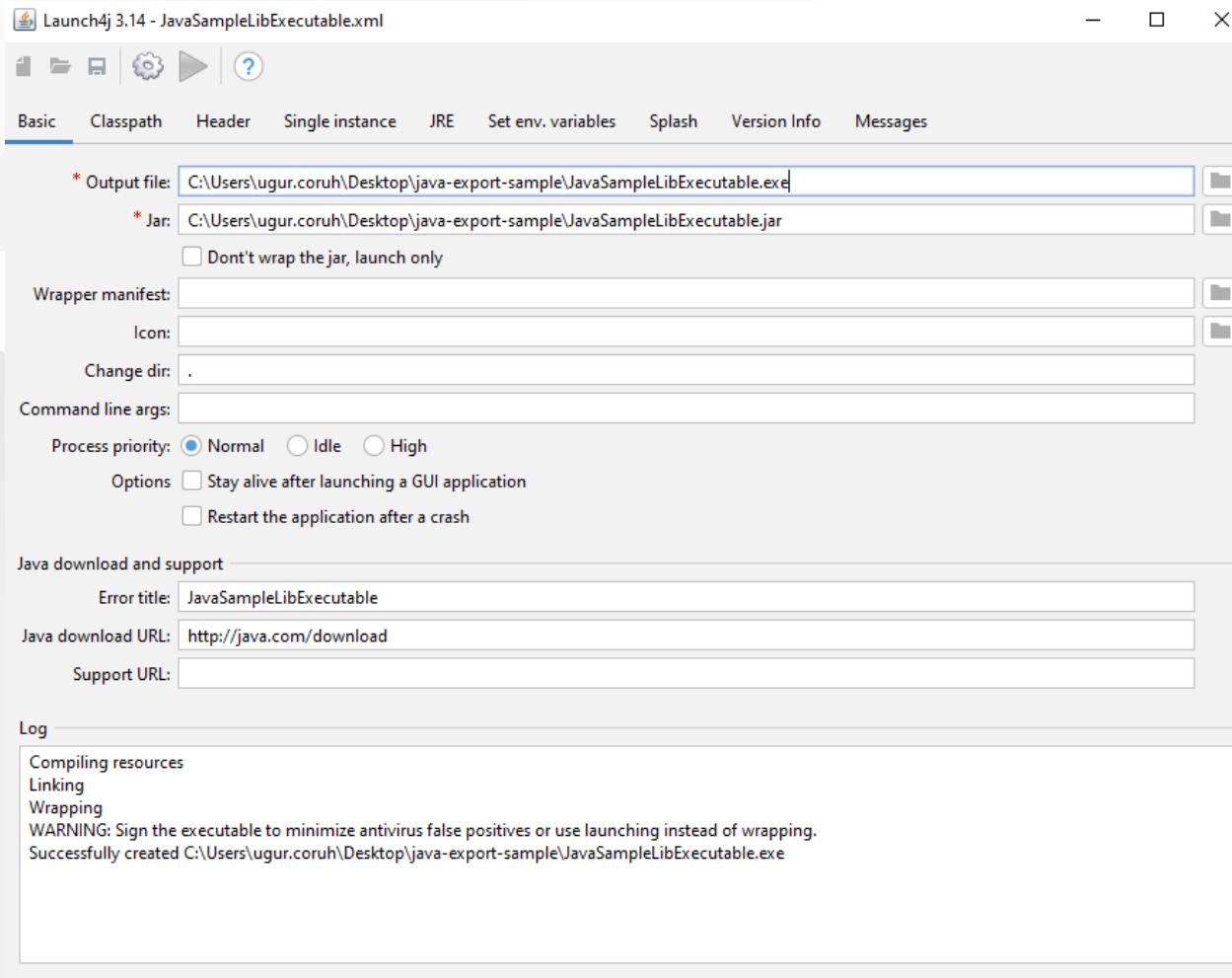
[How to convert jar to exe using Launch4J Full explanation - YouTube](#)



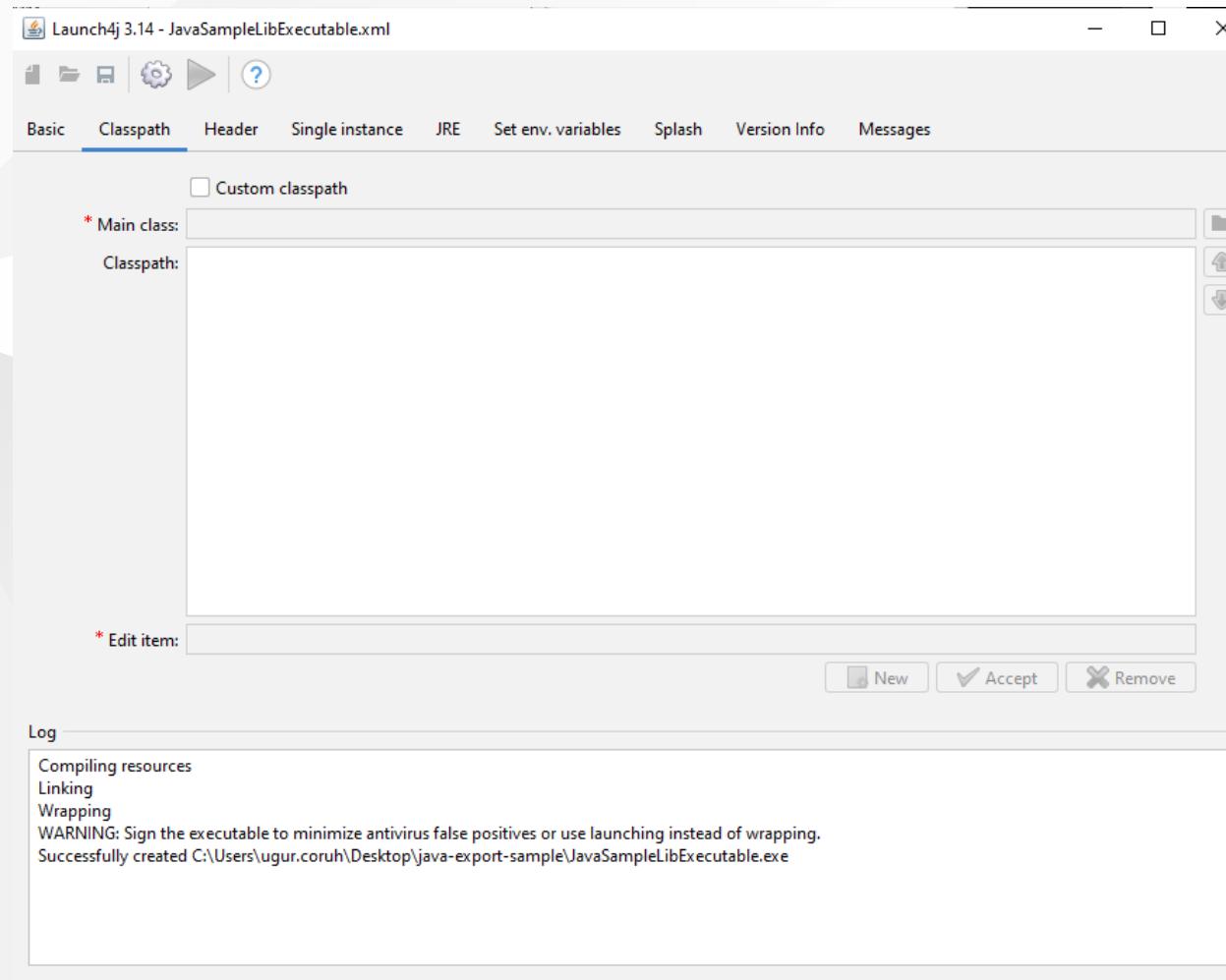
Download and install launch4j and open application



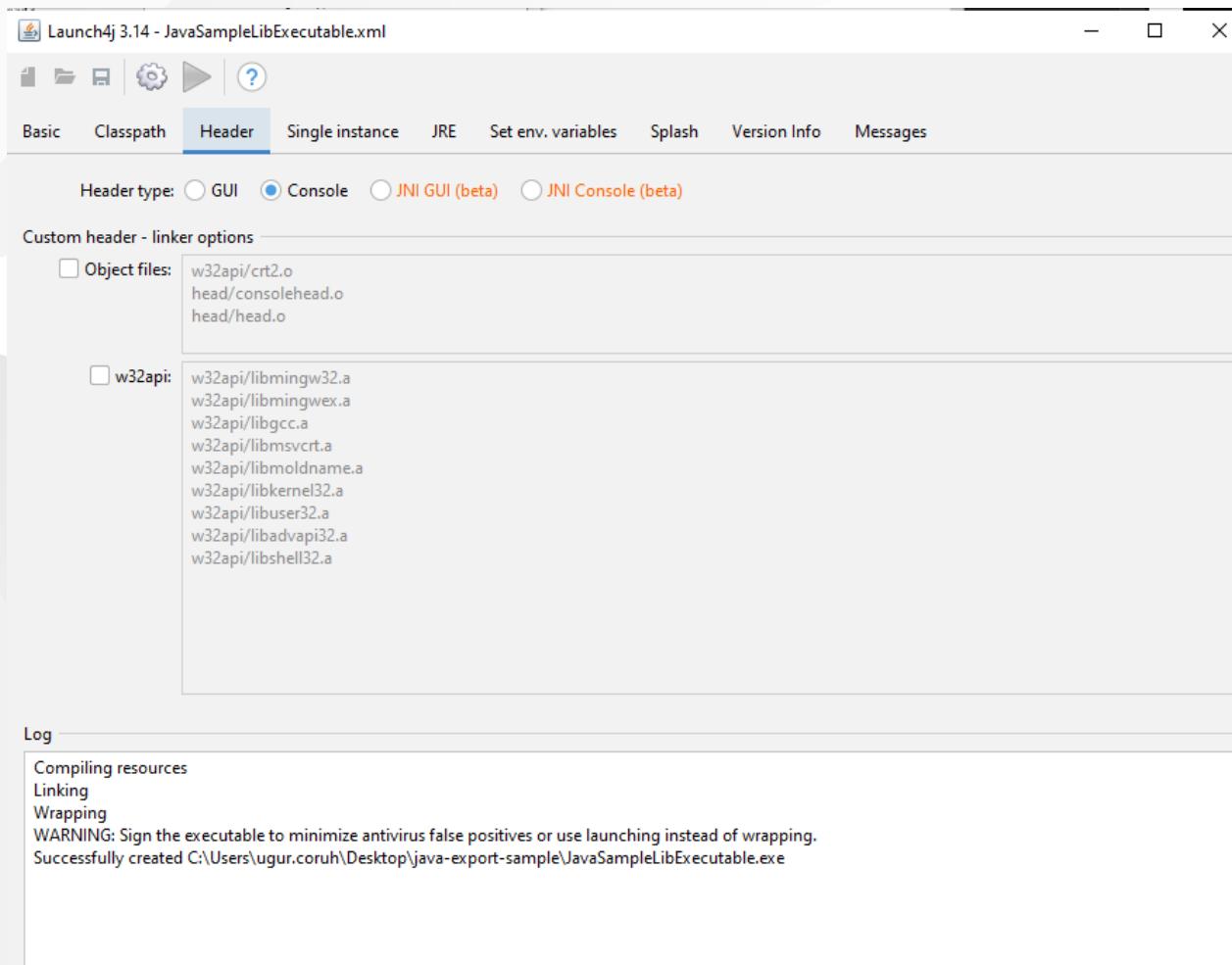
configure your application settings similar to below select jar file and exe output path



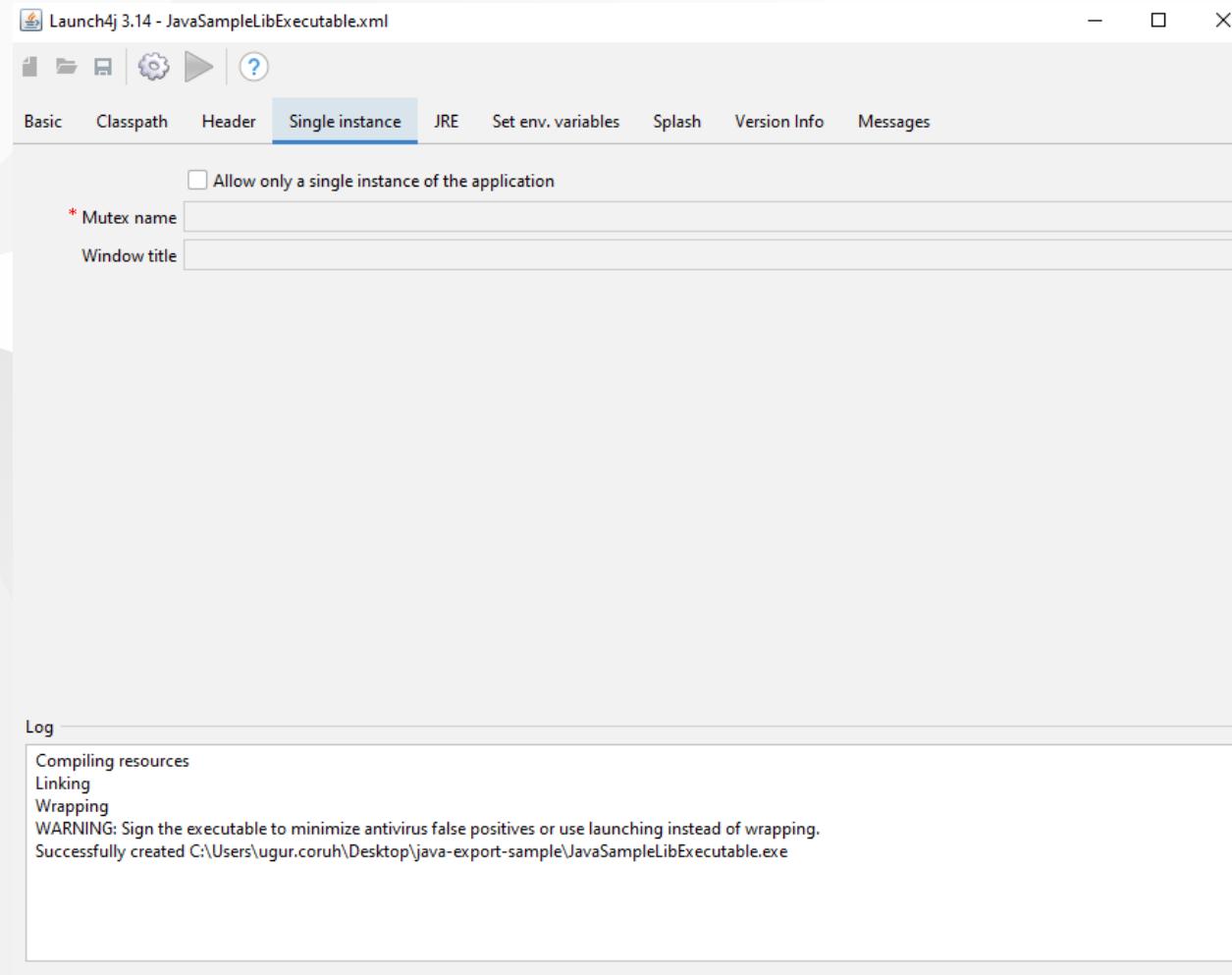
we can customize main class if have multiple main class



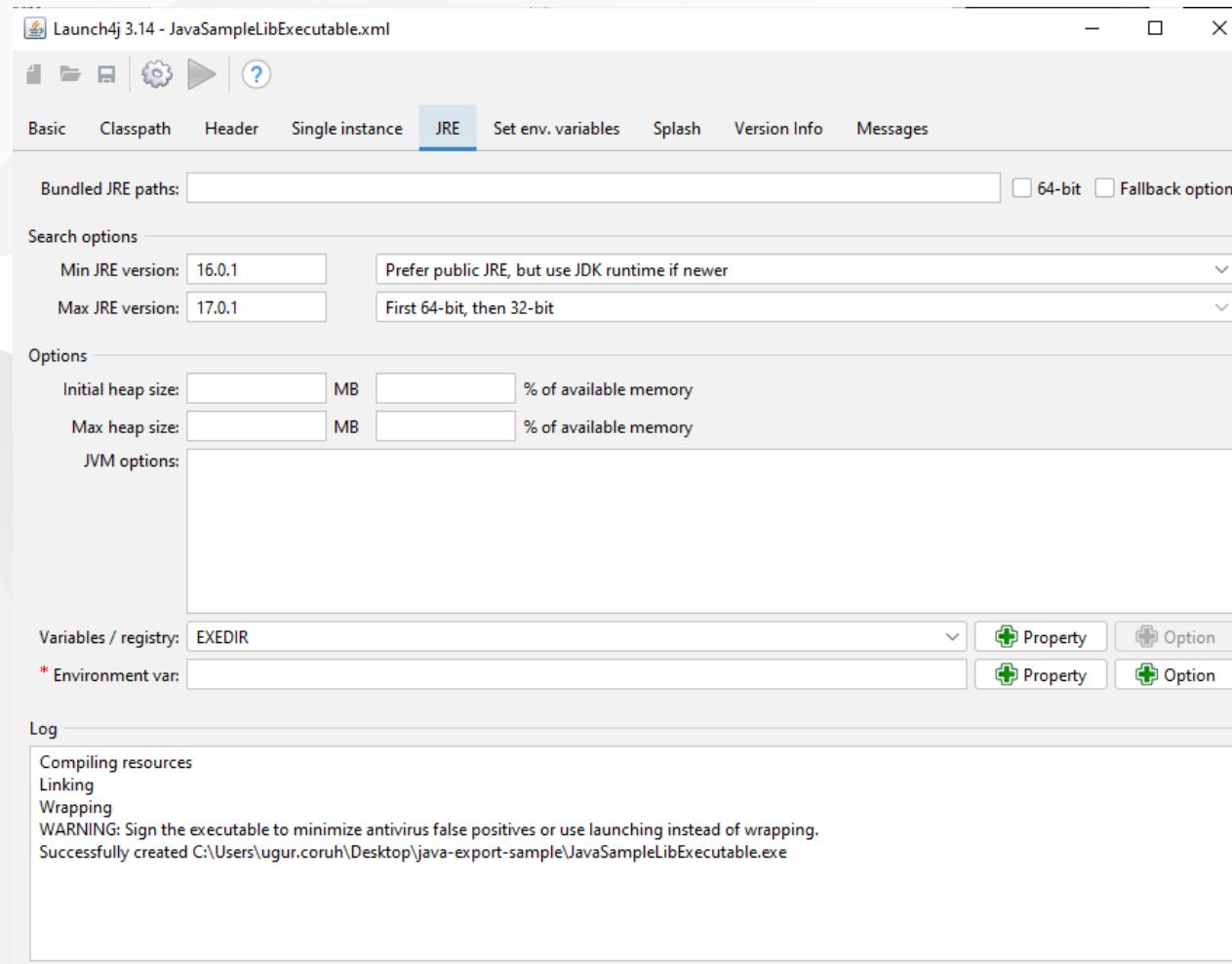
## select console from setting for this application



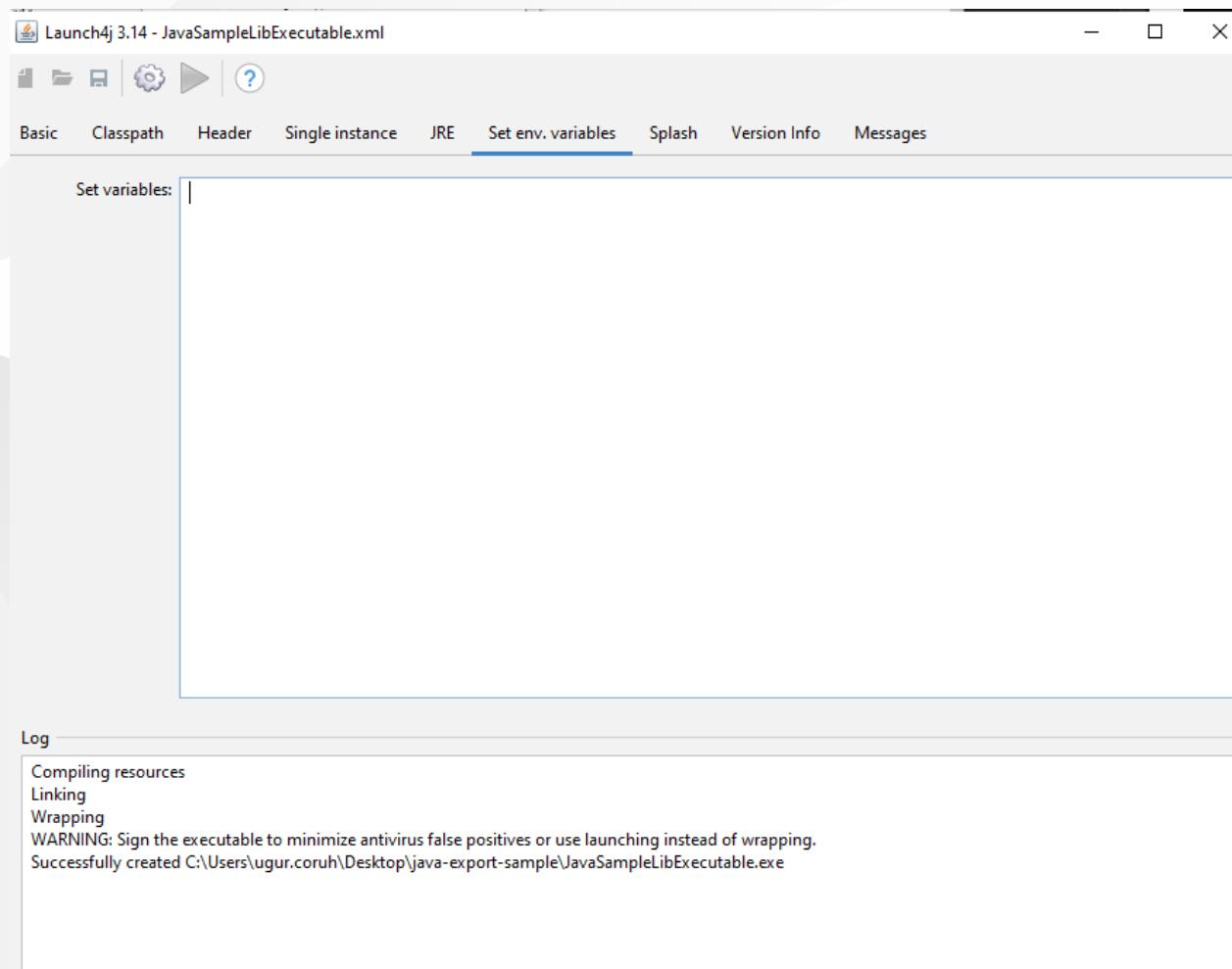
we can provide a single running application, this setting avoid to run multiple instances



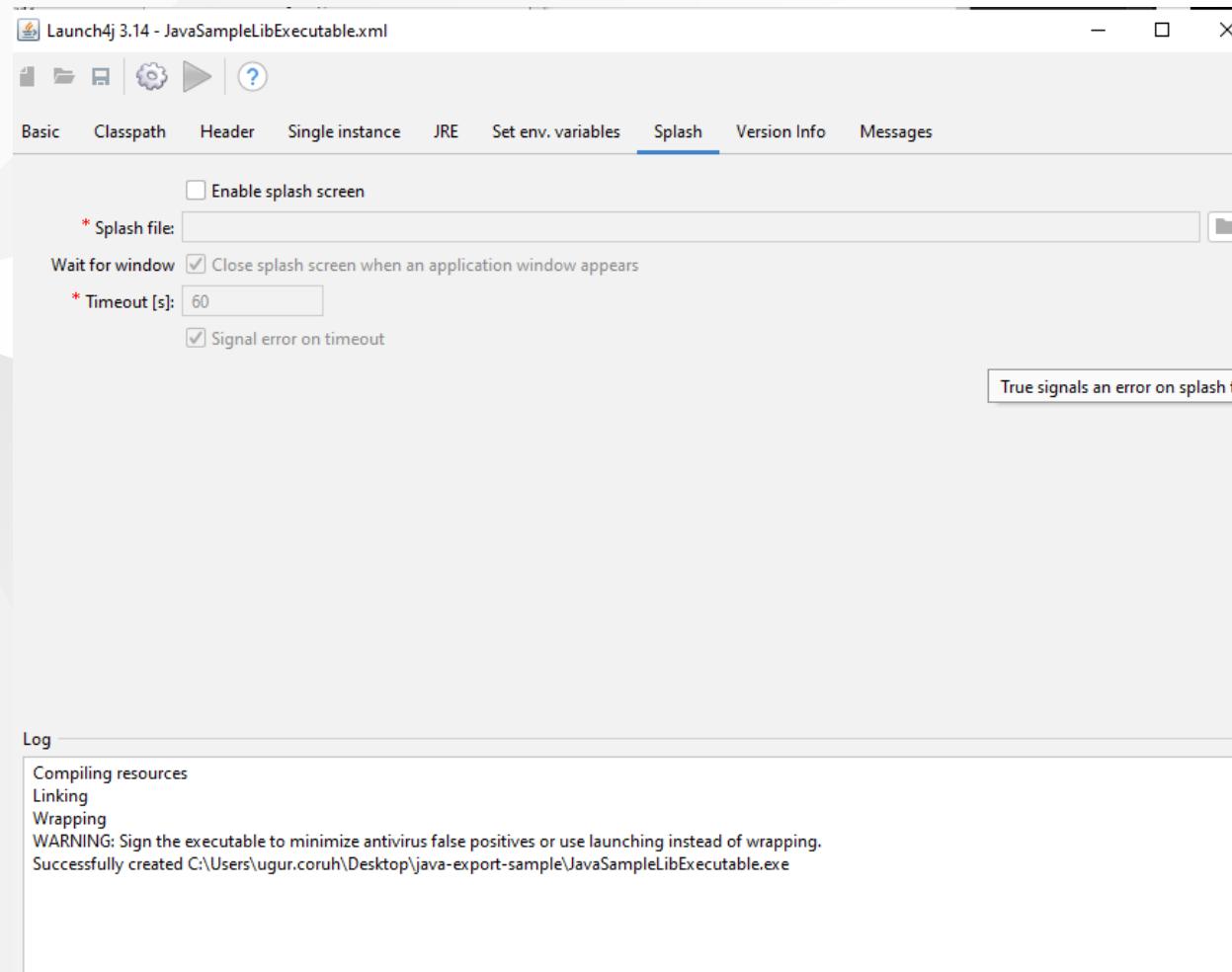
# we need to set runtime environment versions



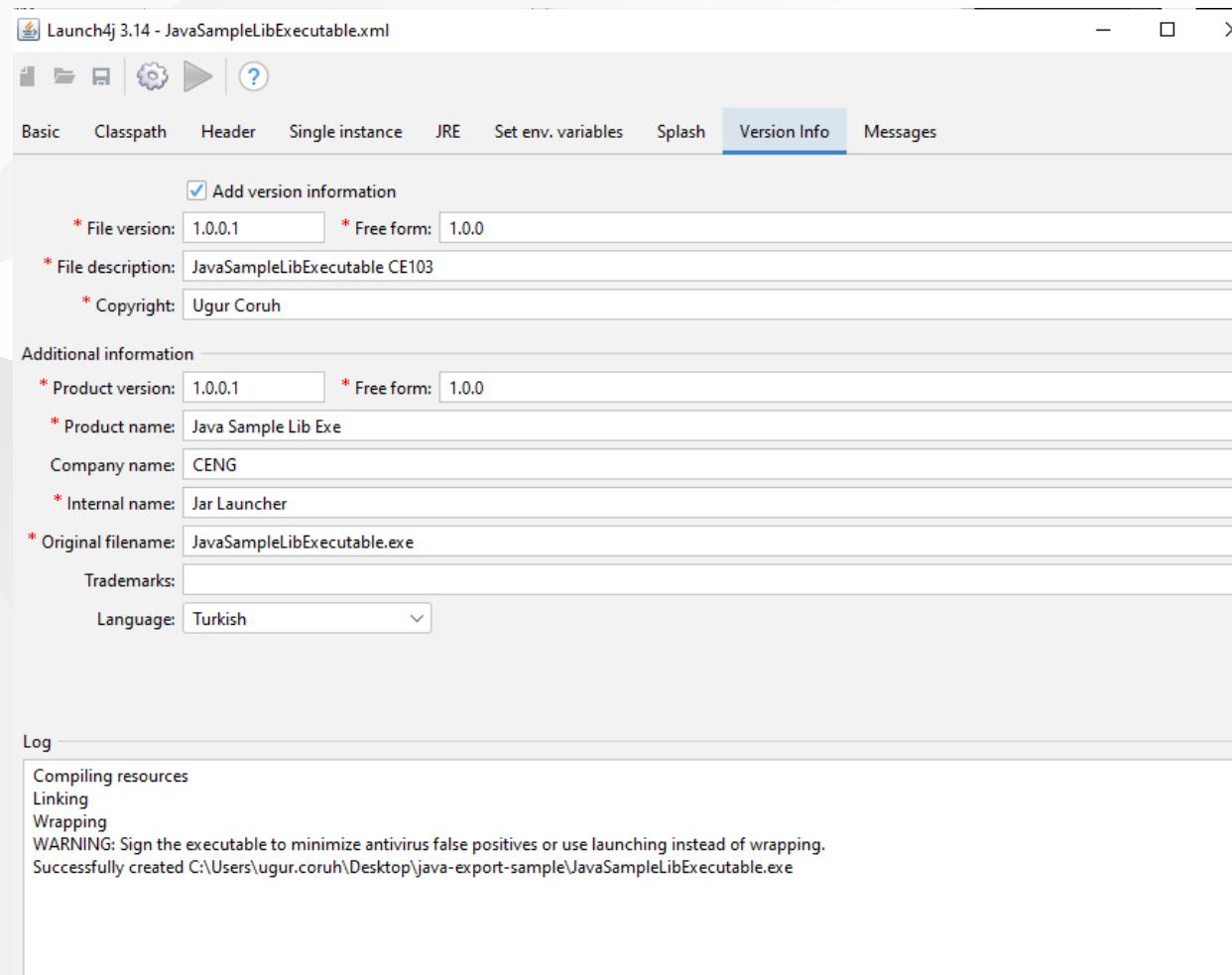
you can set system parameters before running application



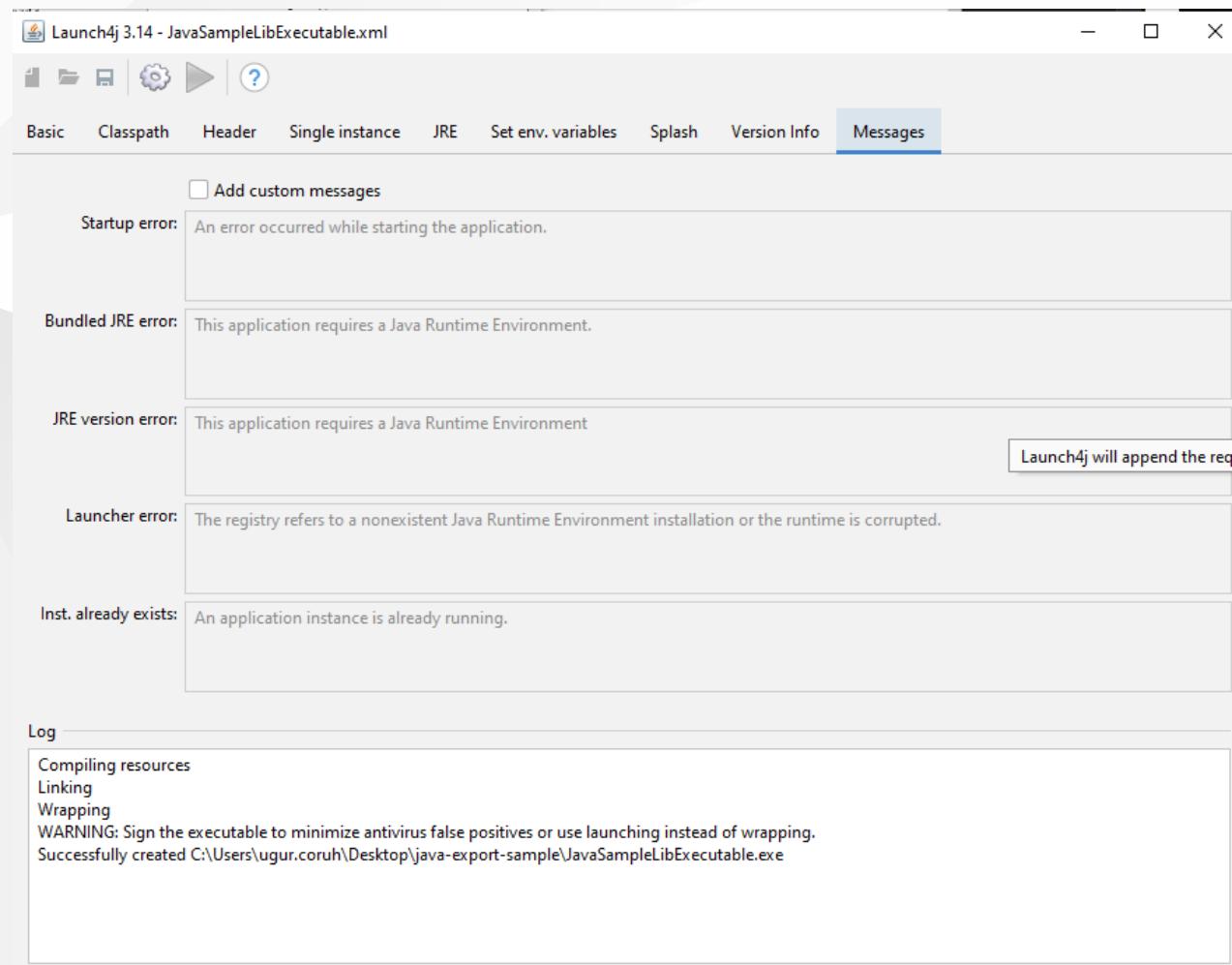
with splash screen you can show a splash screen image for your application



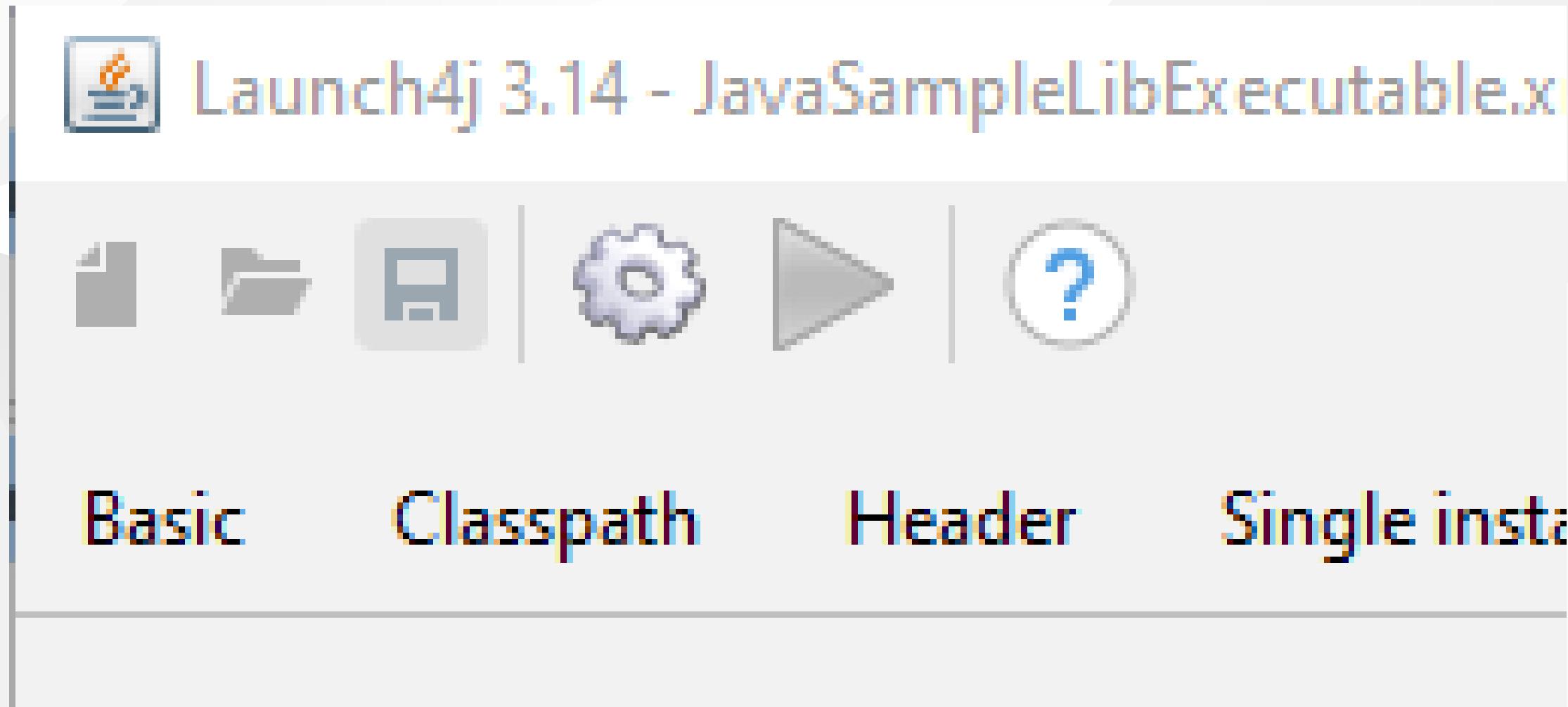
# File attributes such as version product information is configured from version info tab



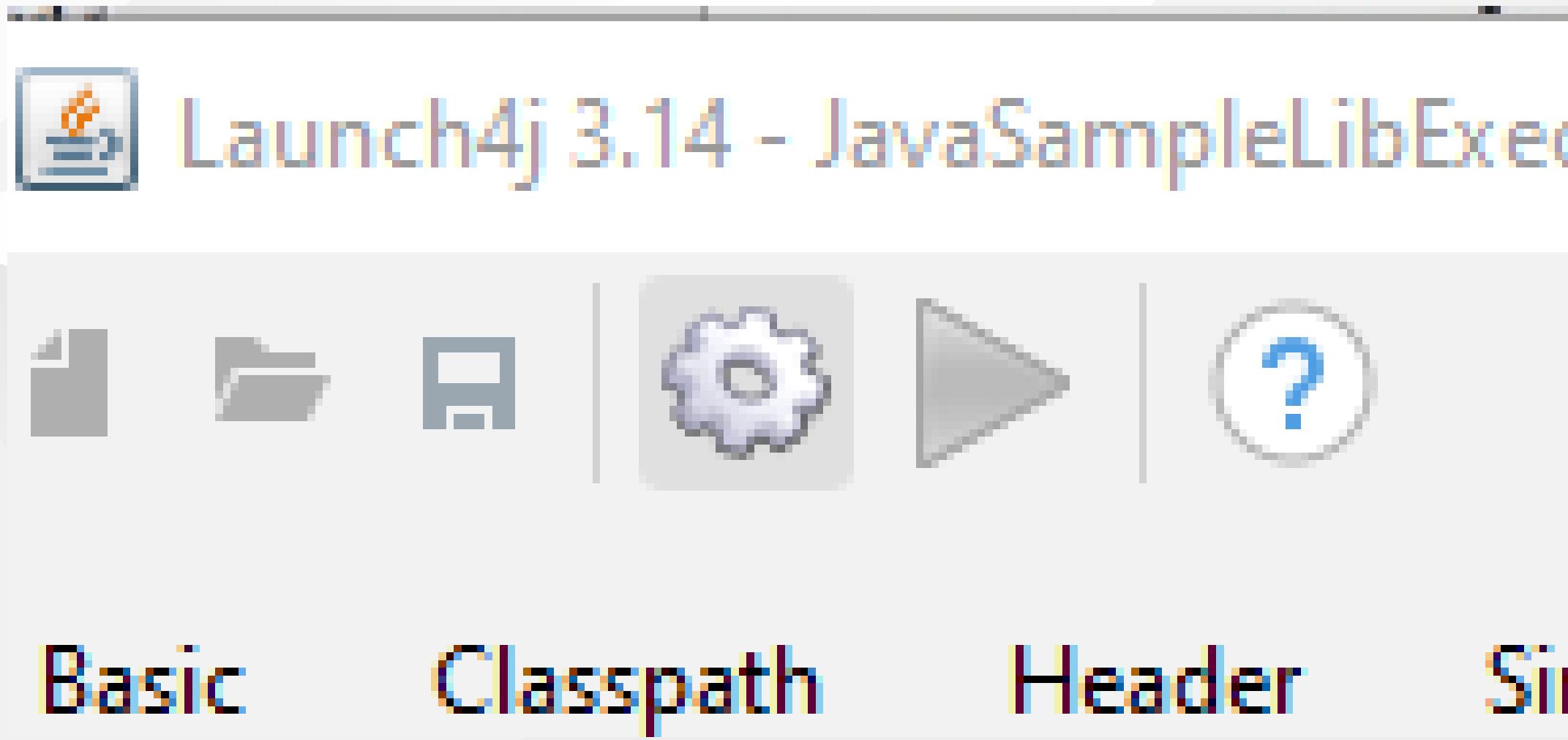
if your application runtime condition has an error then you can show this customized messages also



with this options save configuration file xml



and compile settings



you will see generated output file in log screen

```
Compiling resources
```

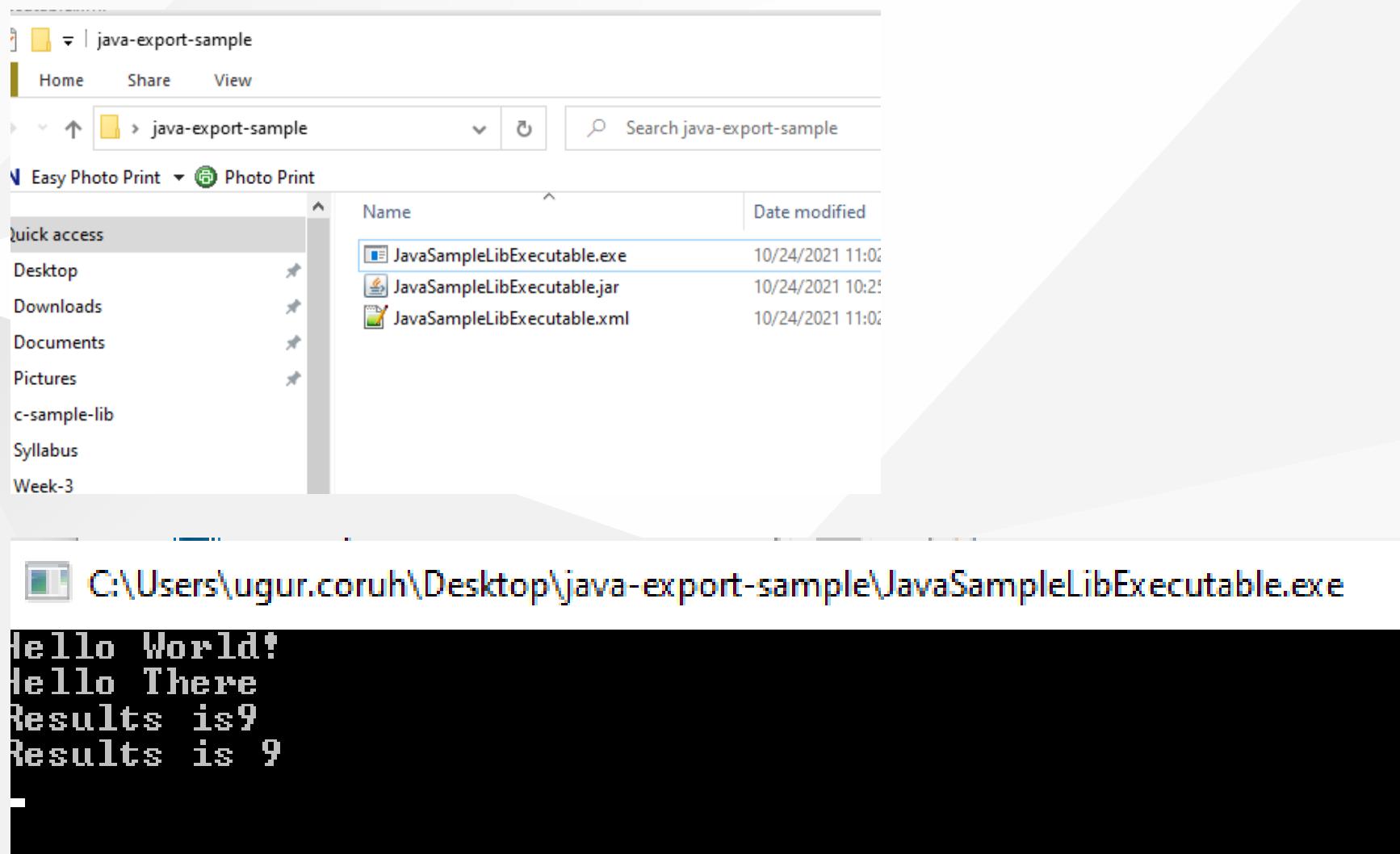
```
Linking
```

```
Wrapping
```

```
WARNING: Sign the executable to minimize antivirus false positives or use launching instead of wrapping.
```

```
Successfully created C:\Users\ugur.coruh\Desktop\java-export-sample\JavaSampleLibExecutable.exe
```

now we can run exe by click

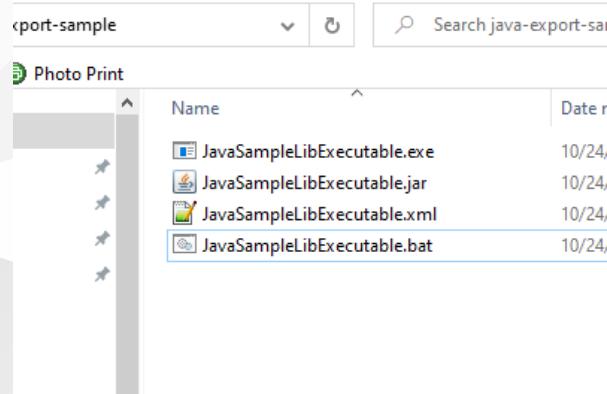


another option here adding a bat file to run current jar file



## JavaSampleLibExecutable.bat

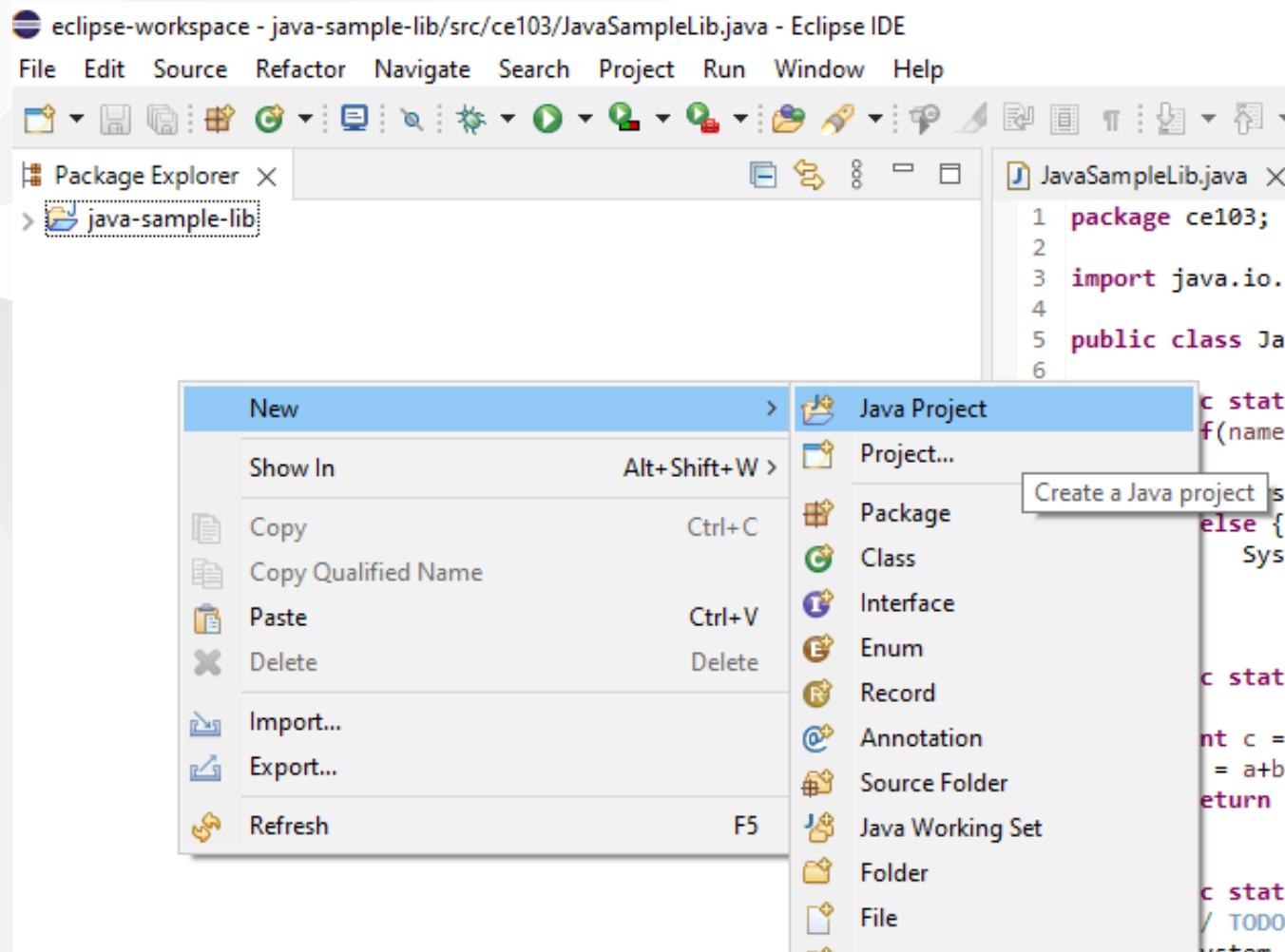
```
java -jar JavaSampleLibExecutable.jar
```

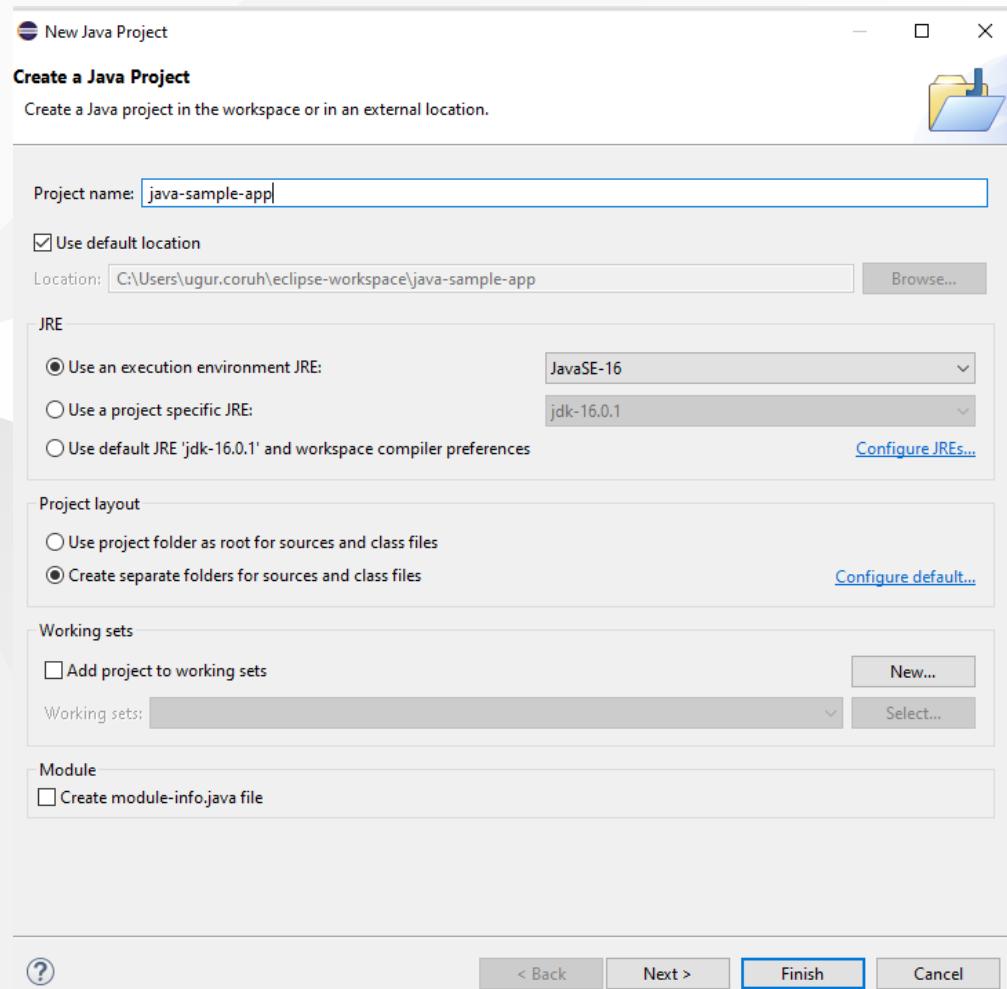


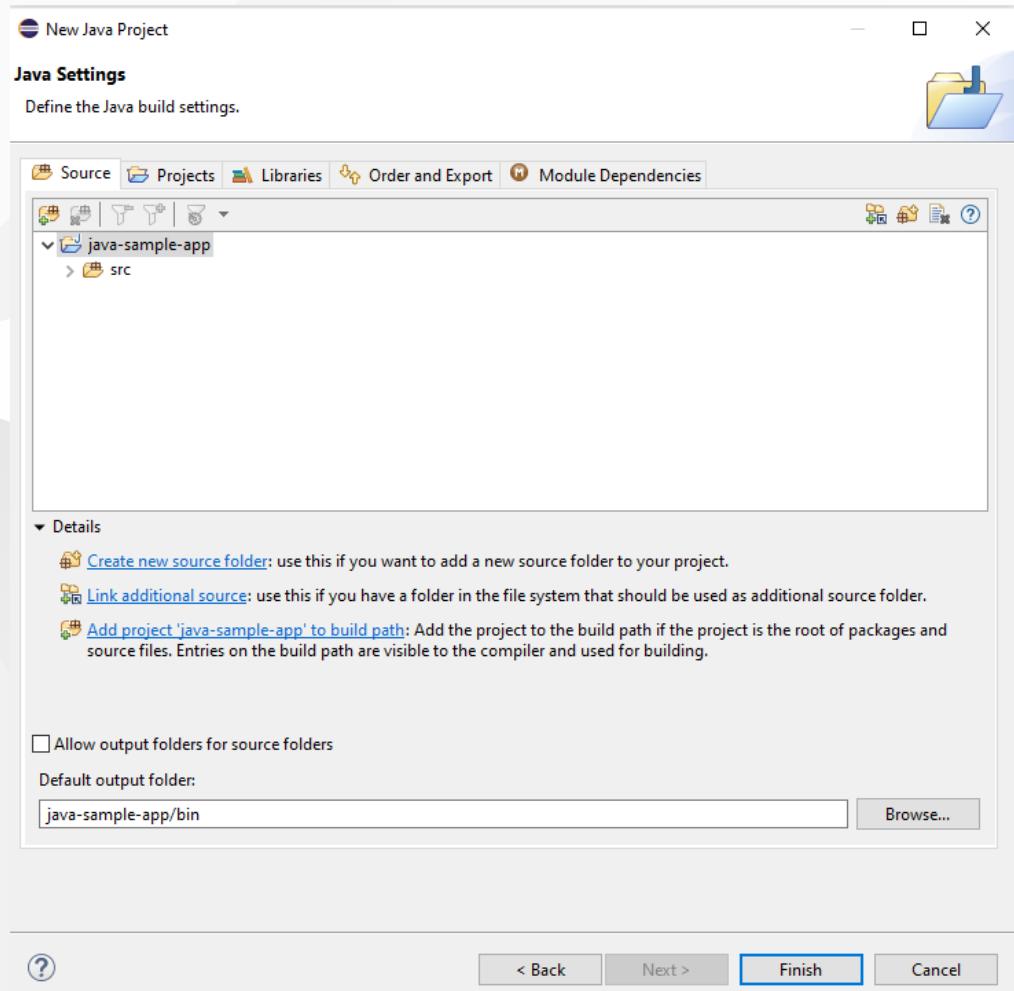
if we click bat file then we will automate command line task for current jar file

```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleLibExecutable.jar
Hello World!
Hello There
Results is 9
Results is 9
```

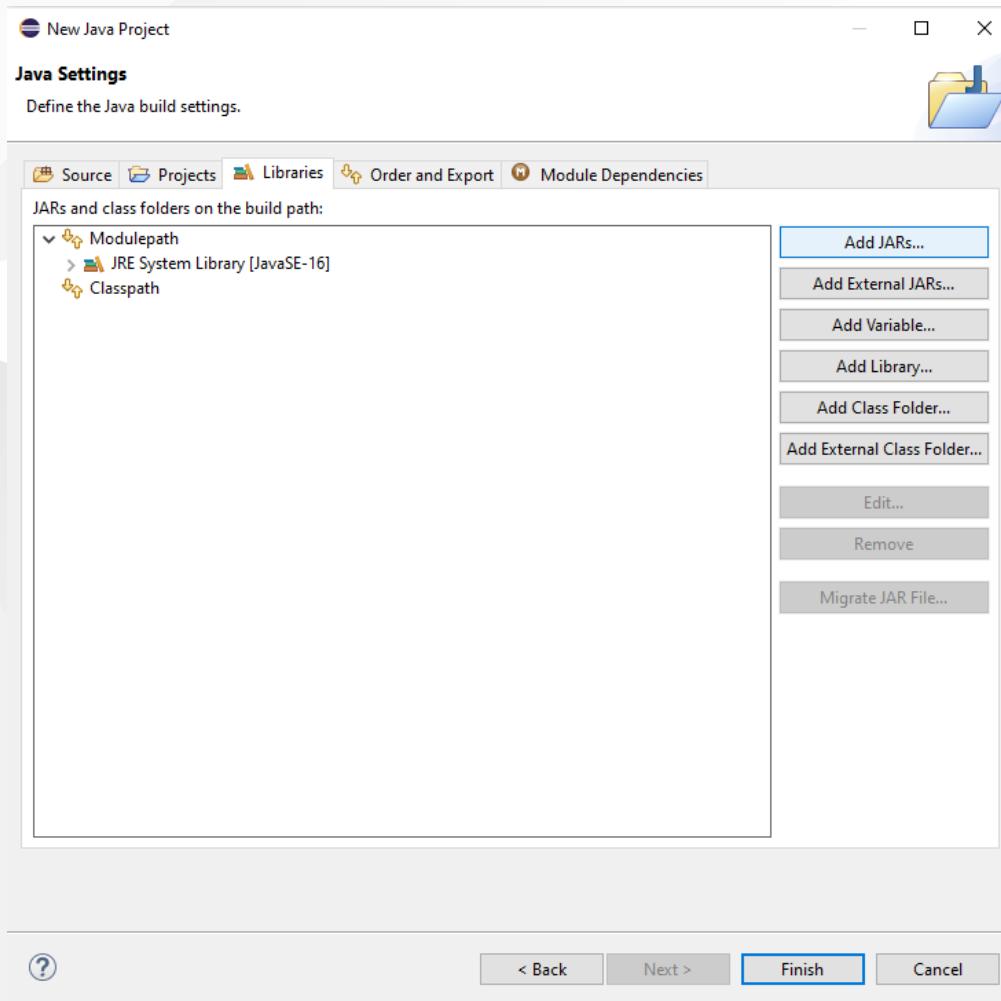
Now return back to our java library and create another console application that use library functions







you can set libraries in this step from but our library should exported for our solution

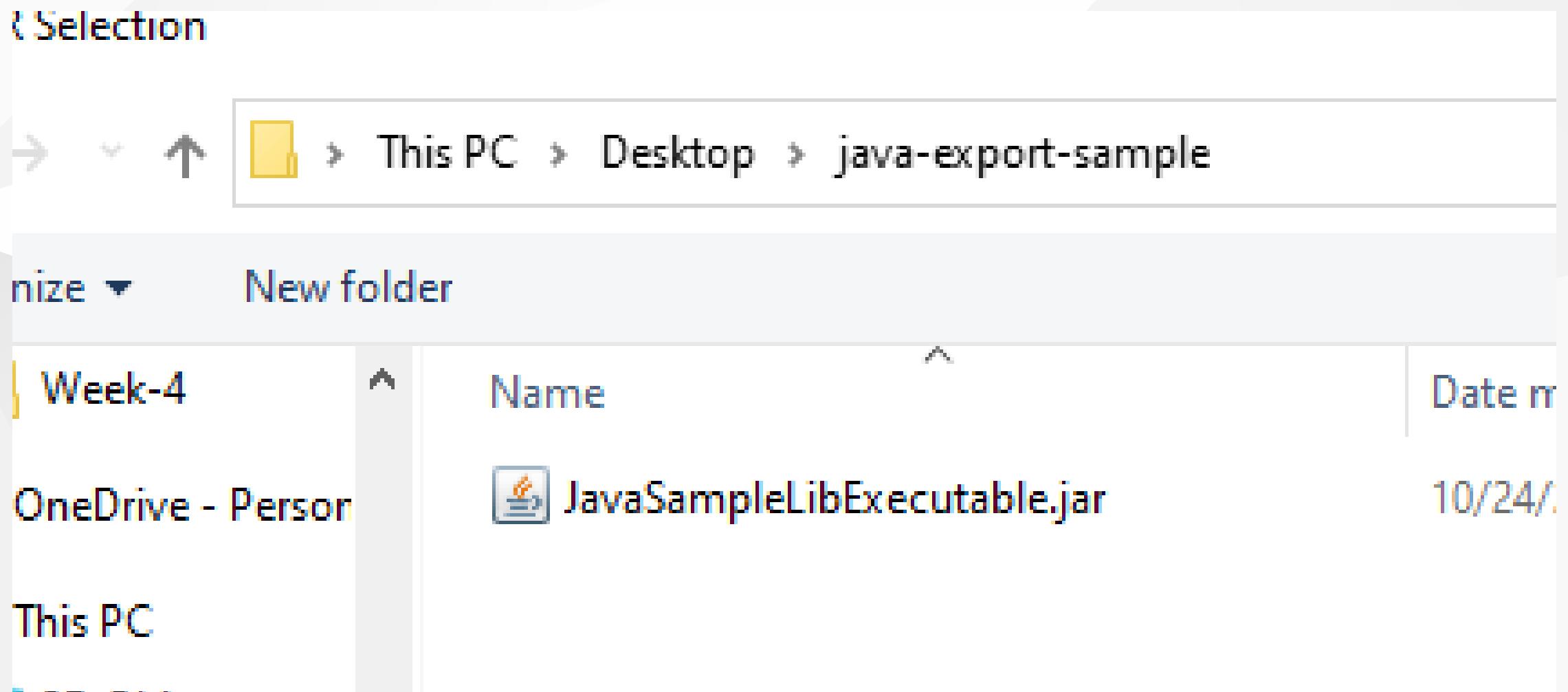


Select Add External JARs...

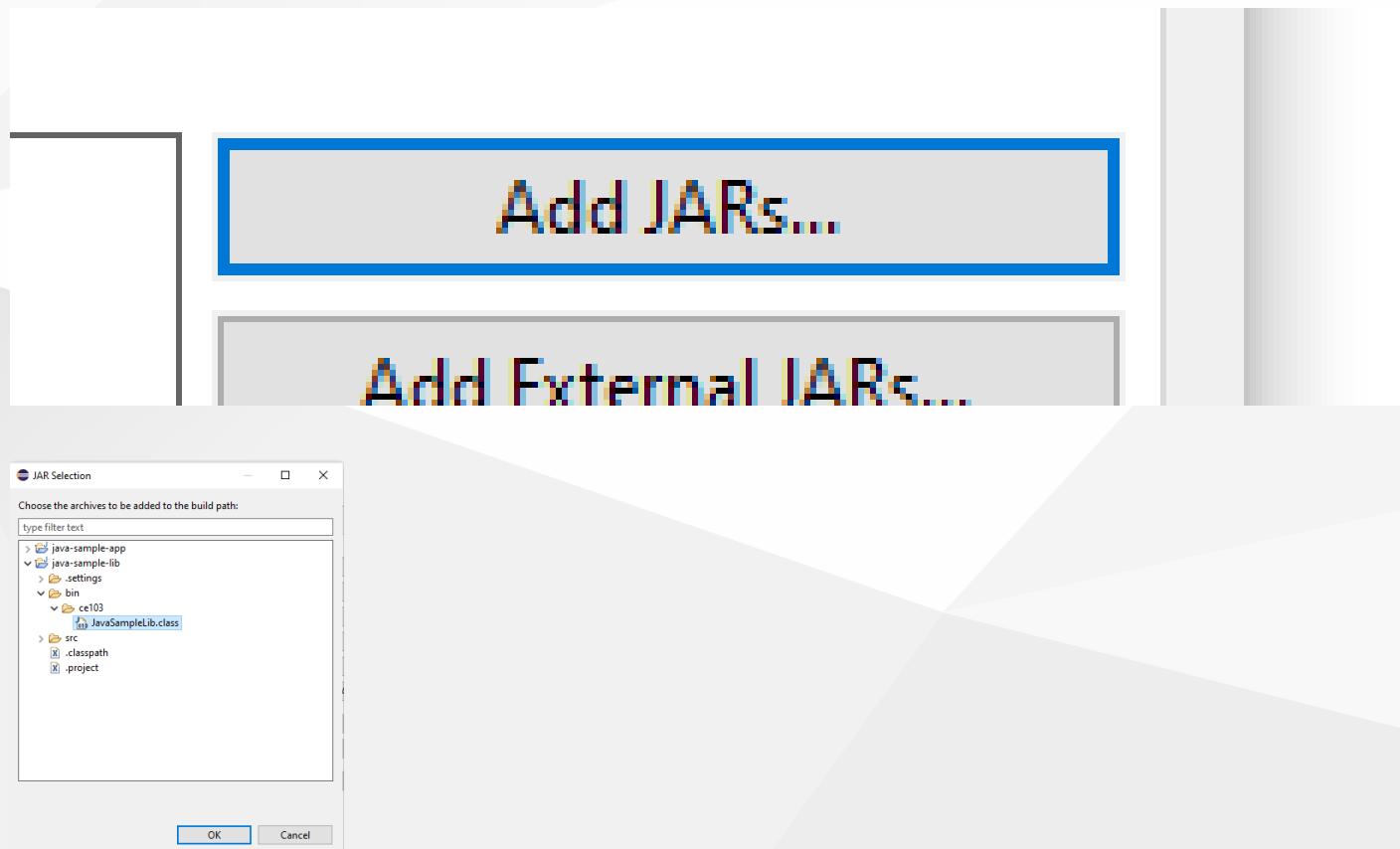
Add External JARs...

Add Variables

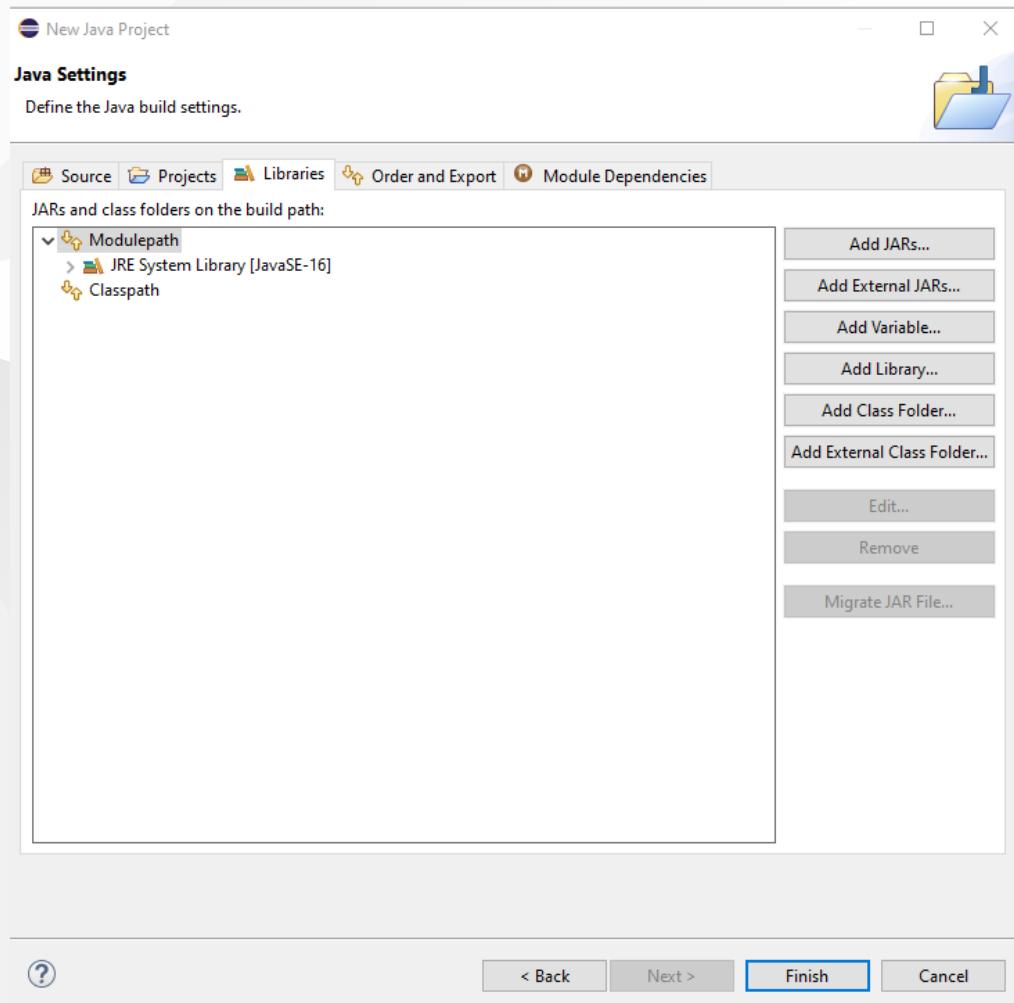
Open Exported jar folder and select



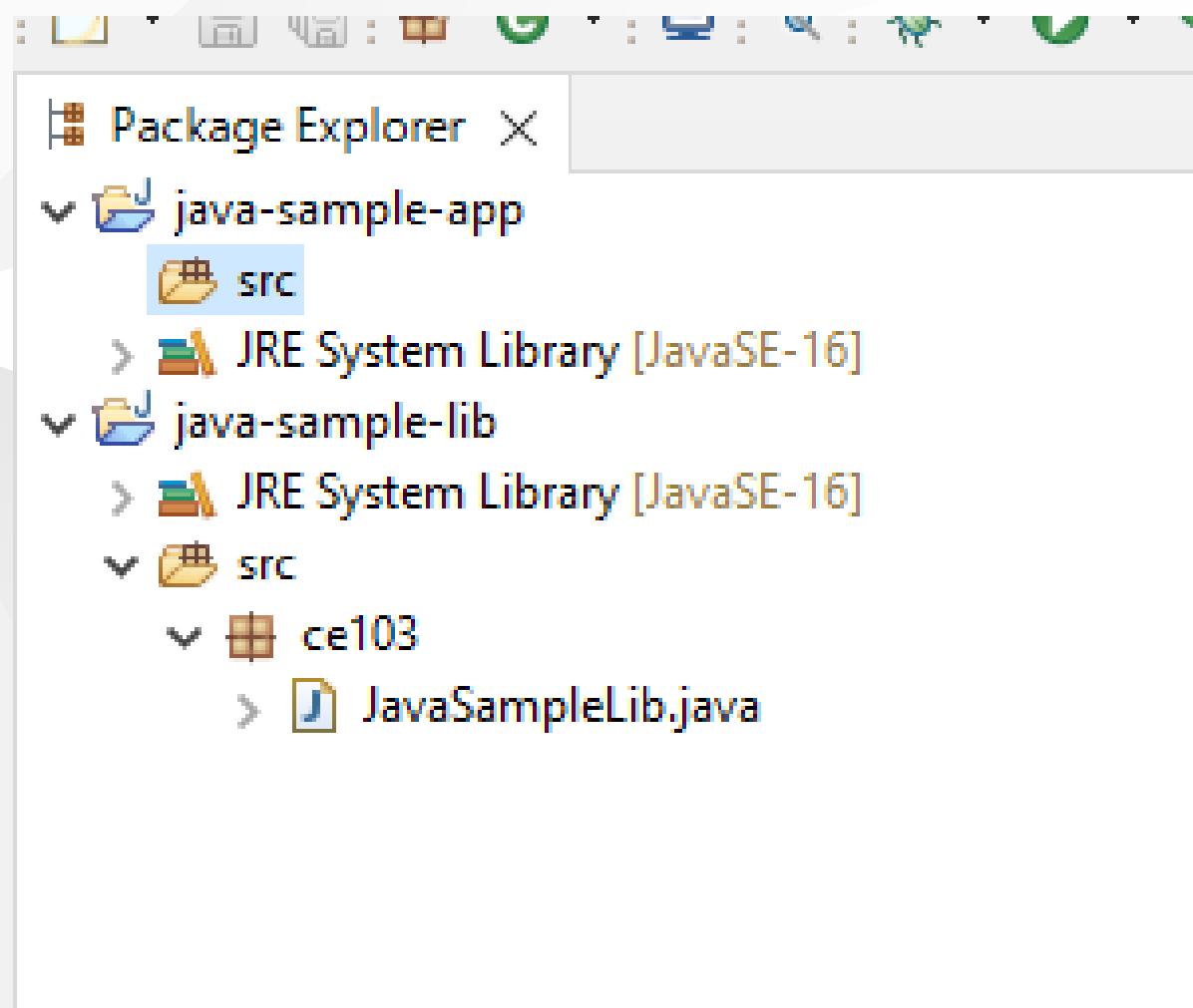
Or we can select by Add jar from current workspace



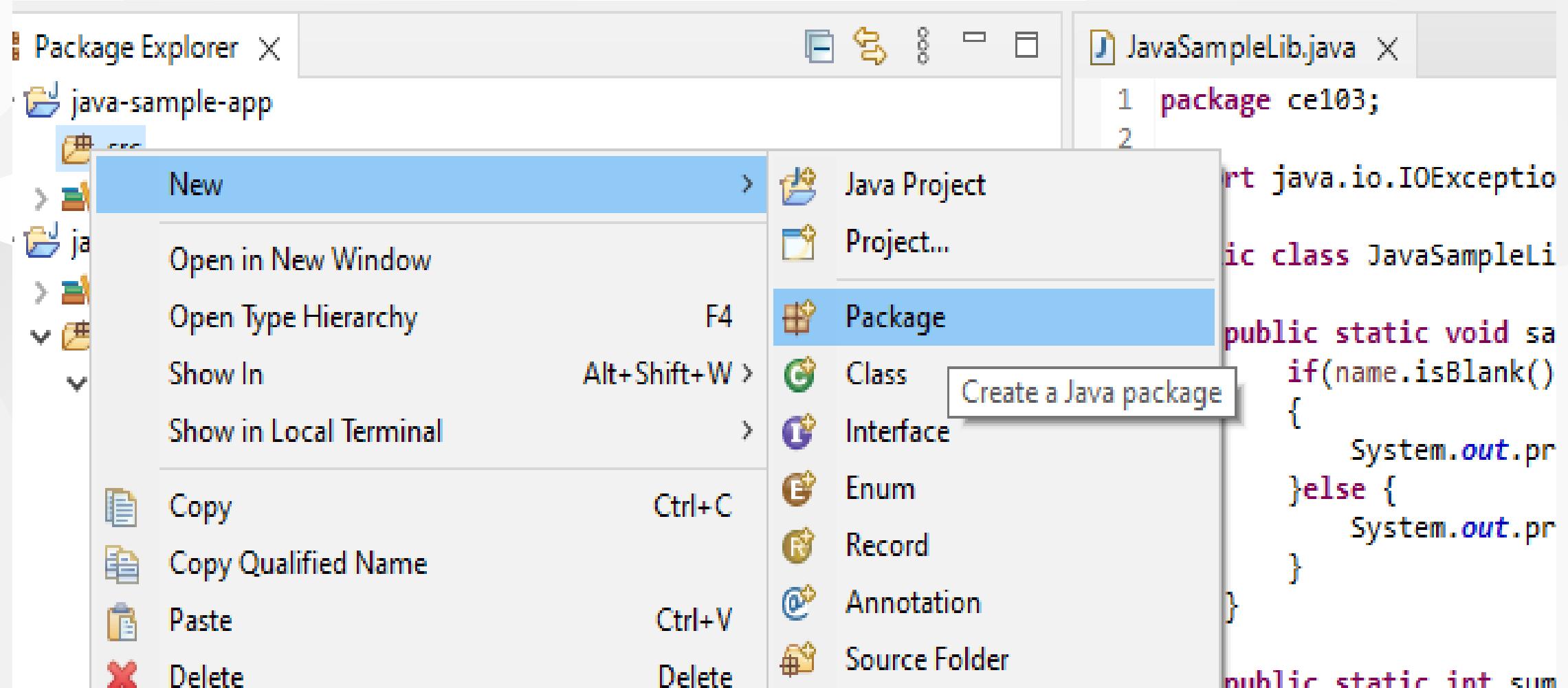
but in this step I won't add anything I'll add references later

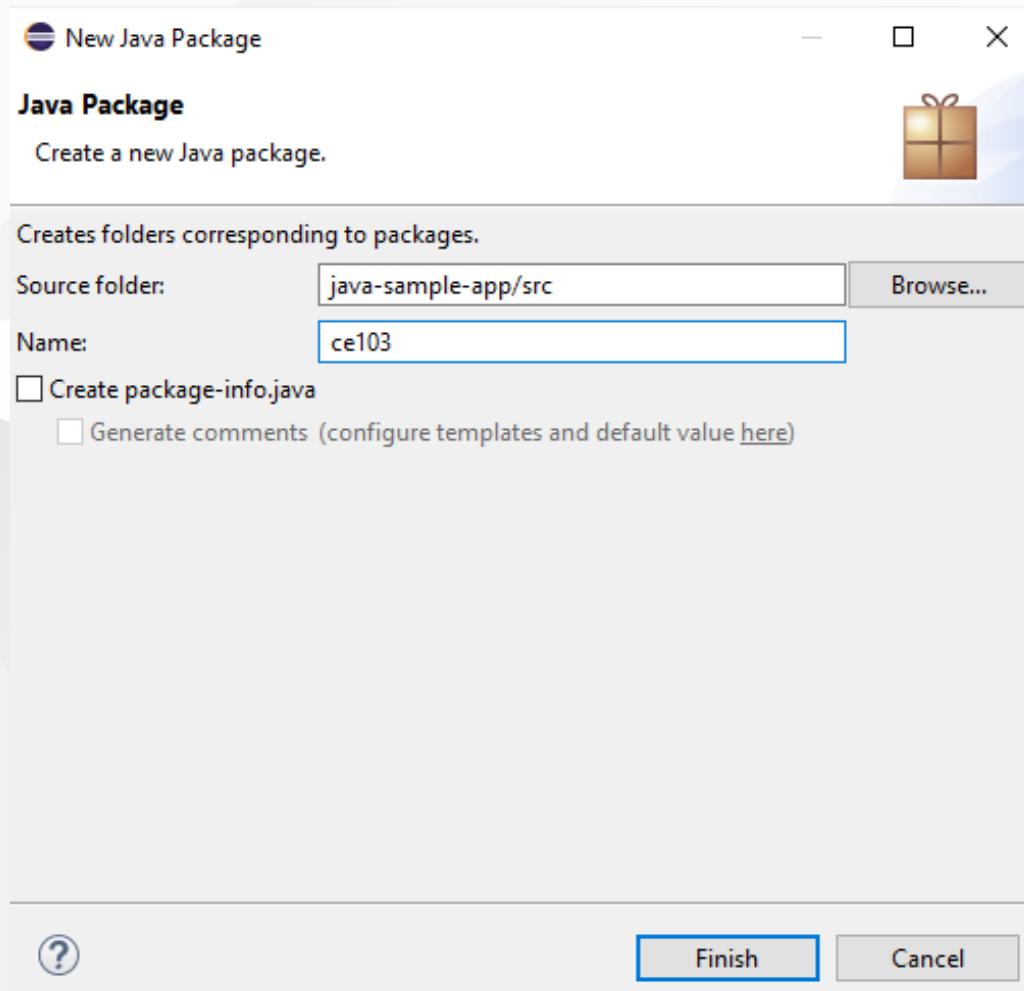


we will have the following project

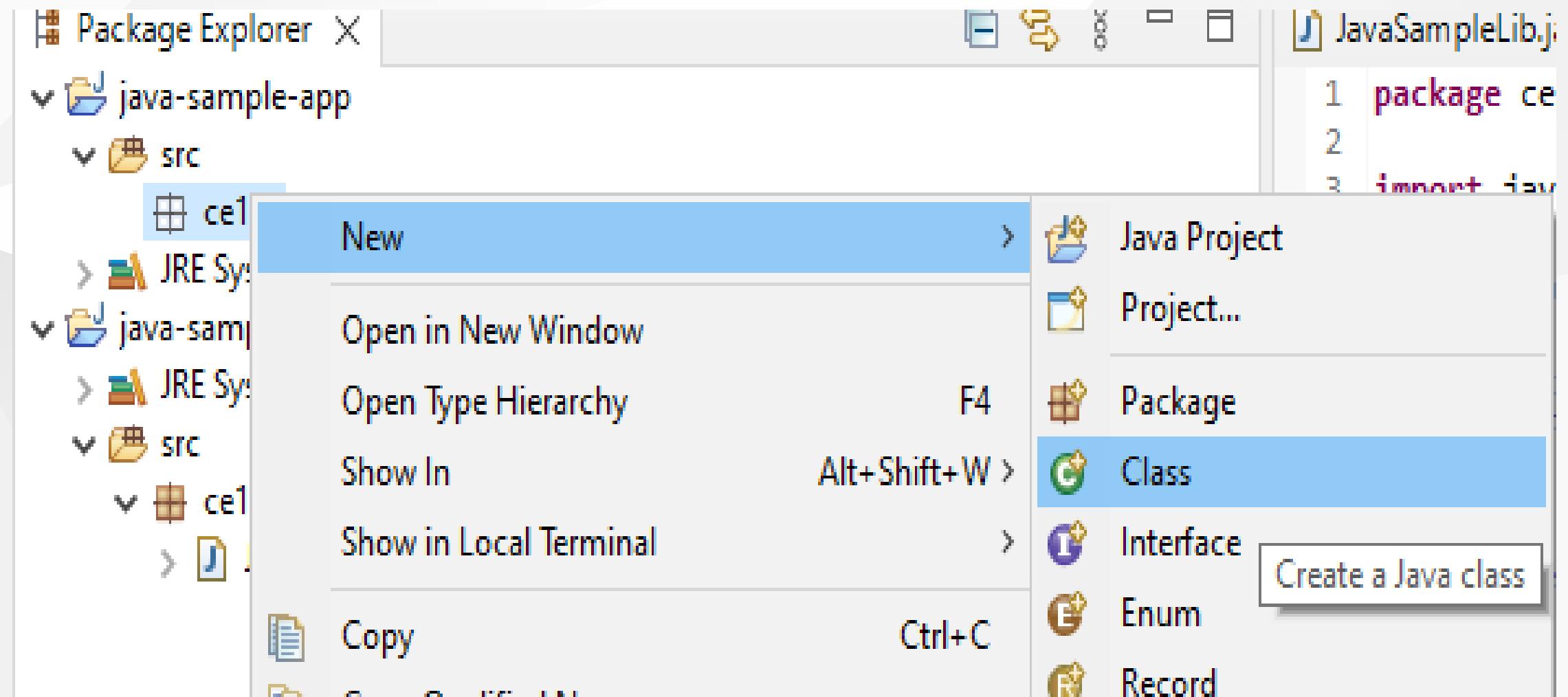


## lets create a package

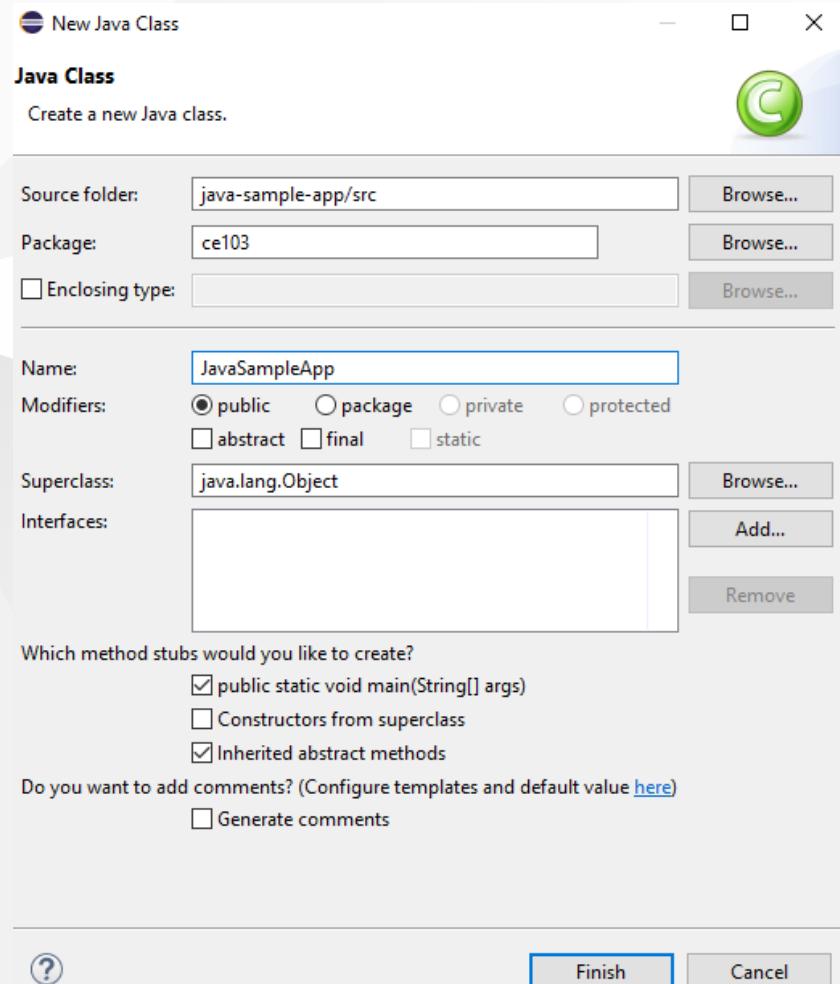




and lets create a main class for our application



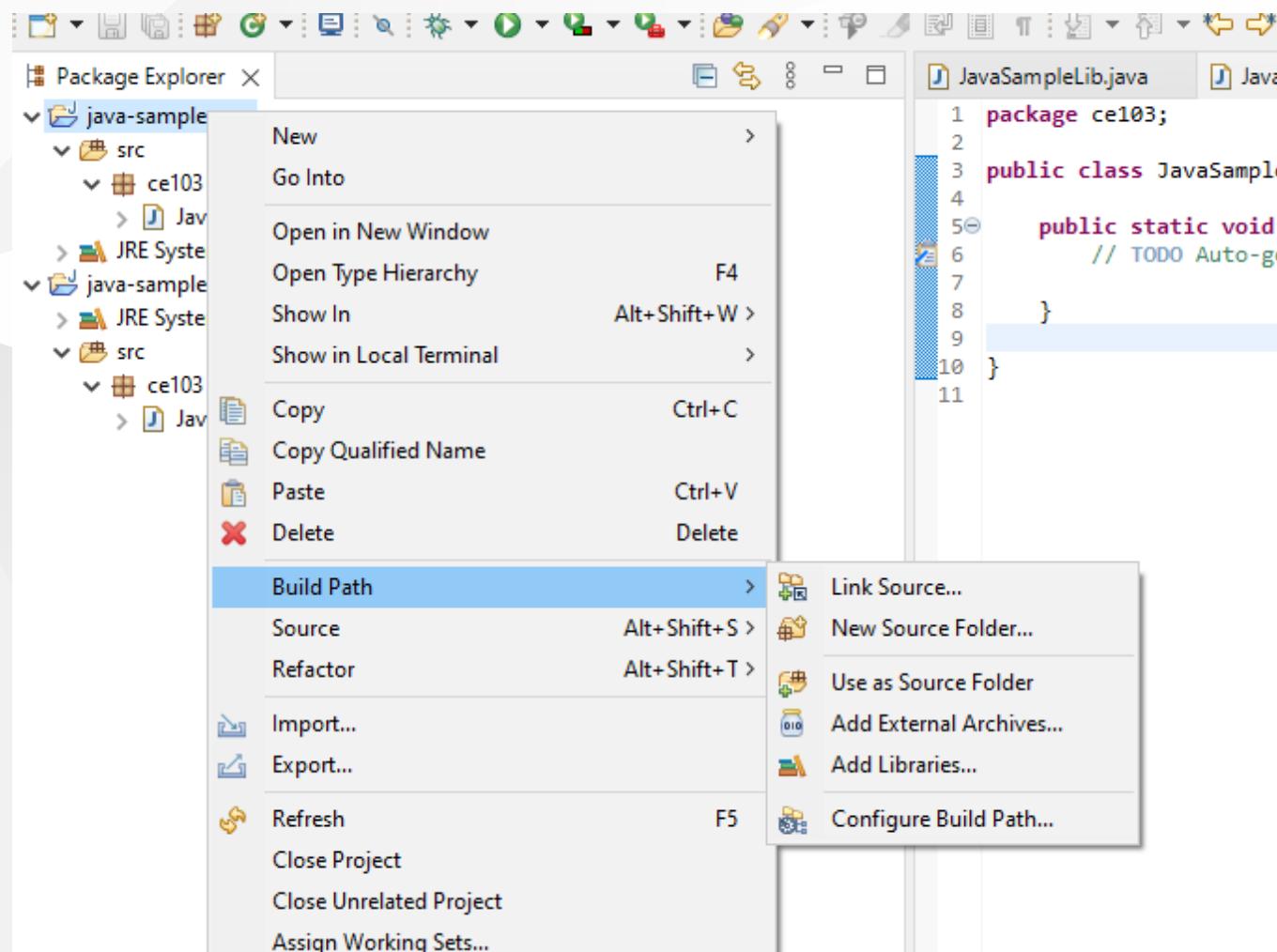
## check create main function



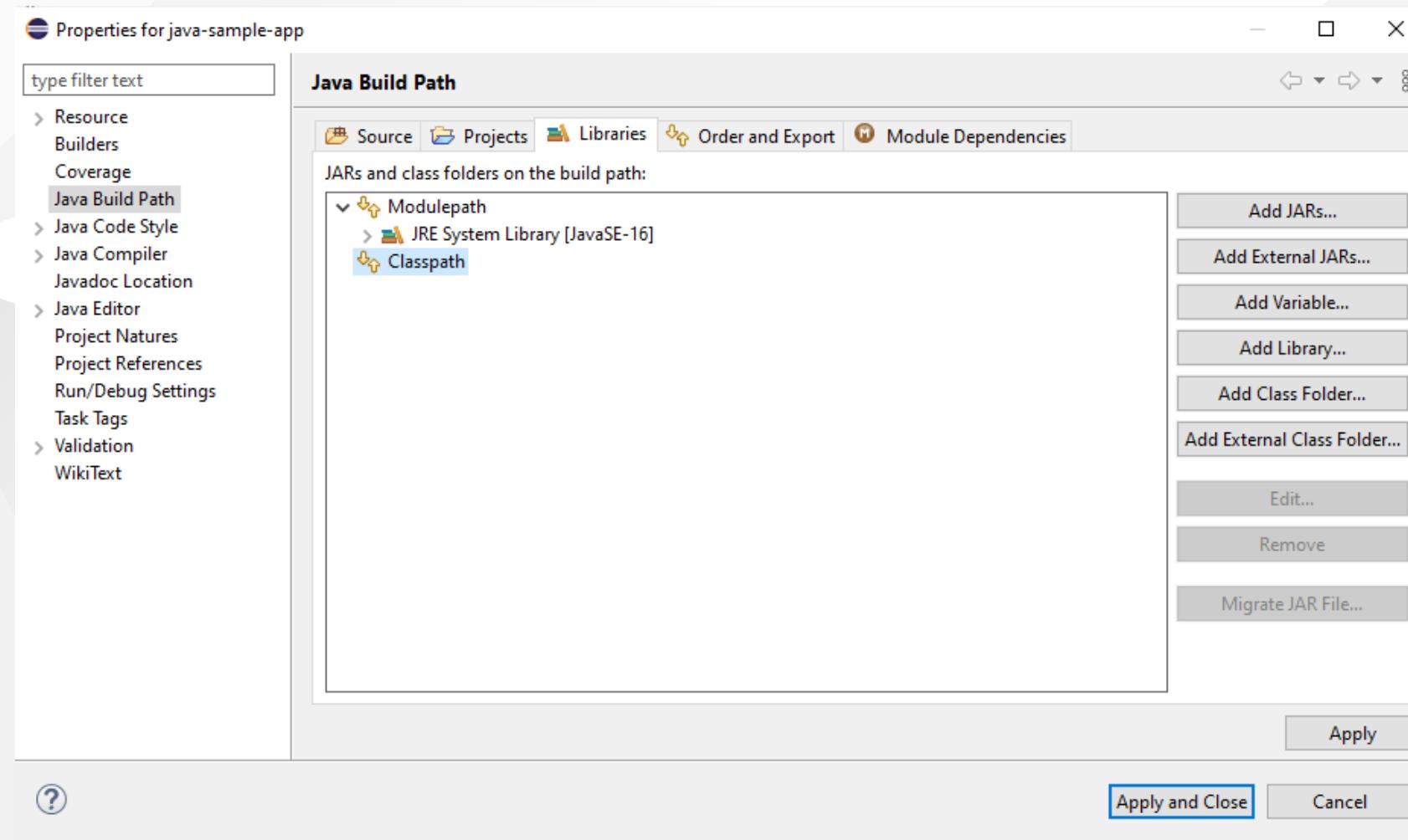
The screenshot shows the Eclipse IDE interface. On the left, the Package Explorer view displays two Java projects: 'java-sample-app' and 'java-sample-lib'. The 'java-sample-app' project contains a 'src' folder with a 'ce103' package containing a file named 'JavaSampleApp.java'. This file is currently open in the editor. The 'java-sample-lib' project also contains a 'src' folder with a 'ce103' package containing a file named 'JavaSampleLib.java'. Both projects have a dependency on the 'JRE System Library [JavaSE-16]'. The Java code in 'JavaSampleApp.java' is as follows:

```
1 package ce103;
2
3 public class JavaSampleApp {
4
5     public static void main(String[] args) {
6         // TODO Auto-generated method stub
7
8     }
9
10 }
11
```

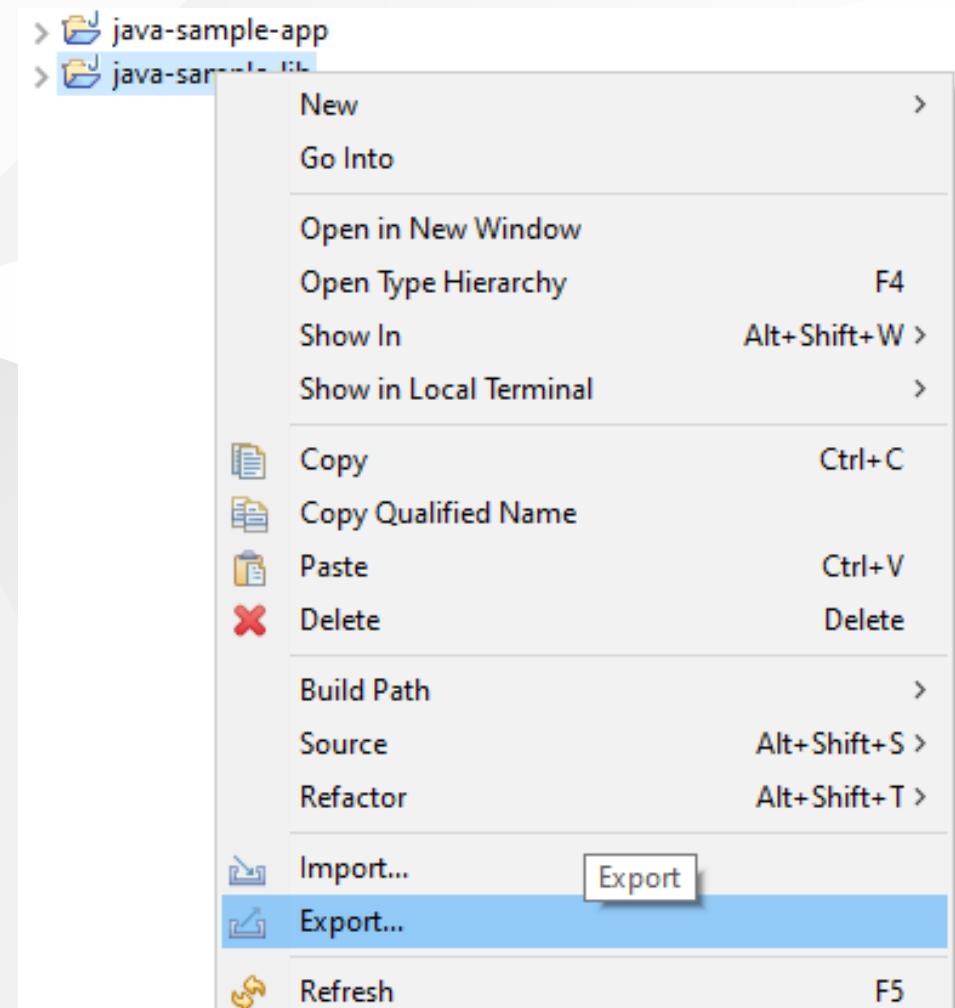
right click to project and add reference



you can enter same configurations from project properties



Lets export our library as a JAR file and then add to our classpath



## Select JAR file

Export resources into a JAR file on the local file system

Select an export wizard:

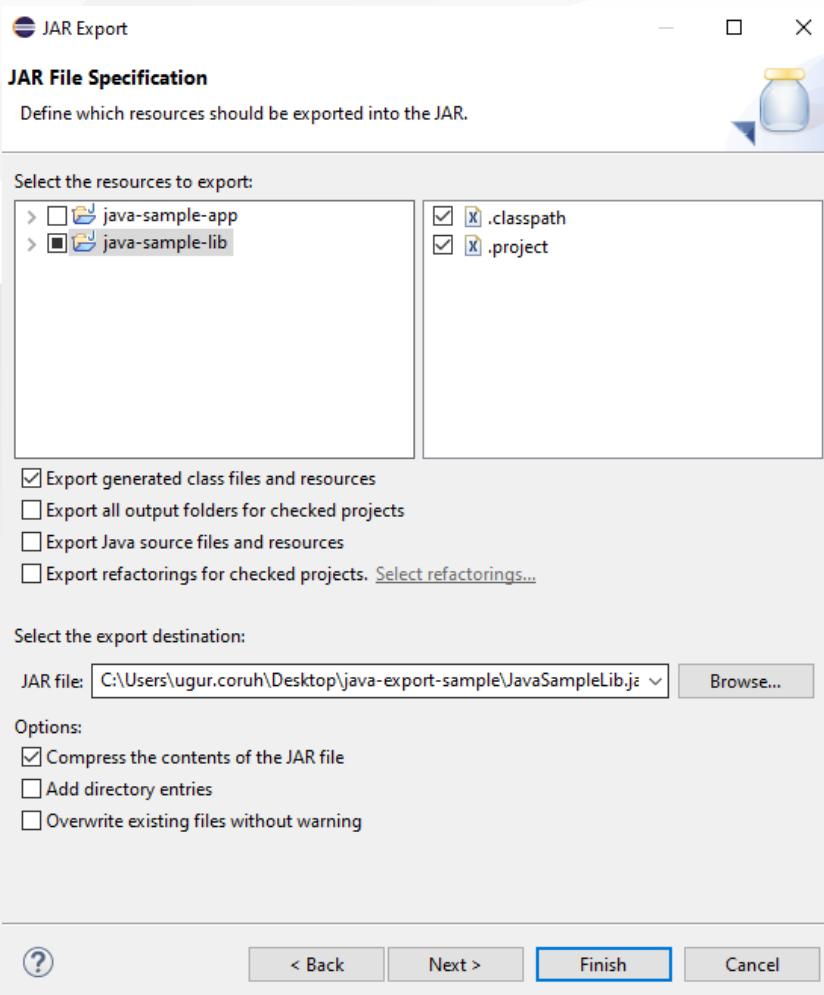
type filter text

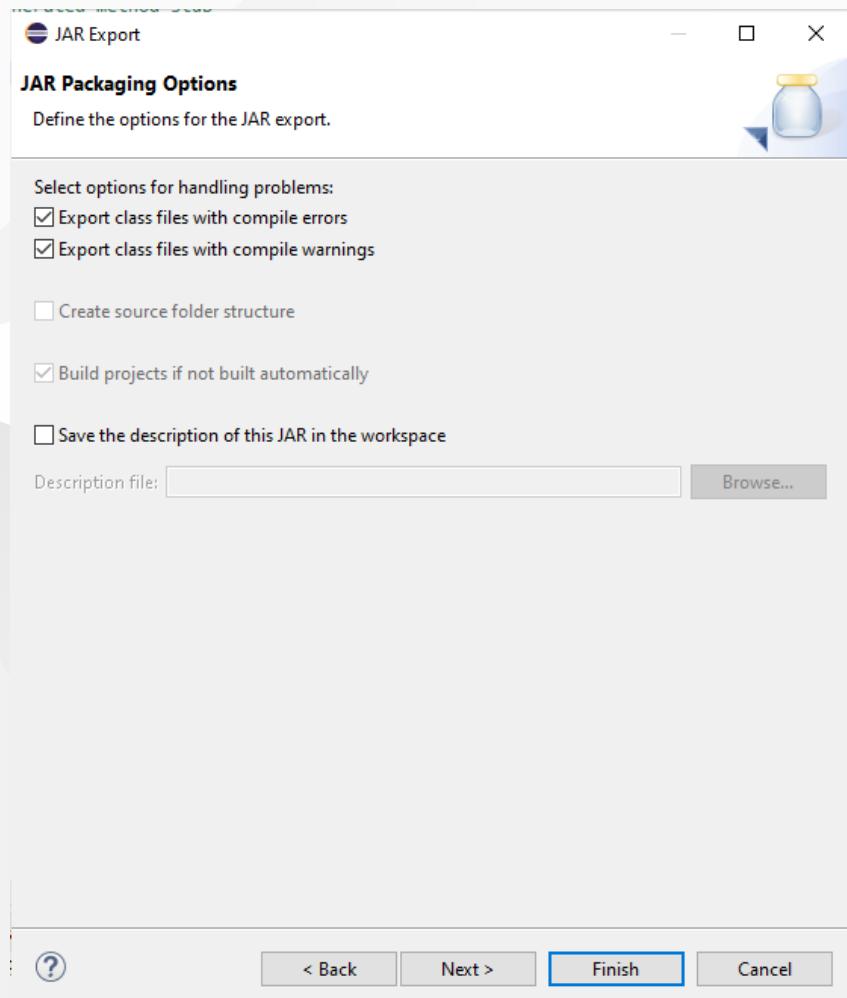
- ▼  Install
  -  Installed Software Items to File
- ▼  Java
  -  JAR file
  -  Javadoc
  -  Runnable JAR file
- ▼  Run/Debug
  -  Breakpoints
  -  Coverage Session
  -  Launch Configurations
- ▼  Team

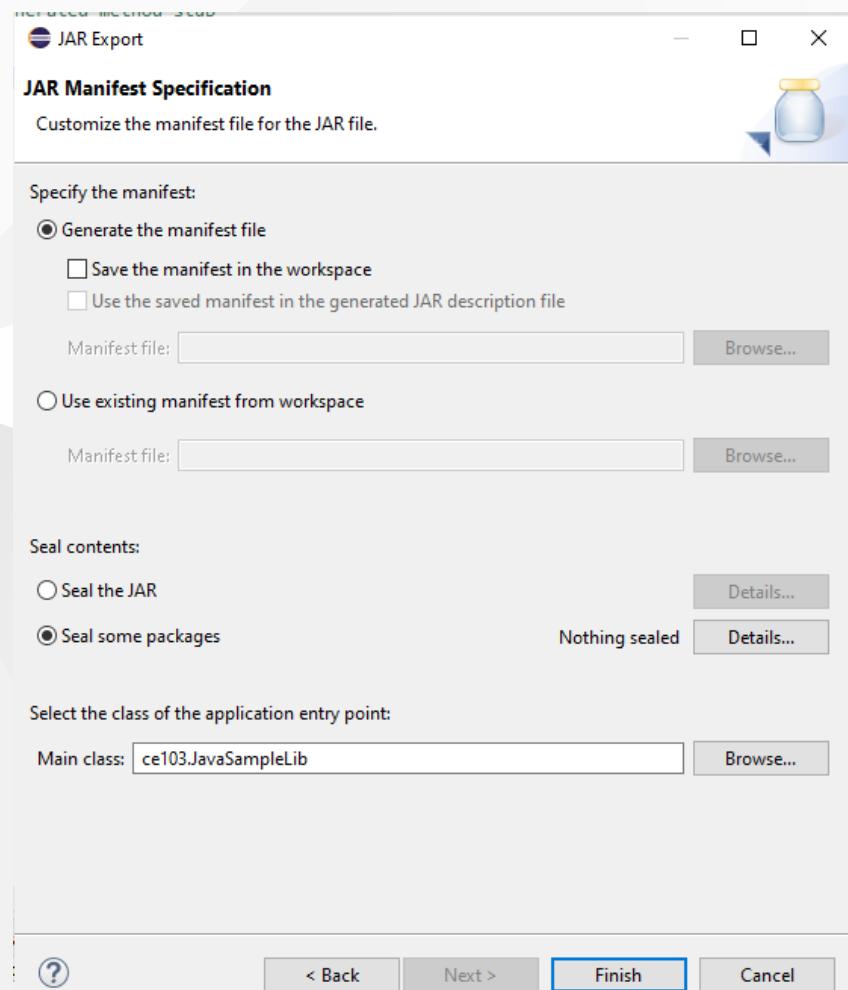


we configured output as

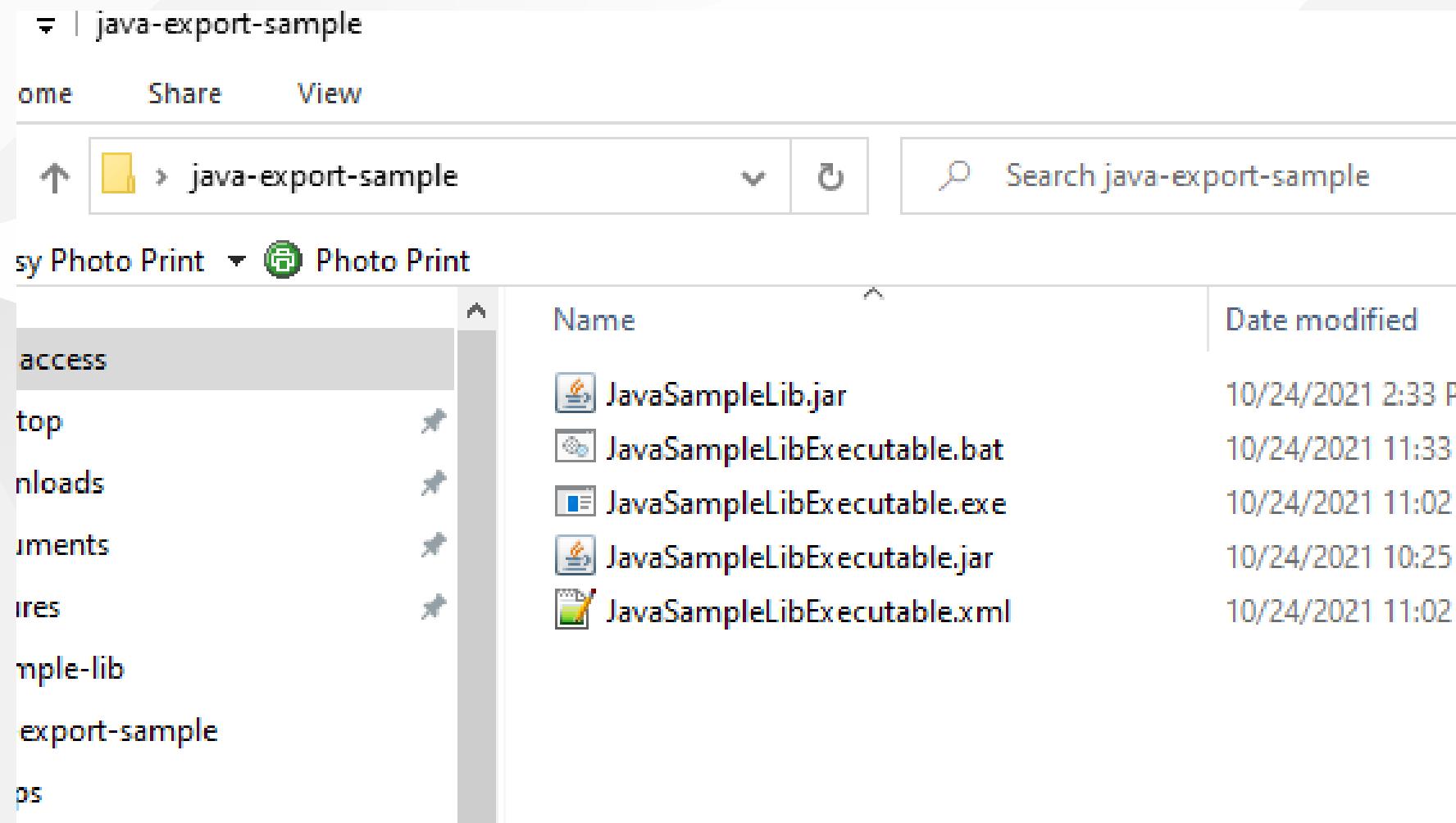
C:\Users\ugur.coruh\Desktop\java-export-sample\JavaSampleLib.jar





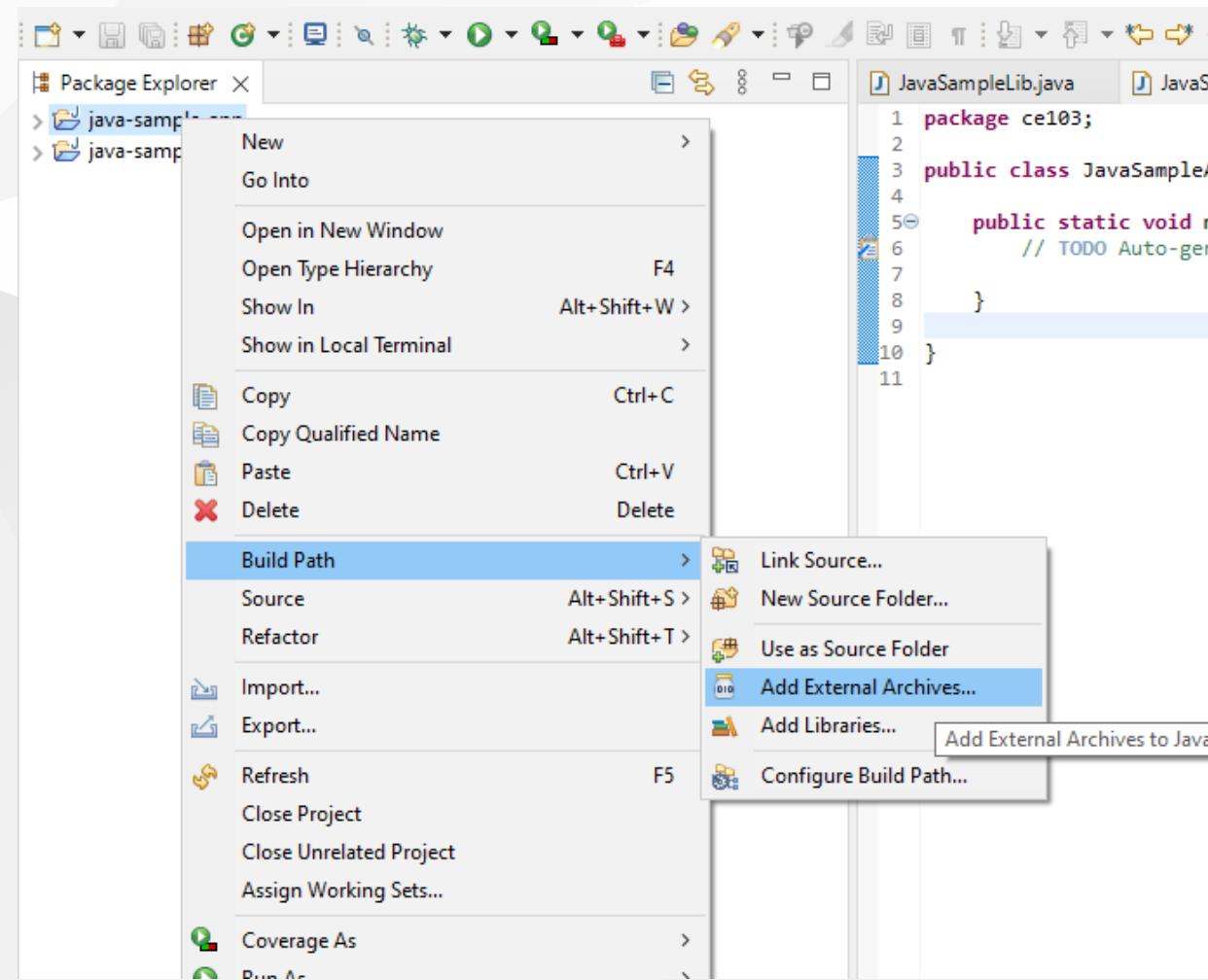


In the same export folder now we have JavaSampleLib.jar

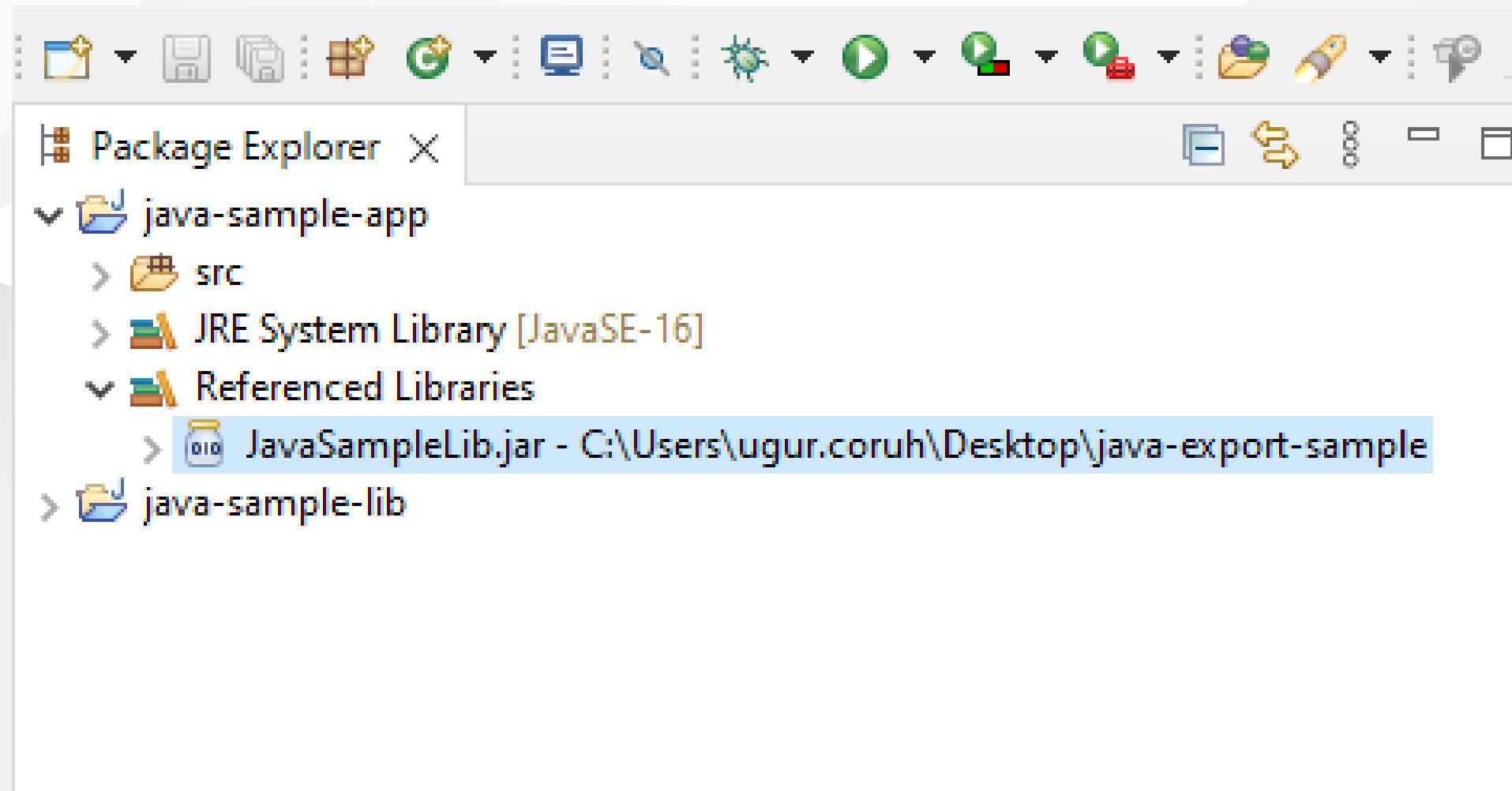


return back to java-sample-app and then add this jar file to our project

Build Path->Add External Archives



you will see its added to reference libraries



in our JavaSampleApp.java we can use the following source codes

```
package ce103;

import java.io.IOException;

public class JavaSampleApp {

    public static void main(String[] args) {
        // TODO Auto-generated method stub

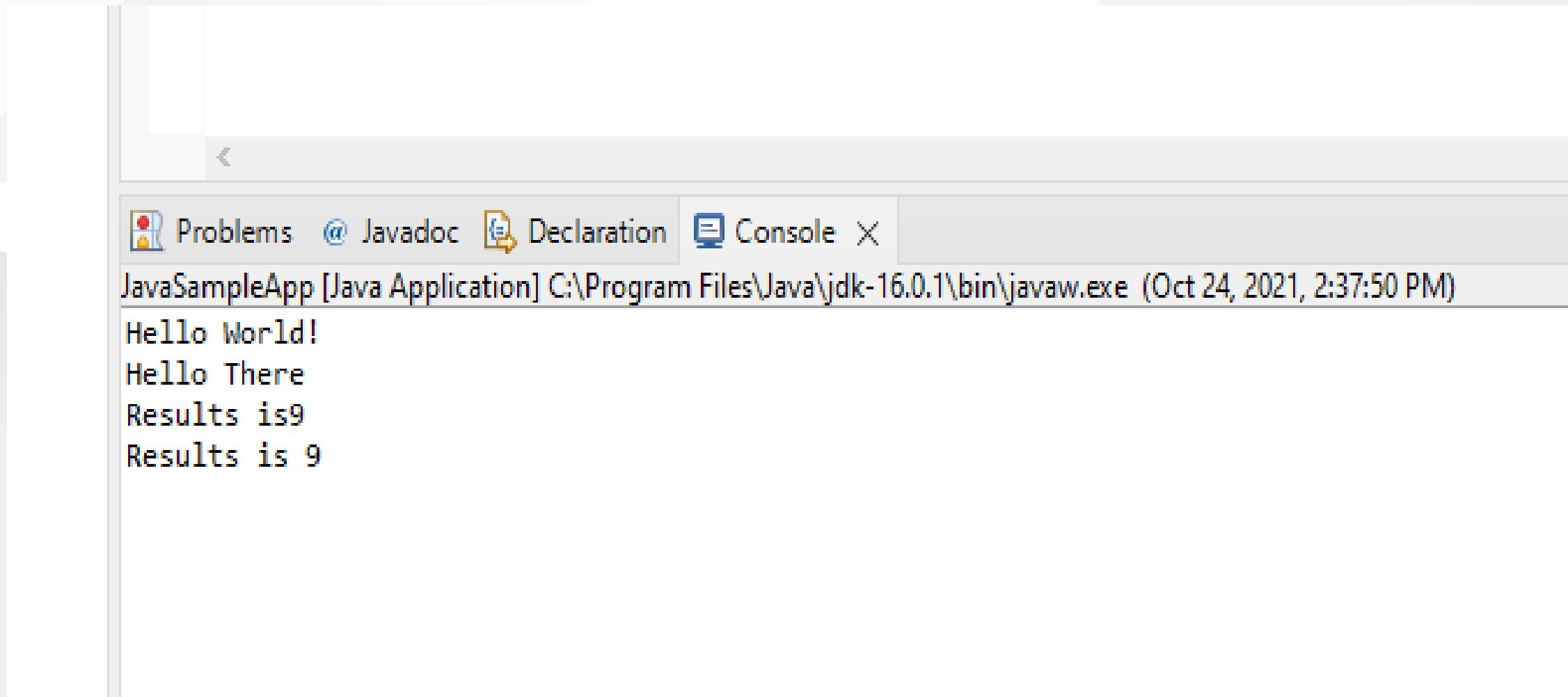
        System.out.println("Hello World!");

        JavaSampleLib.sayHelloTo("Computer");
        int result = JavaSampleLib.sum(5, 4);
        System.out.println("Results is" + result);
        System.out.printf("Results is %d \n", result);

        try {
            System.in.read();
        } catch (IOException e) {
            // TODO Auto-generated catch block
            e.printStackTrace();
        }
    }
}
```



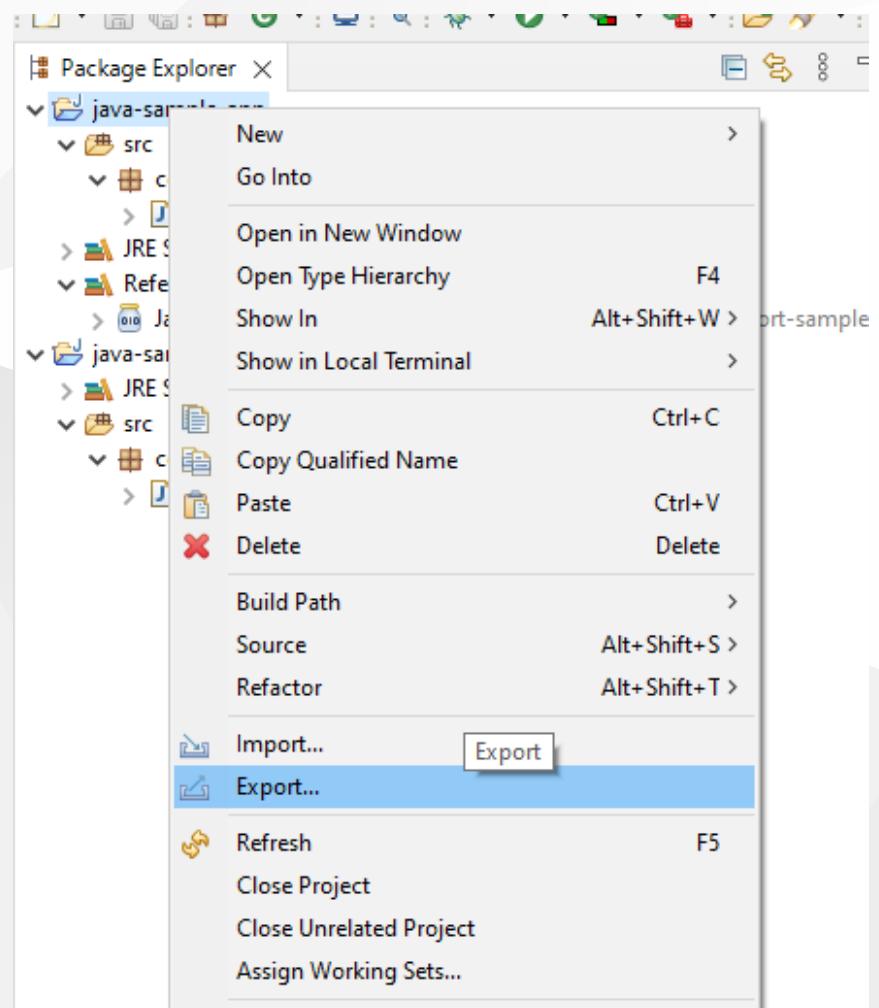
When we run application we will see similar output



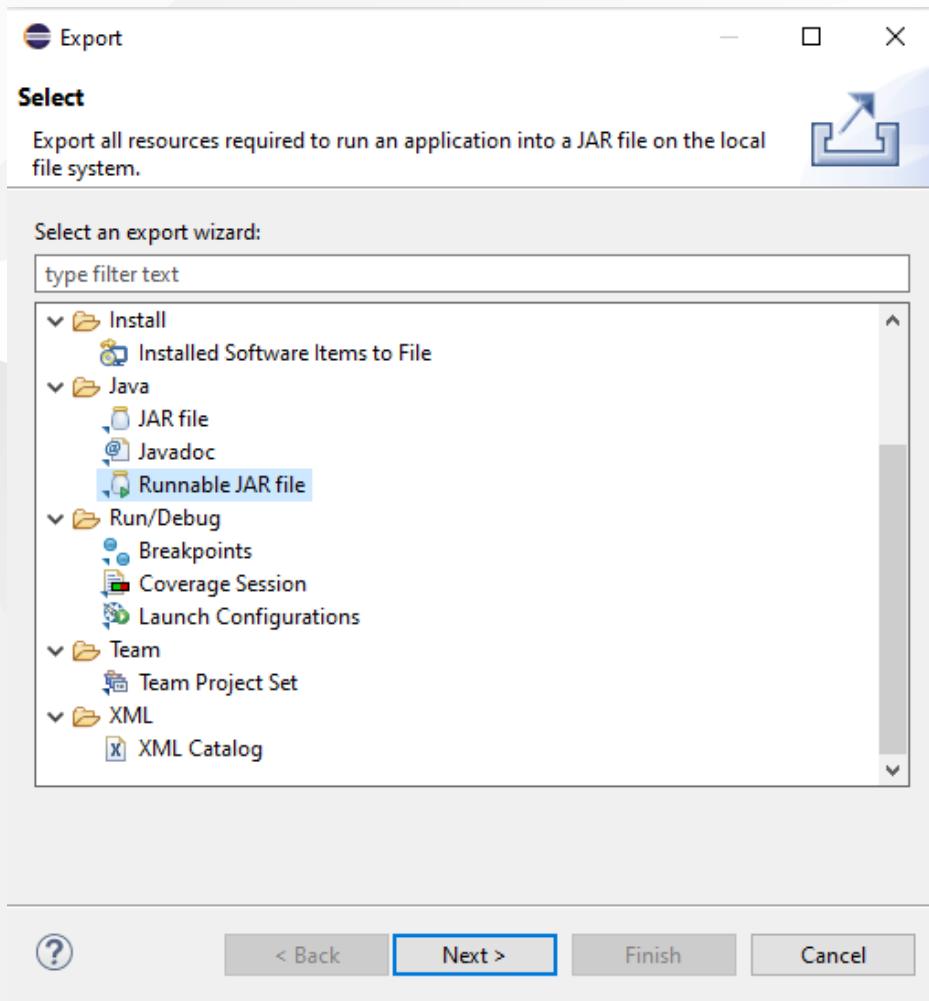
A screenshot of a Java IDE interface, specifically showing the 'Console' tab. The title bar indicates the project is 'JavaSampleApp [Java Application]' and the build path is 'C:\Program Files\Java\jdk-16.0.1\bin\javaw.exe'. The date and time of the log entry are '(Oct 24, 2021, 2:37:50 PM)'. The console output displays the following text:

```
Hello World!  
Hello There  
Results is9  
Results is 9
```

Lets export this application with its dependent library

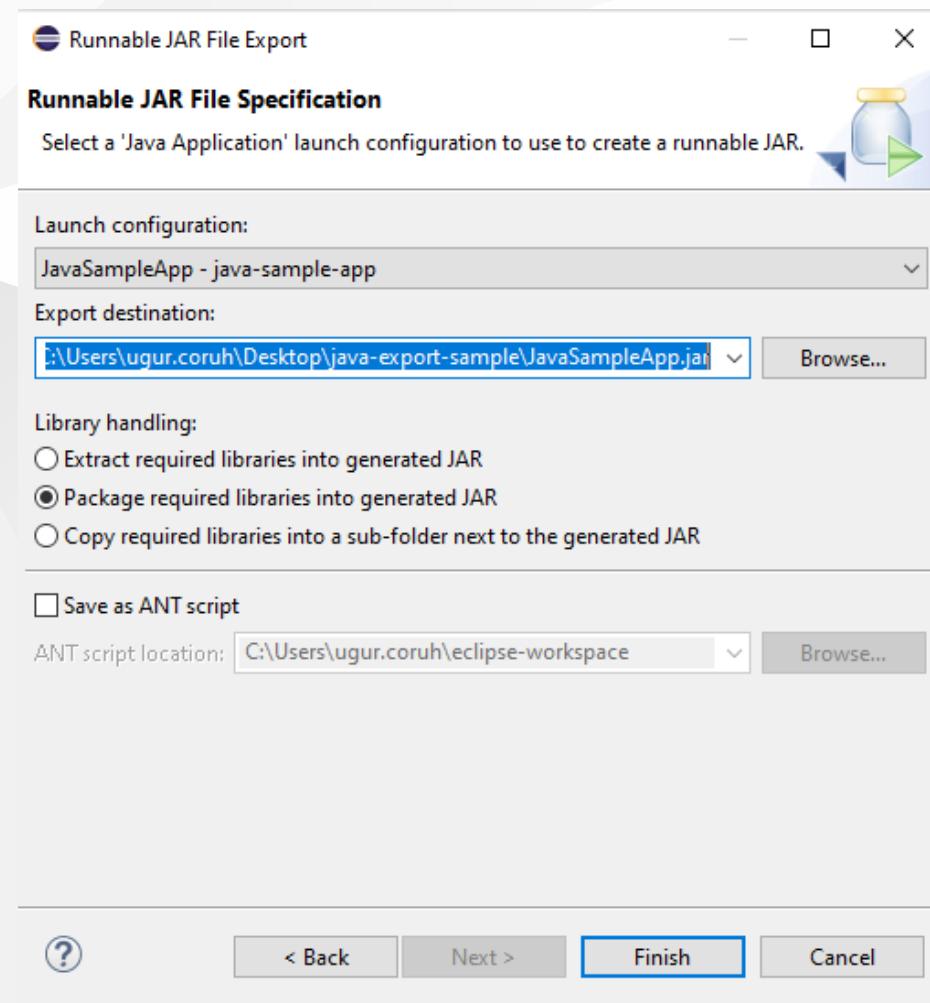


## Select runnable jar



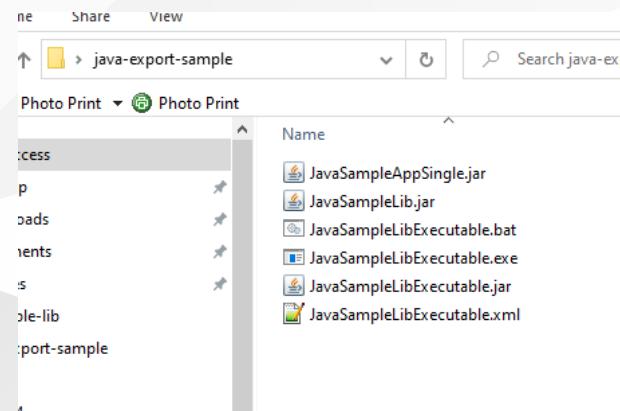
# Set Launch configuration and Export destination

C:\Users\ugur.coruh\Desktop\java-export-sample\JavaSampleAppSingle.jar



In this option we will have single jar file

In the export folder we do not see reference libraries



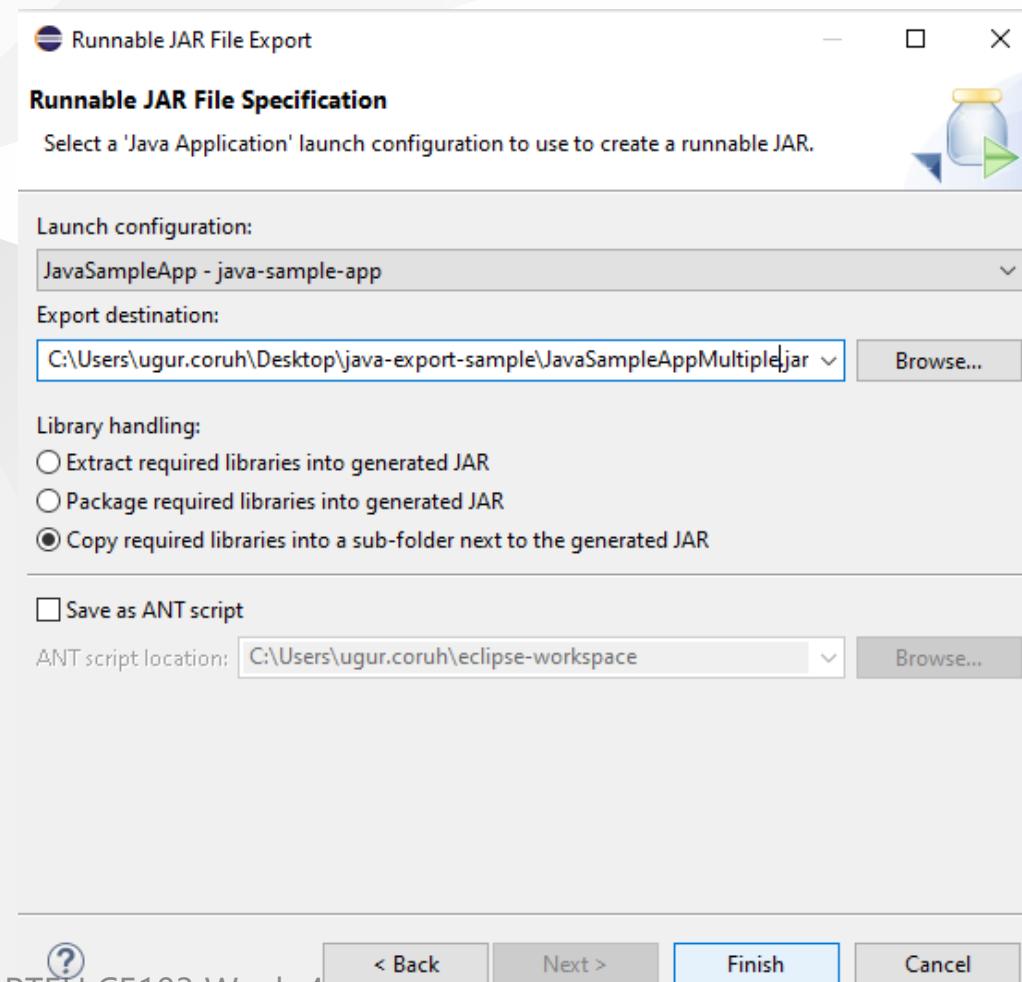
and we can run with command line



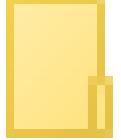
```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleAppSingle.jar
Hello World!
Hello There
Results is 9
Results is 9
```

only change copy required libraries setting and then give a new name for new jar file and export

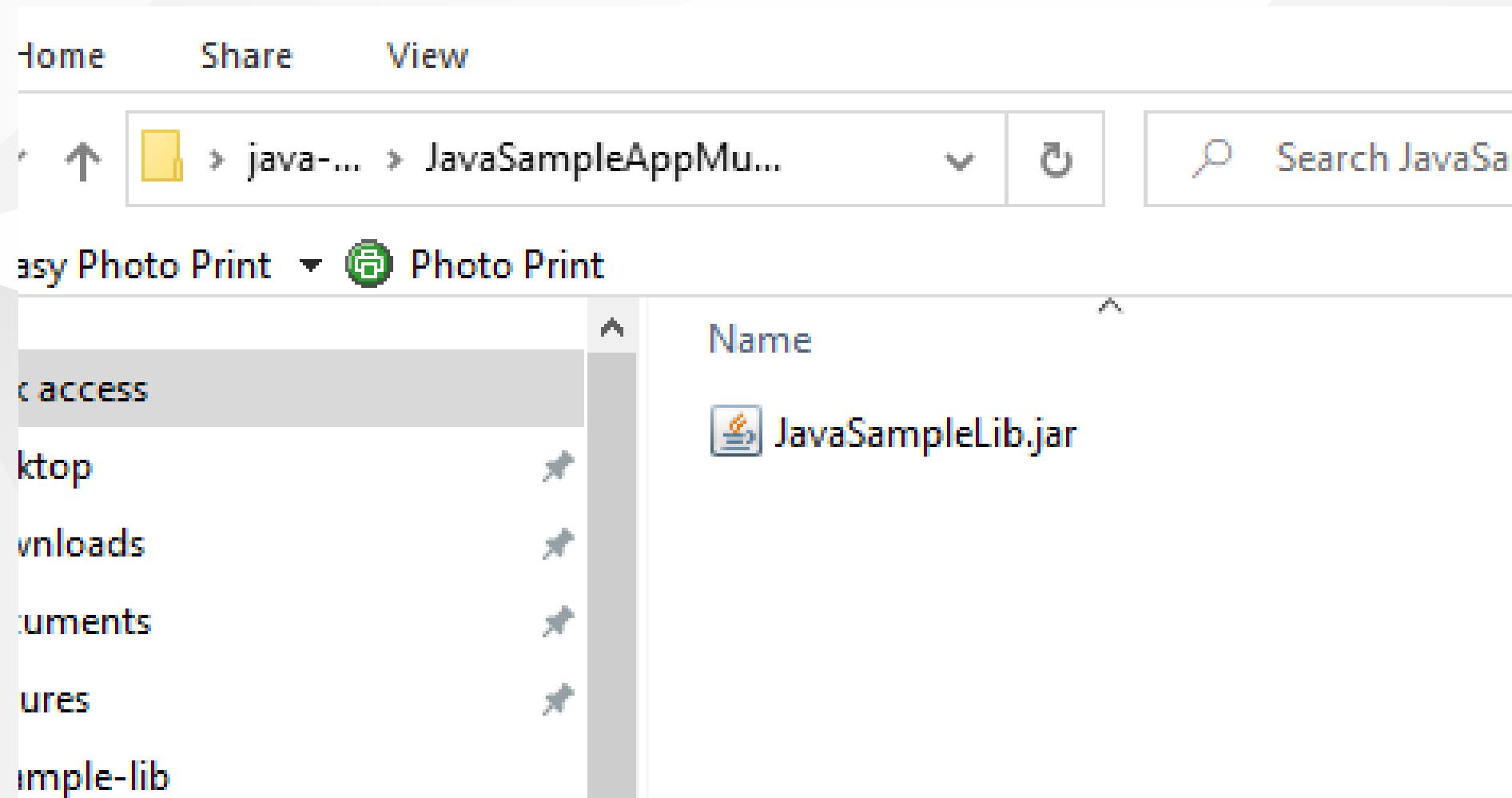
C:\Users\ugur.coruh\Desktop\java-export-sample\JavaSampleAppMultiple.jar



now we have a folder that contains our libraries referenced

Name
 JavaSampleAppMultiple_lib
 JavaSampleAppMultiple.jar
 JavaSampleAppSingle.jar
 JavaSampleLib.jar

in this file we can find our library



if we test our application we will see it will work

```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleAppMultiple.jar
Hello World!
Hello There
Results is9
Results is 9
```



if we delete JavaSampleLib.jar and then try running application we will get error

```
C:\Users\ugur.coruh\Desktop\java-export-sample>java -jar JavaSampleAppMultiple.jar
Hello World!
Exception in thread "main" java.lang.NoClassDefFoundError: ce103/JavaSampleLib
        at ce103.JavaSampleApp.main(JavaSampleApp.java:12)
Caused by: java.lang.ClassNotFoundException: ce103.JavaSampleLib
        at java.base/jdk.internal.loader.BuiltinClassLoader.loadClass(BuiltinClassLoader.java:636)
        at java.base/jdk.internal.loader.ClassLoaders$AppClassLoader.loadClass(ClassLoaders.java:182)
        at java.base/java.lang.ClassLoader.loadClass(ClassLoader.java:519)
        ... 1 more
C:\Users\ugur.coruh\Desktop\java-export-sample>
```

# Program Testing

# Unit Test Development

## C Unit Tests

## Visual Studio Community Edition

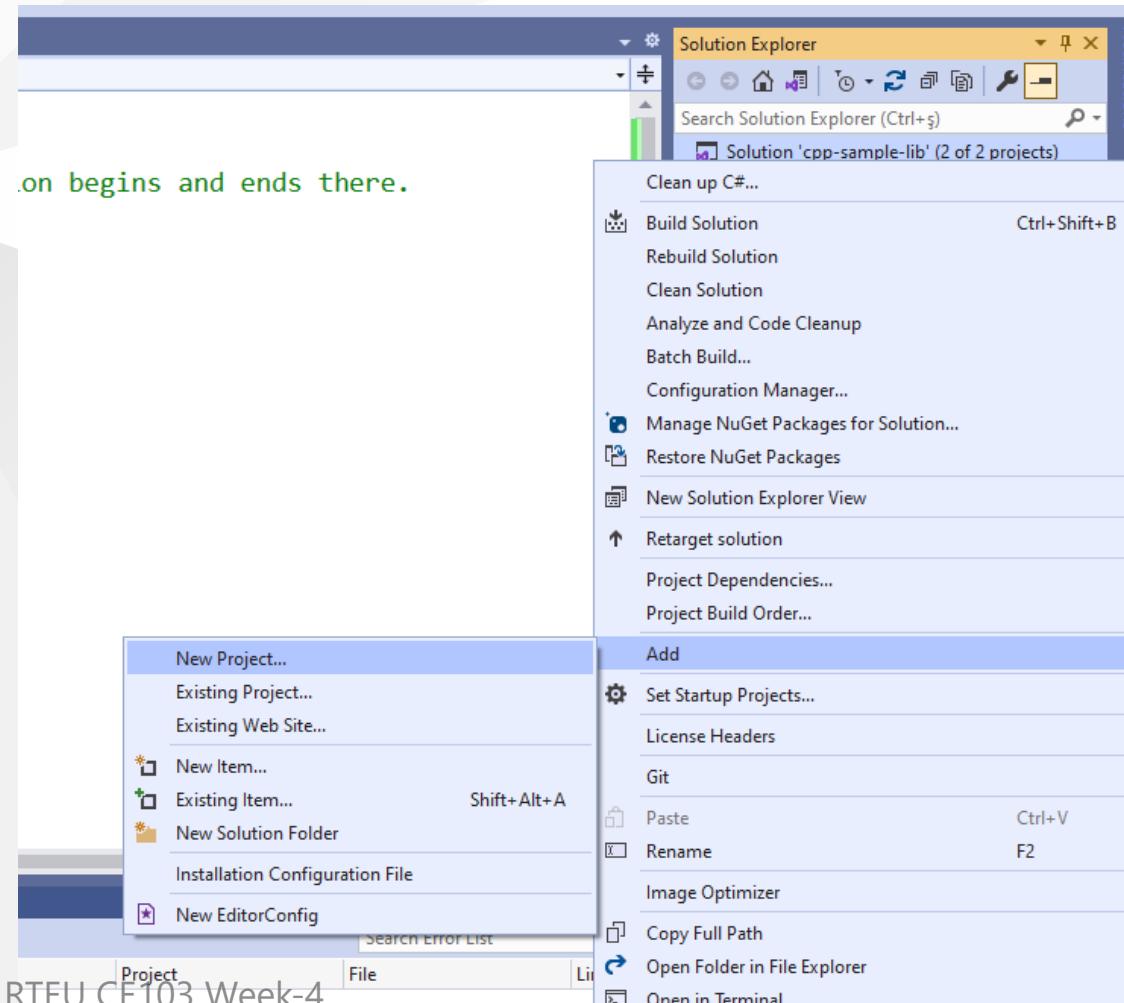


## C++ Unit Tests

# Visual Studio Community Edition

## C/C++ için birim testleri yazma - Visual Studio (Windows) | Microsoft Docs

Use cpp-sample-lib project and add



## select Native Unit Test

The screenshot shows the Microsoft Visual Studio Marketplace search interface. At the top, there is a search bar with the placeholder "Search for templates (Alt+S)" and a magnifying glass icon. To the right of the search bar is a "Clear all" link. Below the search bar are three dropdown menus: "C++" (selected), "All platforms" (selected), and a dropdown menu currently set to "Test". A blue border highlights the "Test" dropdown. Below these filters, two template cards are displayed:

- Native Unit Test Project**: Write C++ unit tests using the native Microsoft CppUnitTest framework. This card includes a C++ icon, a Windows icon, and a Test icon.
- Google Test**: Write C++ unit tests using Google Test. Includes a copy of the Google Test library for use. This card also includes a C++ icon, a Windows icon, and a Test icon.

At the bottom right of the search results area, there is a callout with the text "Not finding what you're looking for? [Install more tools and features](#)".

set project path and name

# Configure your new project

Native Unit Test Project

C++

Windows

Test

Project name

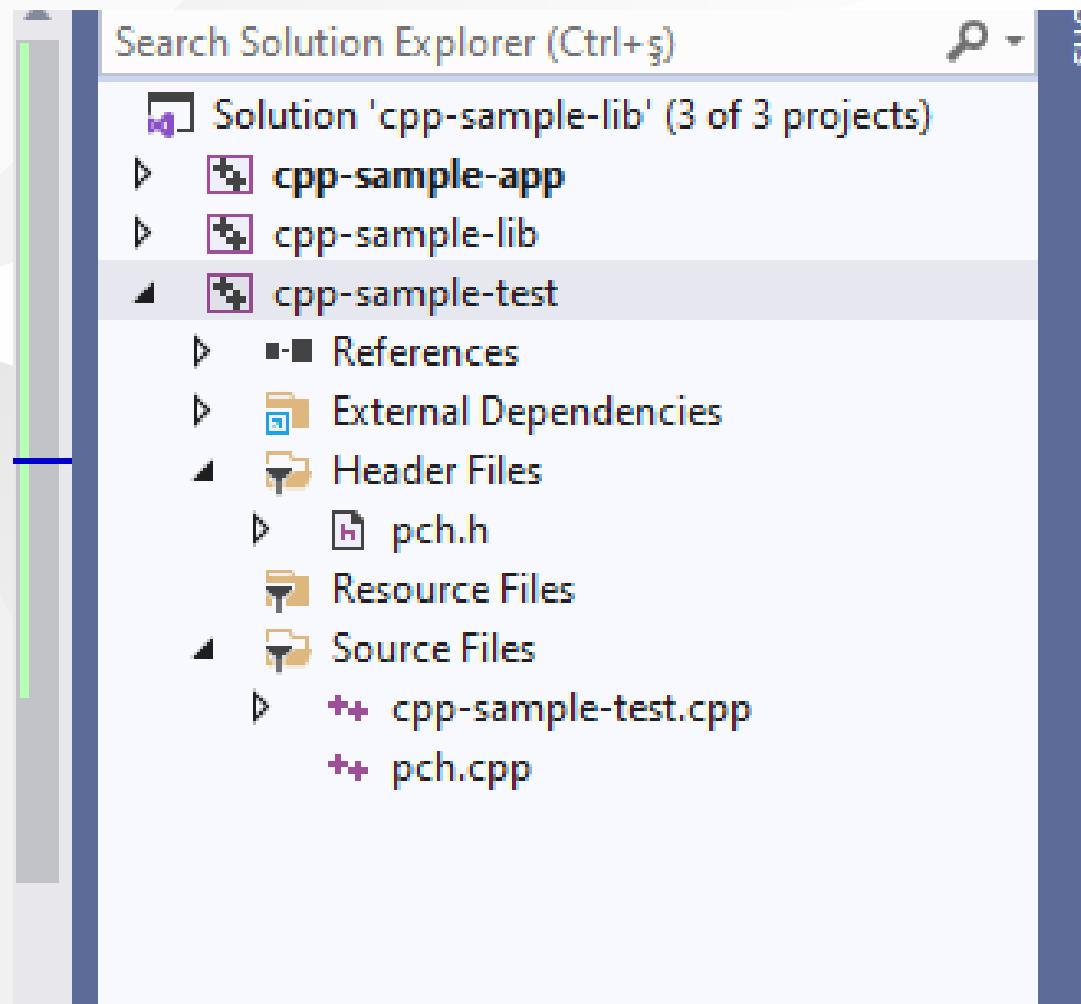
cpp-sample-test

Location

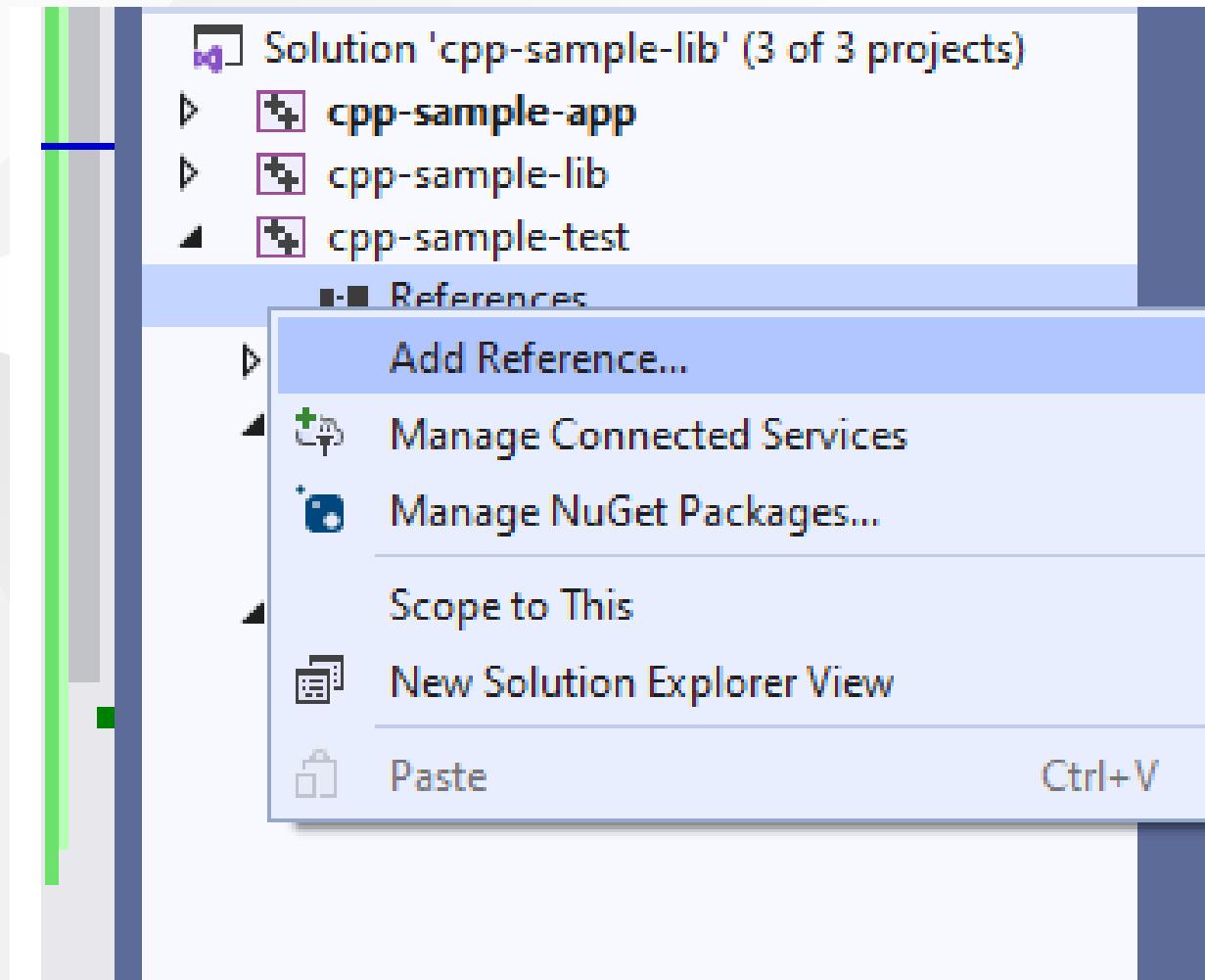
E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce111

...

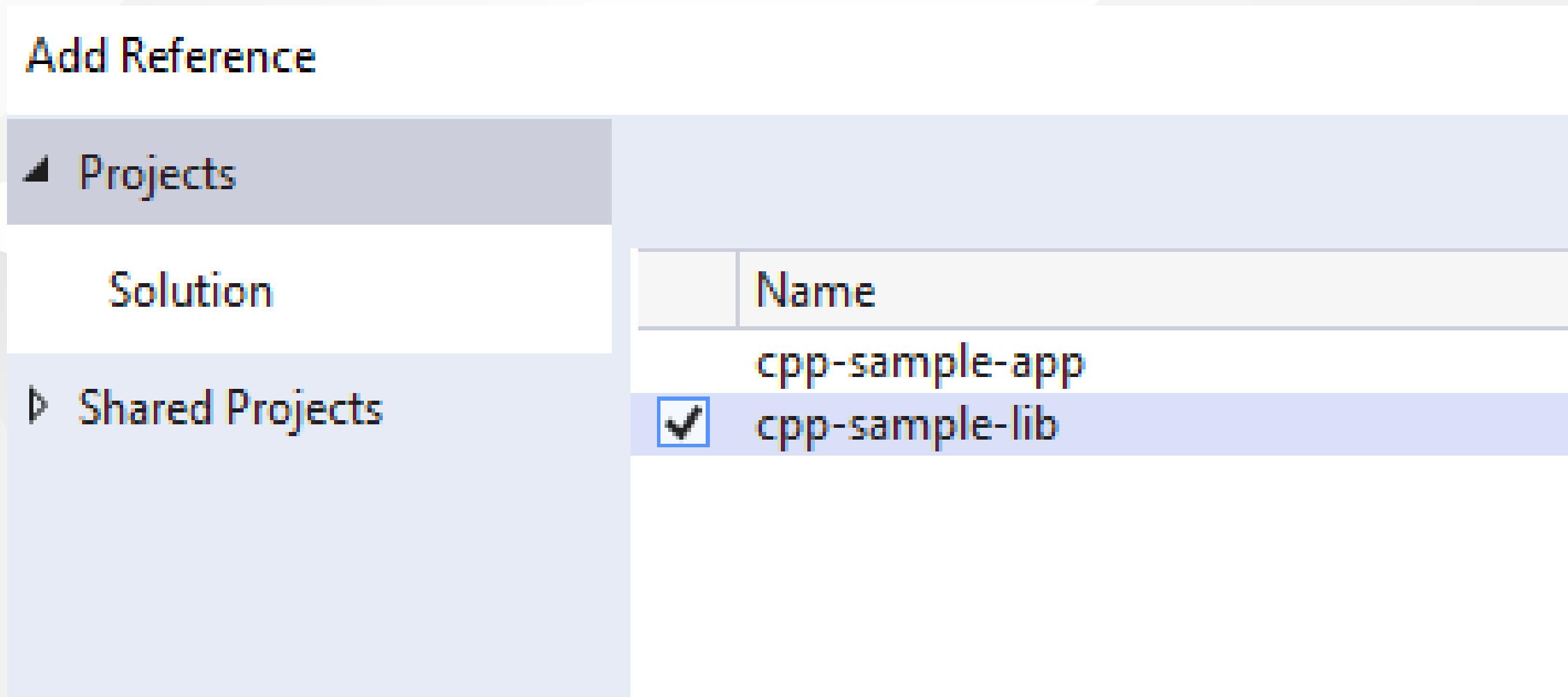
you will have cpp-sample-test project



## add library project from references



## Add cpp-sample-lib to cpp-sample-test project



## cpp-sample-test.cpp

```
#include "pch.h"
#include "CppUnitTest.h"
#include "..\cpp-sample-lib\samplelib.h"

using namespace Microsoft::VisualStudio::CppUnitTestFramework;

namespace cppsampletest
{
    TEST_CLASS(cppsampletest)
    {
        public:

            TEST_METHOD(TestSumCorrect)
            {
                Assert::AreEqual(9, sum(4, 5));
            }

            TEST_METHOD(TestSumInCorrect)
            {
                Assert::AreEqual(10, sum(4, 5));
            }
    };
}
```



The screenshot shows a Visual Studio interface. The top part is the code editor with the file `cpp-sample-test.cpp` open. The code defines a test class `cppsampletest` with two test methods: `TestSumCorrect` (passing) and `TestSumInCorrect` (failing). The bottom part is the Test Explorer window, which lists all tests and their results. It shows 2 passed tests and 2 failed tests under the category `cppsampletest`. The failed test `TestSumInCorrect` has an error message: "Assert failed. Expected:<10> Actual:<9>".

```
cpp-sample-test.cpp  + x cpp-sample-app.cpp    samplelib.h    pch.h    pch.cpp    cpp-sample-lib.cpp
cpp-sample-test
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24

namespace cppsampletest
{
    TEST_CLASS(cppsampletest)
    {
        public:

            TEST_METHOD(TestSumCorrect)
            {
                Assert::AreEqual(9, sum(4, 5));
            }

            TEST_METHOD(TestSumInCorrect)
            {
                Assert::AreEqual(10, sum(4, 5));
            }
    };
}

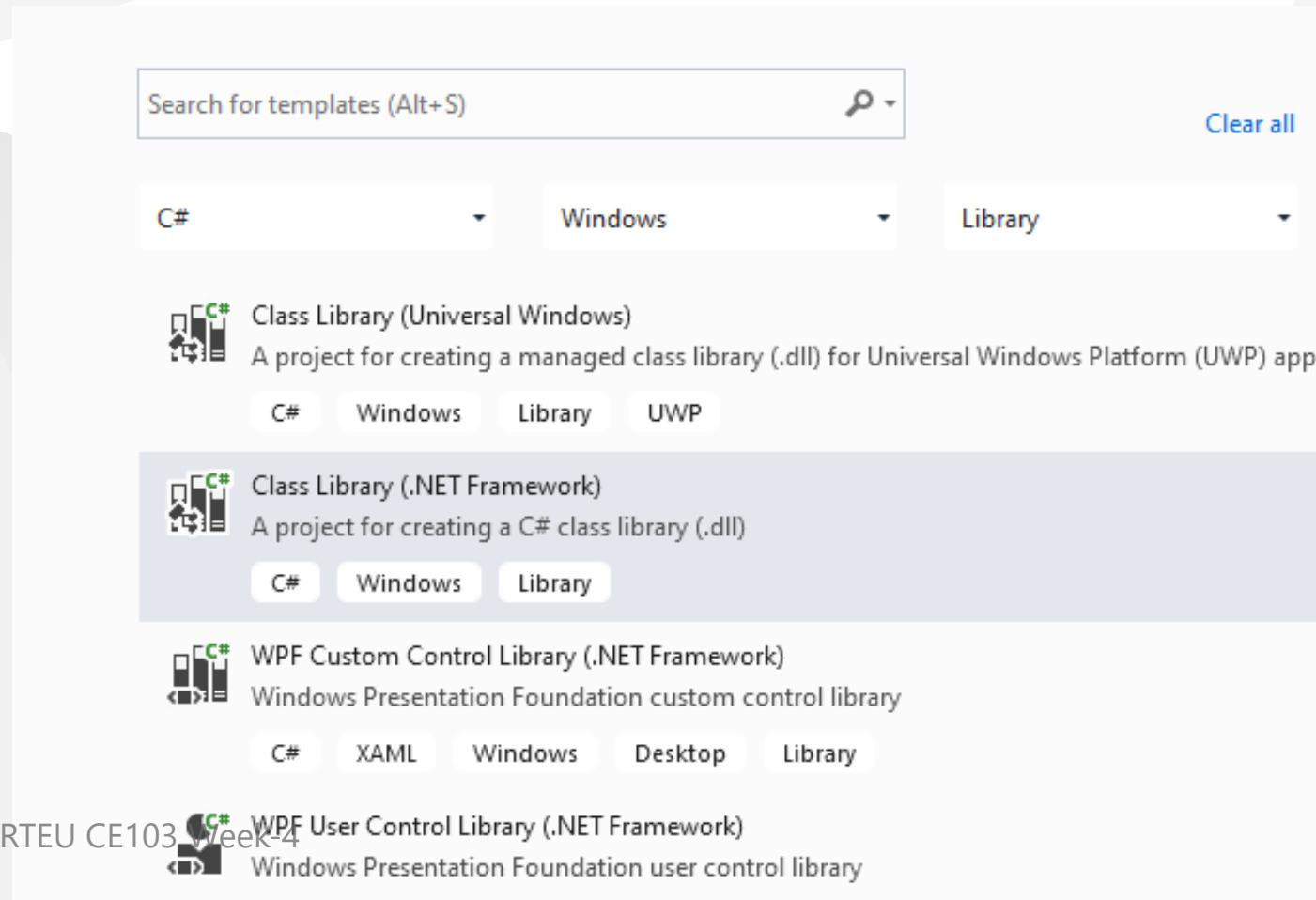
144 %  No issues found
Test Explorer
2  1  1
Test Duration Traits Error Message
cpp-sample-test (2) 253 ms
cppsampletest (2) 253 ms
    cppsampletest (2) 253 ms
        TestSumCorrect  < 1 ms
        TestSumInCorrect  253 ms
                        Assert failed. Expected:<10> Actual:<9>
```

## C# Unit Tests

# Install extension fine code coverage

[https://marketplace.visualstudio.com/items?  
itemName=FortuneNgwenya.FineCodeCoverage](https://marketplace.visualstudio.com/items?itemName=FortuneNgwenya.FineCodeCoverage)

Create a .Net Framework Library



## set project framework and path

### Configure your new project

Class Library (.NET Framework)    C#    Windows    Library

Project name

cs-lib-sample

Location

C:\Users\ugur.coruh\Desktop\cs-lib-sample\



Solution name i

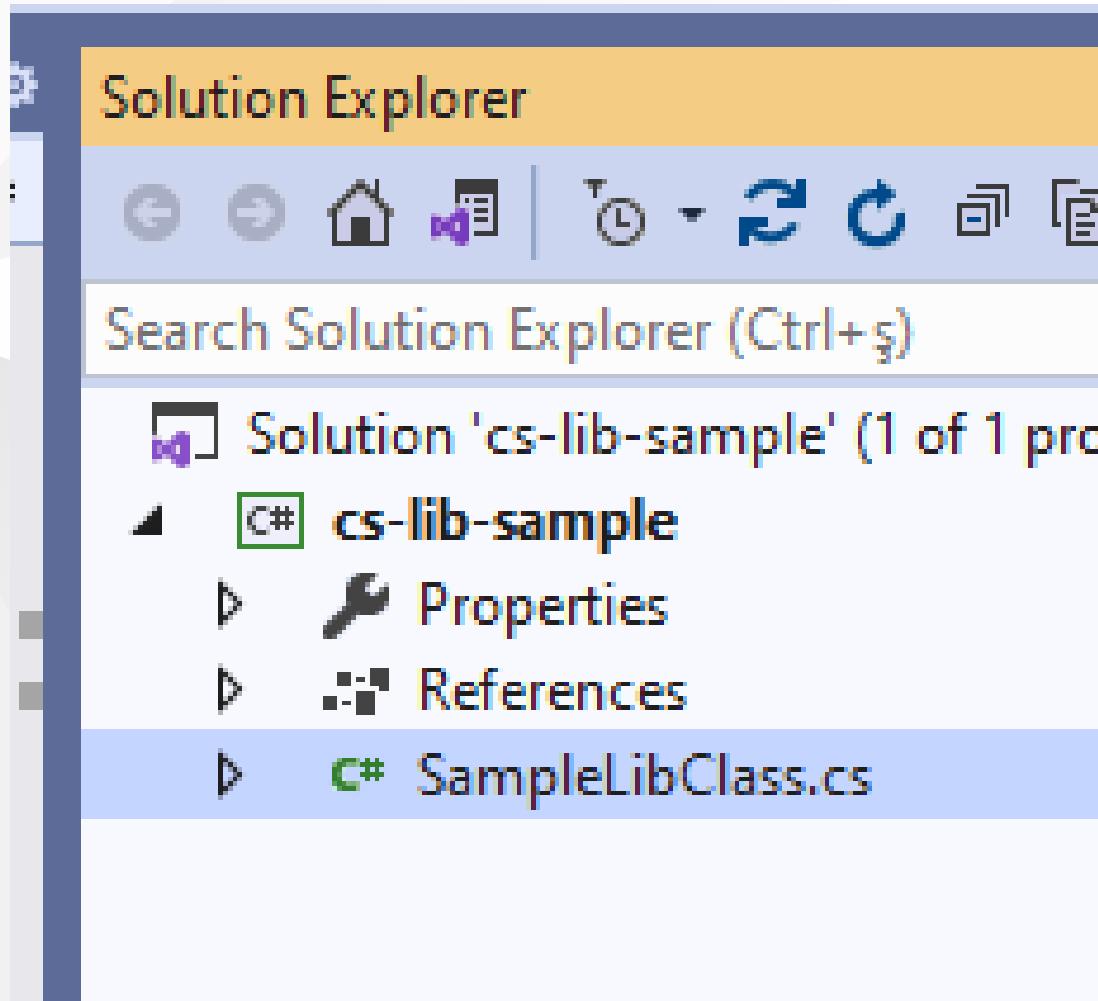
cs-lib-sample

Place solution and project in the same directory

Framework

.NET Framework 3.0

## Create library functions



```
using System;
using System.Collections.Generic;
using System.Text;

namespace cs_lib_sample
{
    public class SampleLibClass
    {
        public static string sayHelloTo(string name)
        {
            string result = String.Empty;

            if (!String.IsNullOrEmpty(name))
            {
                result = "Hello " + name;
            }
            else
            {
                result = "Hello There";
            }

            Console.WriteLine(result);

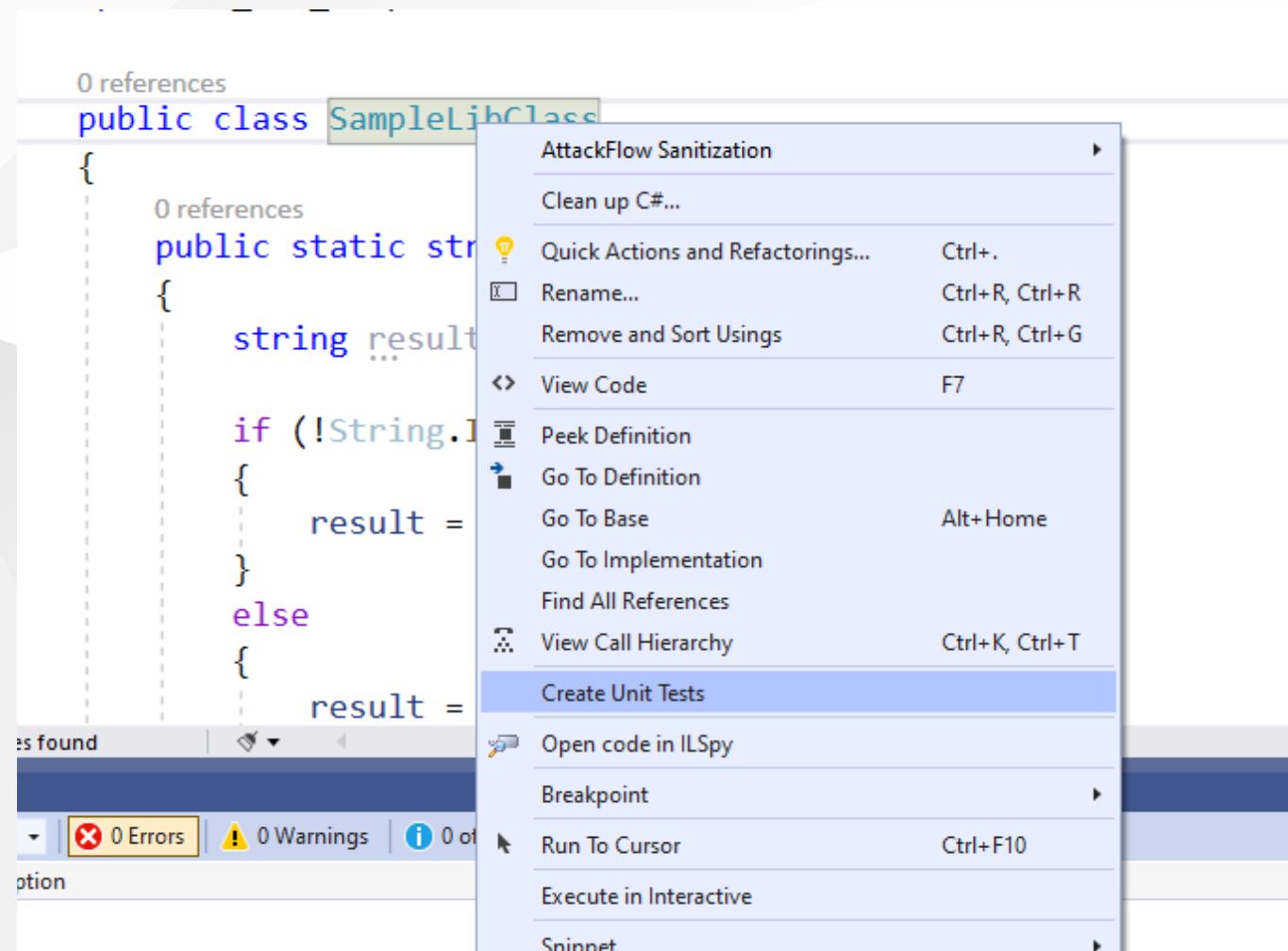
            return result;
        }

        public static int sum(int a, int b)
        {
            int c = 0;
            c = a + b;
            return c;
        }

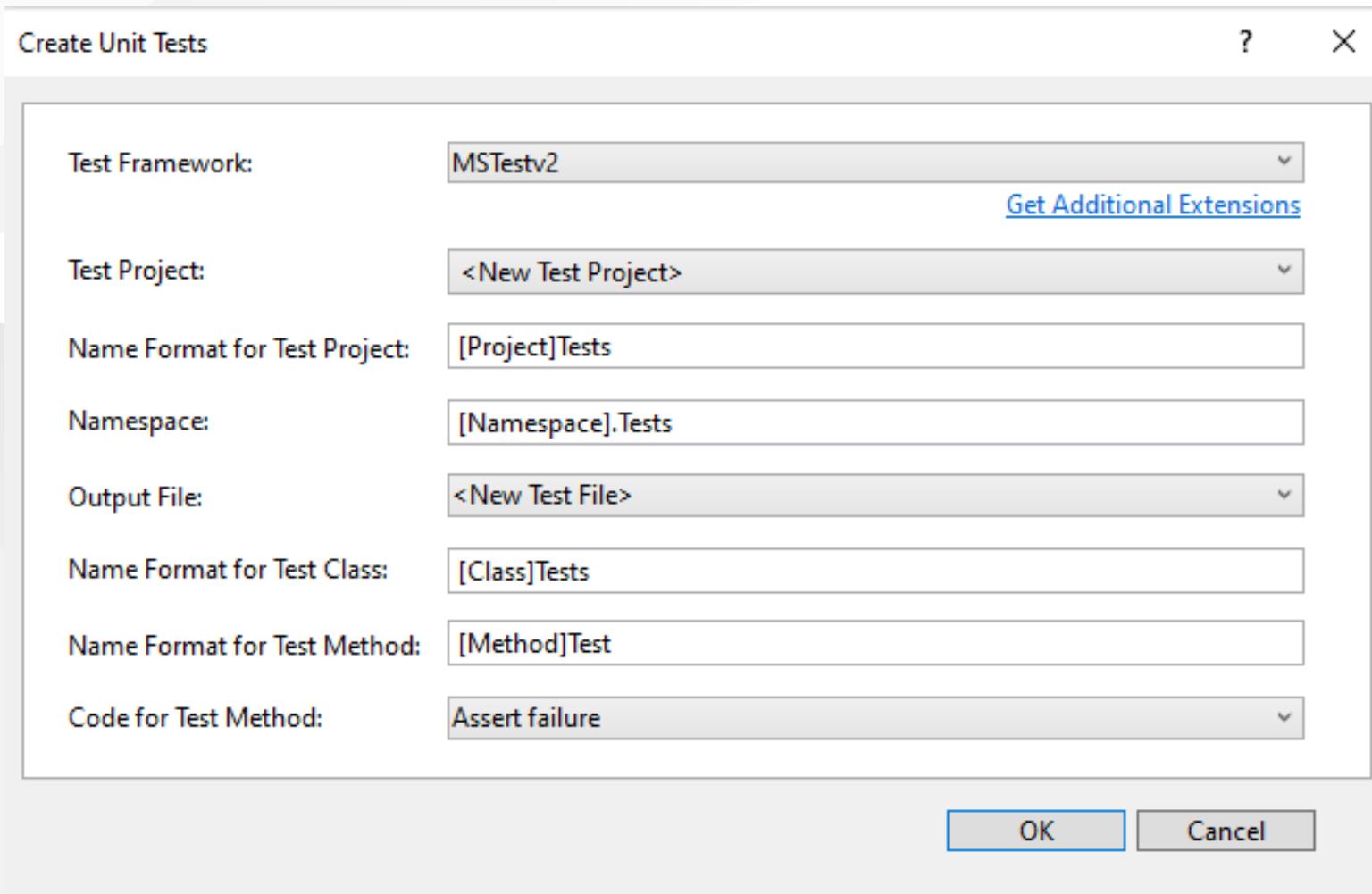
        public int multiply(int a, int b)
        {
            return a * b;
        }
    }
}
```



right click and then create unit test project



press OK



# enter test code

```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using cs_lib_sample;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace cs_lib_sample.Tests
{
    [TestClass()]
    public class SampleLibClassTests
    {

        [TestMethod()]
        public void testSayHelloTo()
        {

            Assert.AreEqual("Hello Computer", SampleLibClass.sayHelloTo("Computer"), "Regular say hello should work");
        }
        [TestMethod()]
        public void testSayHelloToWrong()
        {
            Assert.AreEqual("Hello All", SampleLibClass.sayHelloTo("Computer"), "Regular say hello won't work");
        }

        [TestMethod()]
        public void testSumCorrect()
        {
            Assert.AreEqual(9, SampleLibClass.sum(4, 5), "Regular sum should work");
        }

        [TestMethod()]
        public void testSumWrong()
        {
            Assert.AreEqual(10, SampleLibClass.sum(4, 5), "Regular sum shouldn't work");
        }

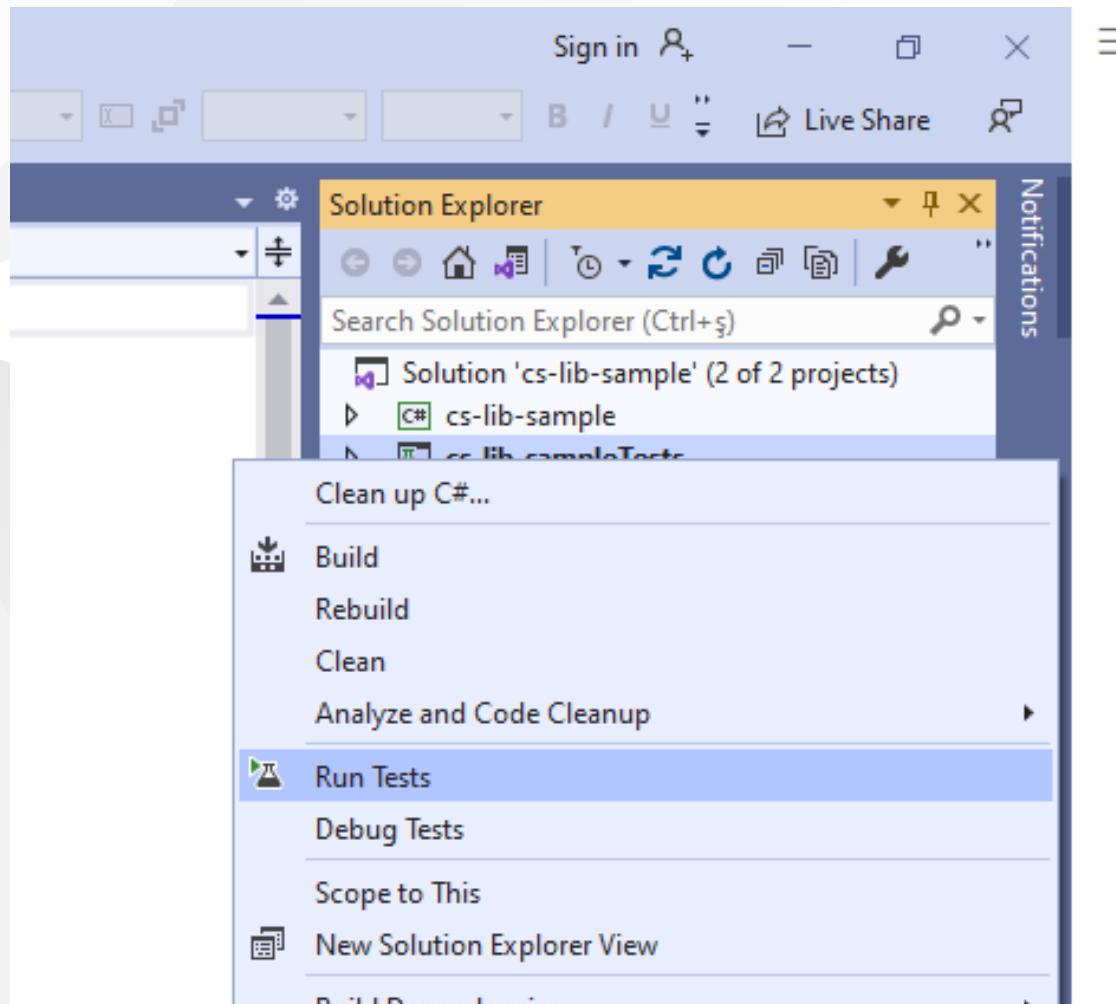
        [TestMethod()]
        public void testMultiply()
        {
            SampleLibClass sampleLib = new SampleLibClass();

            Assert.AreEqual(20, sampleLib.multiply(4, 5), "Regular multiplication should work");
        }

    }
}
```



## Run tests



# you will code coverage and entered or passed branches

The screenshot shows the Microsoft Visual Studio IDE interface. The main window displays the code for `SampleLibClass`:7 public class SampleLibClass
{
 2 references | 1/2 passing
8 public static string sayHelloTo(string name)
9 {
10 string result = String.Empty;
11
12 if (!String.IsNullOrEmpty(name))
13 {
14 result = "Hello " + name;
15 }
16 else
17 {
18 result = "Hello There";
19 }
20
21 Console.WriteLine(result);
22
23 return result;
24 }
25
26 }

Below the code editor, the status bar shows "2 references | 1/2 passing". The bottom of the screen features the "Fine Code Coverage" tool window.

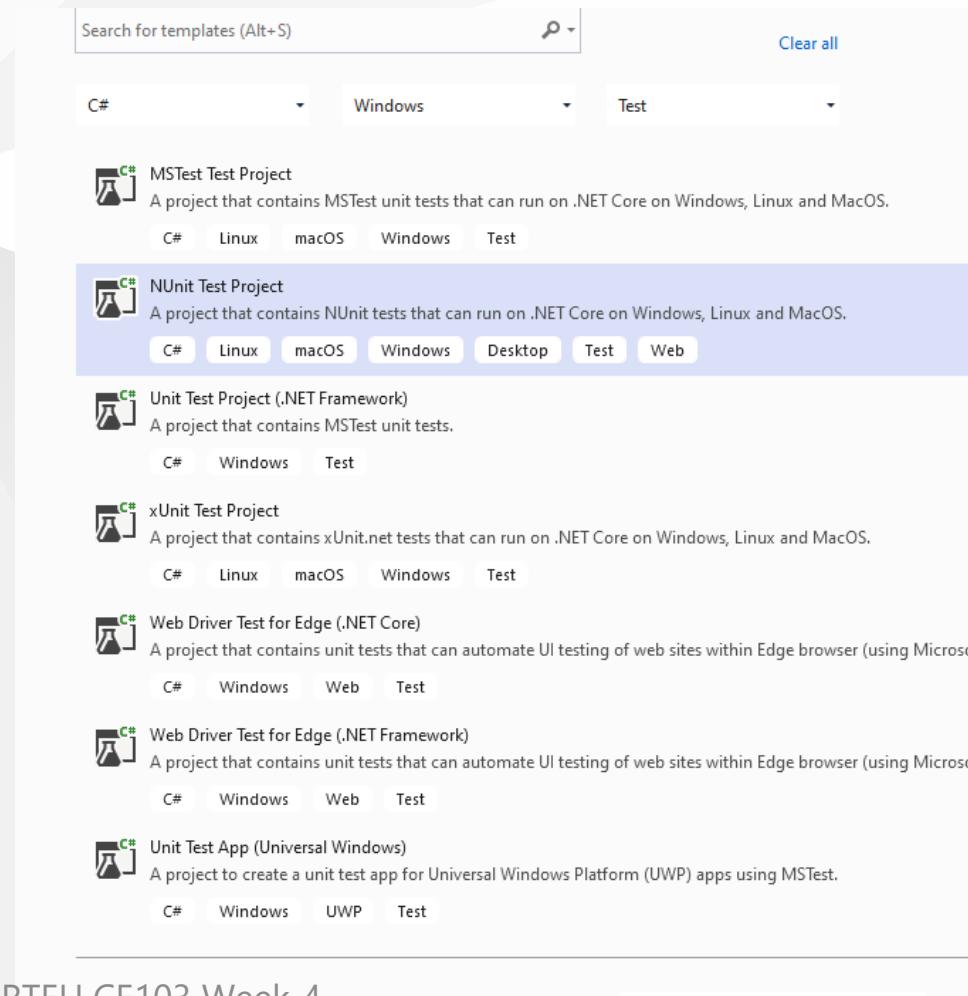
**Fine Code Coverage**

Name	Covered	Uncovered	Coverable	Total	Line coverage
- cs-lib-sample	17	3	20	39	85%
SampleLibClass	17	3	20	39	85%
- cs-lib-sampleTests	14	2	16	51	87.5%
SampleLibClassTests	14	2	16	51	87.5%

# Visual Studio Community Edition (NUnit+.NETCore)

use csharp-sample-lib for this example

create and add a unit test project to solution



# Configure your new project

NUnit Test Project

C#

Linux

macOS

Windows

Desktop

Test

Web

Project name

csharp-sample-lib-test

Location

E:\UgurCoruh\RTEU\Lectures\2021-2022 Güz CE103 - Algorithms and Programming I\Lectures\ce11

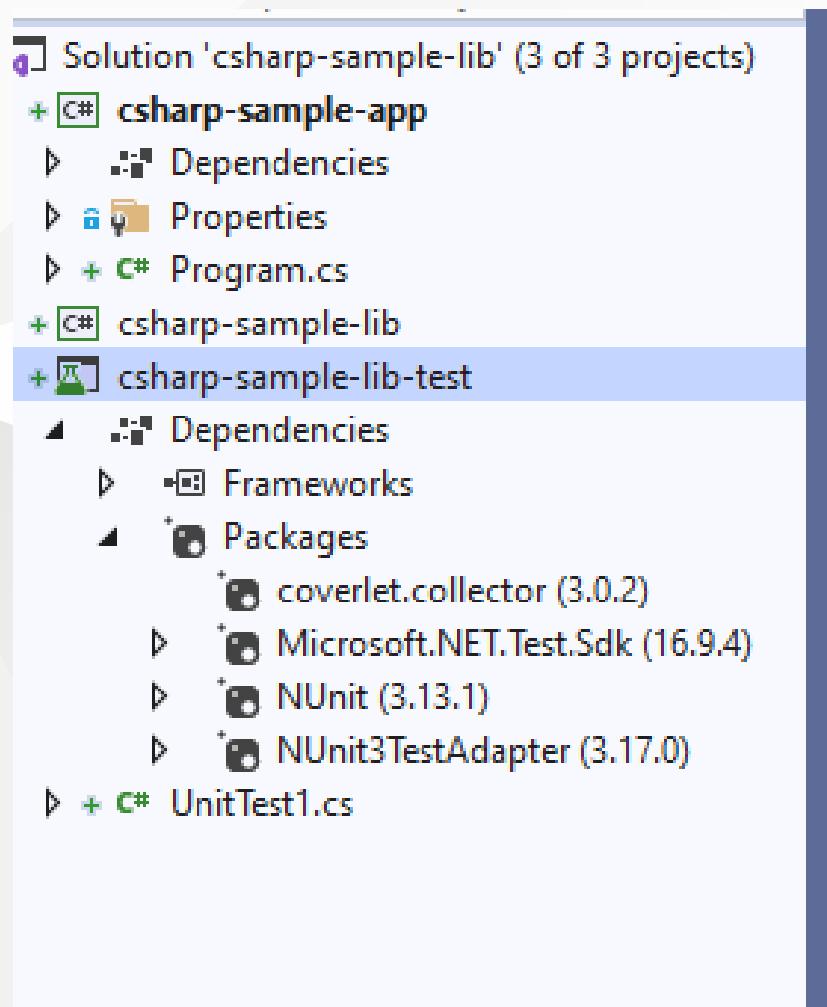
...

## Additional information

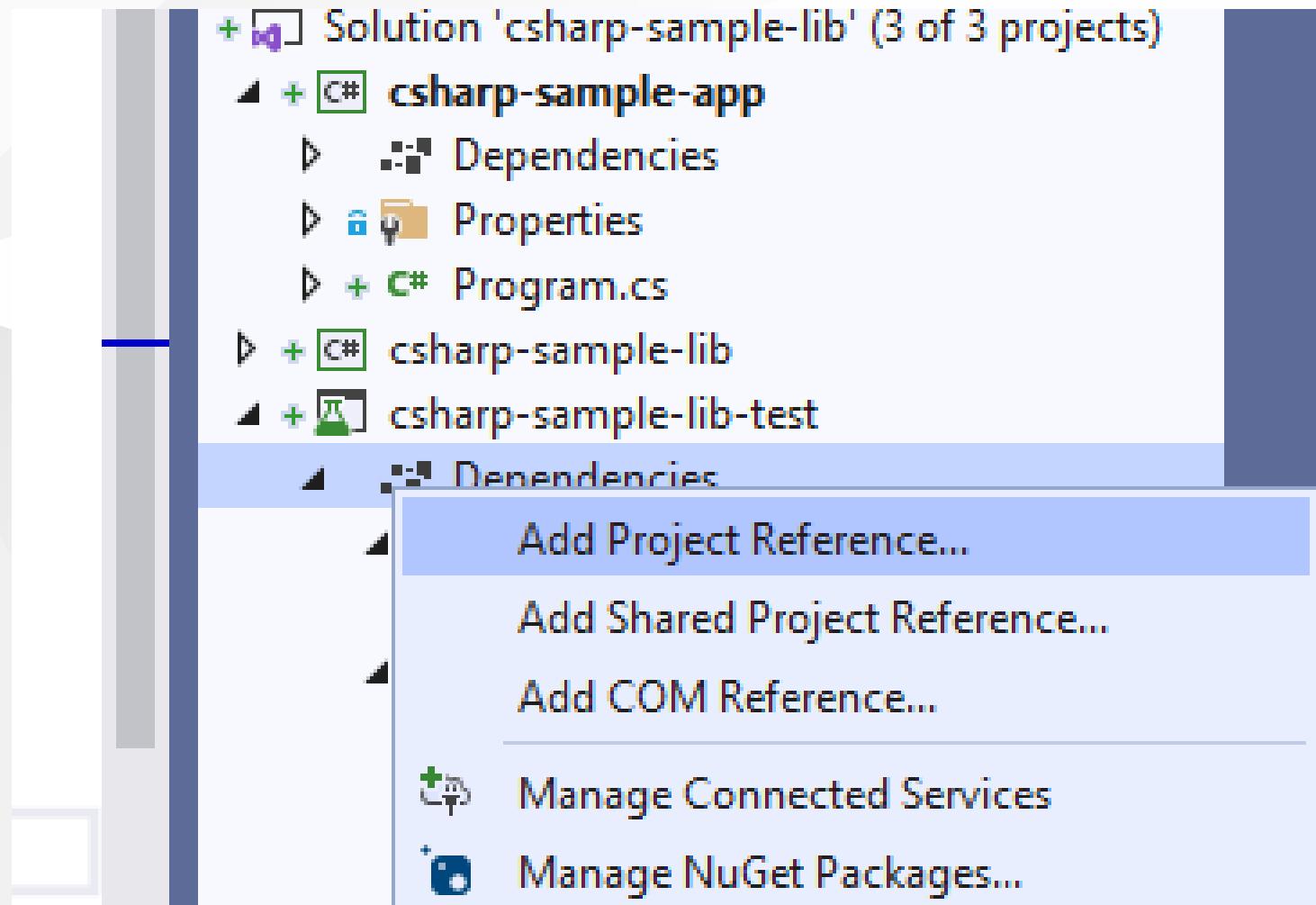
NUnit Test Project    C#    Linux    macOS    Windows    Desktop    Test    Web

Target Framework (i)

- .NET Core 3.1 (Long-term support)
- .NET Framework 4.0
- .NET Framework 4.5
- .NET Framework 4.5.1
- .NET Framework 4.5.2
- .NET Framework 4.6
- .NET Framework 4.6.1
- .NET Framework 4.6.2
- .NET Framework 4.7
- .NET Framework 4.7.1
- .NET Framework 4.7.2
- .NET Framework 4.8
- .NET Core 1.0 (Out of support)
- .NET Core 1.1 (Out of support)
- .NET Core 2.0 (Out of support)
- .NET Core 2.1 (Long-term support)
- .NET Core 2.2 (Out of support)
- .NET Core 3.0 (Out of support)
- .NET Core 3.1 (Long-term support)
- .NET 5.0 (Current)



## Add project reference



## Reference Manager - csharp-sample-lib-test

## Projects

Search (

Solution

Shared Projects

COM

Browse

	Name	Path
	csharp-sample-app	E:\UgurCoruh\RTEU\L...
<input checked="" type="checkbox"/>	csharp-sample-lib	E:\UgurCoruh\RTEU\L...

Name:  
csharp-

# SampleLibraryTestClasss in NUnit Project

```
using csharp_sample_lib;
using NUnit.Framework;

namespace csharp_sample_lib_test
{
    public class SampleLibraryTestClass
    {
        sampleLibClass sampleLib;

        [SetUp]
        public void Setup()
        {
            sampleLib = new sampleLibClass();
        }

        [Test]
        public void testSayHelloTo()
        {
            Assert.AreEqual("Hello Computer", sampleLibClass.sayHelloTo("Computer"), "Regular say hello should work");
        }

        [Test]
        public void testSayHelloToWrong()
        {
            Assert.AreEqual("Hello All", sampleLibClass.sayHelloTo("Computer"), "Regular say hello won't work");
        }

        [Test]
        public void testSumCorrect()
        {
            Assert.AreEqual(9, sampleLibClass.sum(4, 5), "Regular sum should work");
        }

        [Test]
        public void testSumWrong()
        {
            Assert.AreEqual(10, sampleLibClass.sum(4, 5), "Regular sum shouldn't work");
        }

        [Test]
        public void testMultiply()
        {
            Assert.AreEqual(20, sampleLib.multiply(4, 5), "Regular multiplication should work");
        }
    }
}
```



# sample class library

```
using System;

namespace csharp_sample_lib
{
    public class sampleLibClass
    {
        public static string sayHelloTo(string name)
        {
            string result = String.Empty;

            if (!String.IsNullOrEmpty(name))
            {
                result = "Hello " + name;
            }
            else
            {
                result = "Hello There";
            }

            Console.WriteLine(result);

            return result;
        }

        public static int sum(int a, int b)
        {
            int c = 0;
            c = a + b;
            return c;
        }

        public int multiply(int a, int b)
        {
            return a * b;
        }
    }
}
```



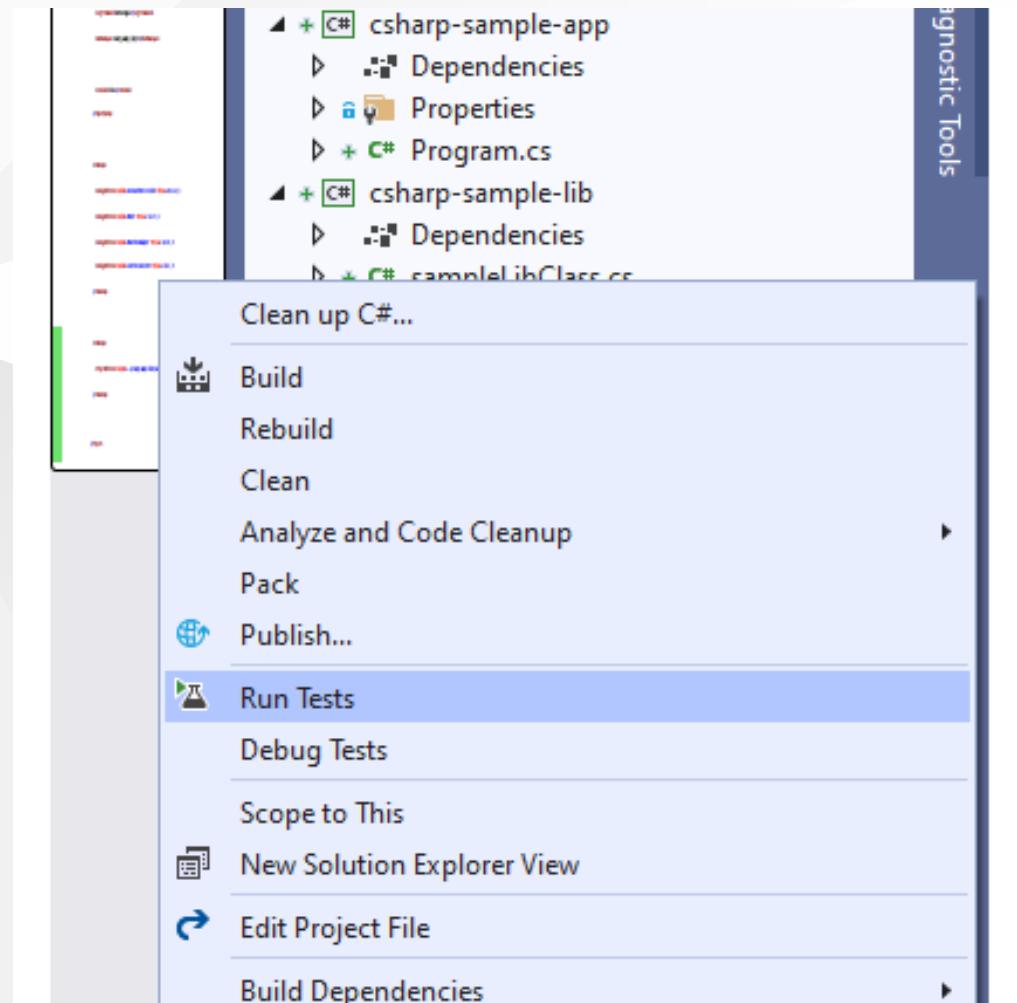
## Open test explorer and run tests

The screenshot shows the Visual Studio Test Explorer window. At the top, there are status indicators: '38' (likely the total number of tests), '144 %' (zoom level), and 'No issues found'. Below the toolbar, the Test Explorer title bar is visible. The main area displays a table of test results:

Test	Duration	Traits	Error Message
✗ csharp-sample-lib-test (5)	71 ms		
✗ csharp_sample_lib_test (5)	71 ms		
✗ SampleLibraryTestClass (5)	71 ms		
✓ testMultiply	16 ms		
✓ testSayHelloTo	< 1 ms		
✗ testSayHelloToWrong	53 ms		Regular say hello won't work. Expected string length 9 but was 14. Strings differ at index 6. Expected: "Hello All" But was:....
✓ testSumCorrect	< 1 ms		
✗ testSumWrong	2 ms		Regular sum shouldn't work. Expected: 10. But was: 9

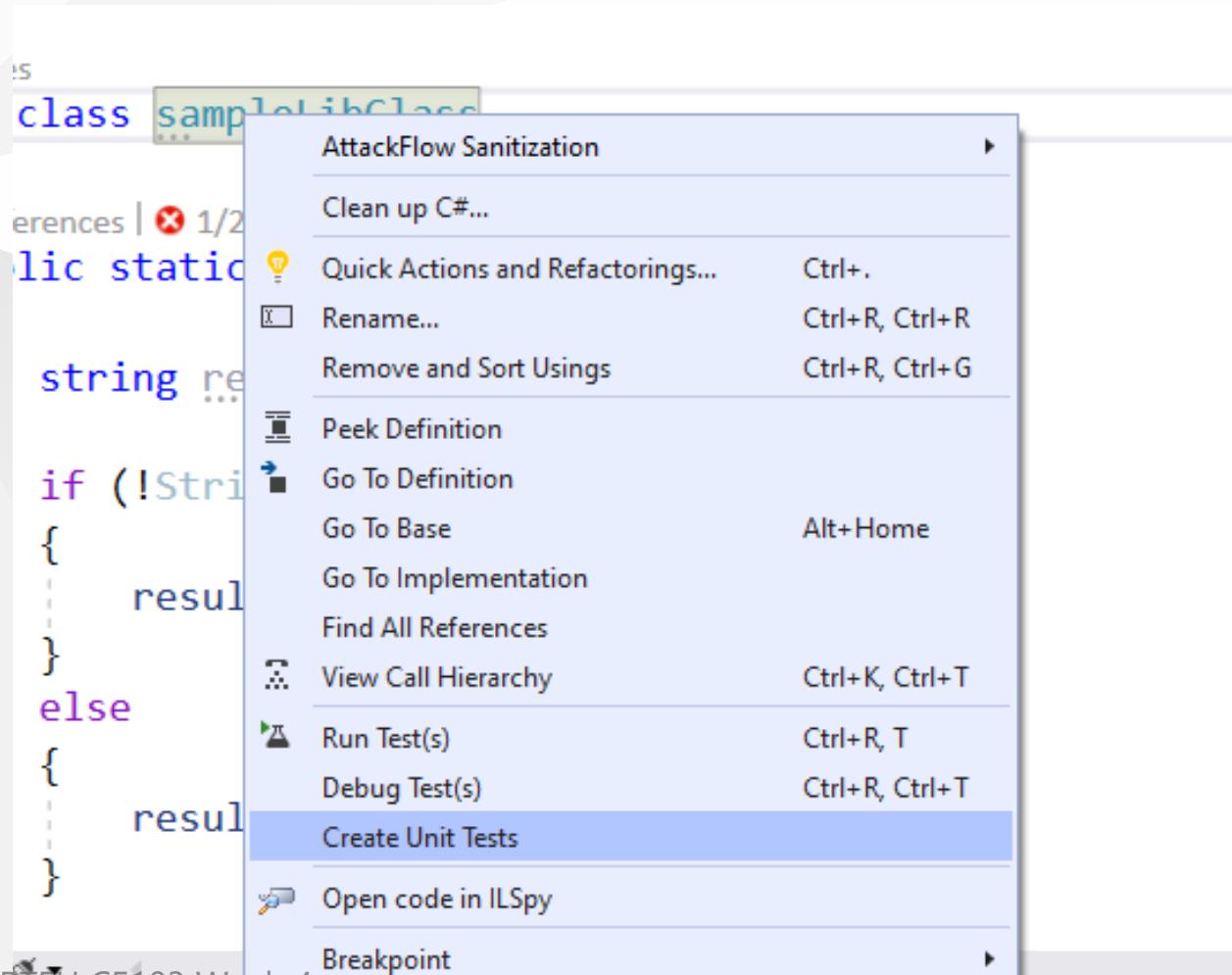
At the bottom of the window, tabs for 'Error List', 'Output', 'Find Symbol Results', and 'Test Explorer' are visible. A status bar at the very bottom shows the URL 'http://ucoruh/ce103-algorithms-and-programming-i'.

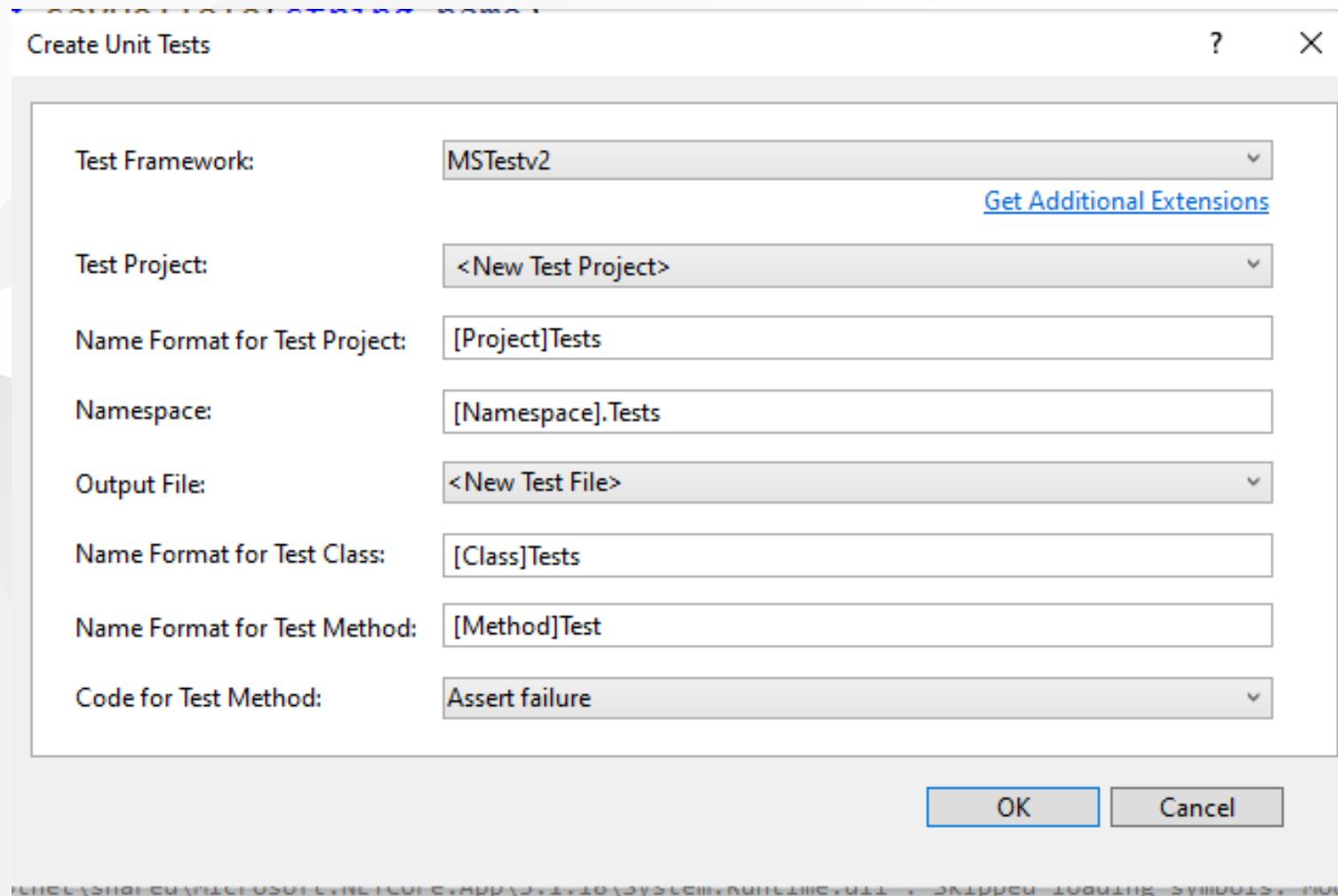
or you can run from project



Also we can create unit test from library class,

right click the sampleLibClass and select create unit tests but this option do not provide nunit tests.

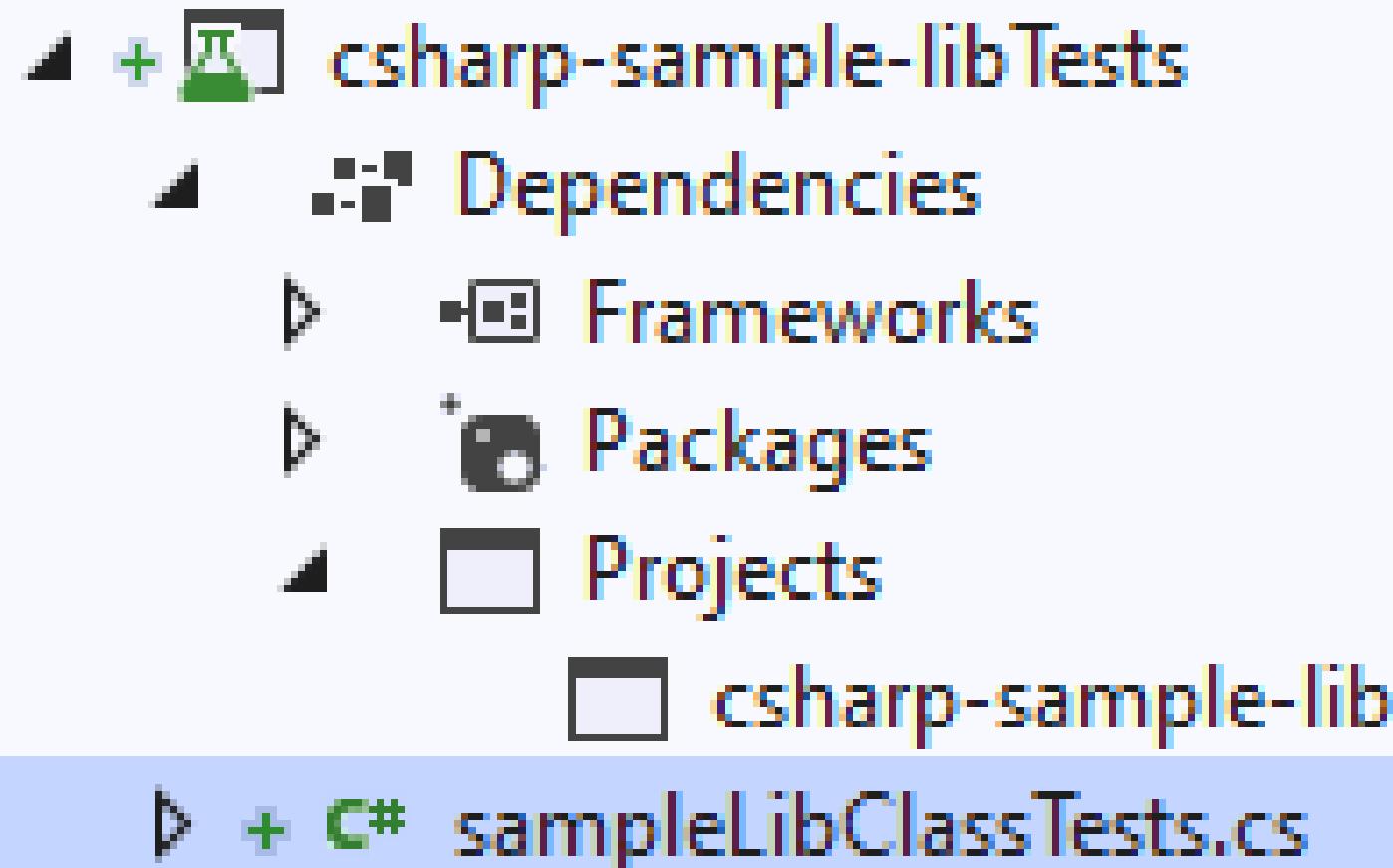




## Create Unit Tests

### Creating Unit Tests





```
using Microsoft.VisualStudio.TestTools.UnitTesting;
using csharp_sample_lib;
using System;
using System.Collections.Generic;
using System.Text;

namespace csharp_sample_lib.Tests
{
    [TestClass()]
    public class sampleLibClassTests
    {
        [TestMethod()]
        public void sayHelloToTest()
        {
            Assert.Fail();
        }

        [TestMethod()]
        public void sumTest()
        {
            Assert.Fail();
        }

        [TestMethod()]
        public void multiplyTest()
        {
            Assert.Fail();
        }
    }
}
```



we will not commit this changes and continue from nunit test project, the fine code coverage also work for nunit test but not provide inline highlighting if we run tests we will have the following outputs

The screenshot shows the Visual Studio IDE interface with the following details:

- Project Explorer:** Shows the solution 'csharp-sample-lib' containing three projects: 'csharp-sample-app', 'csharp-sample-lib', and 'csharp-sample-lib-test'. The 'csharp-sample-lib-test' project is selected.
- Code Editor:** Displays the file 'sampleLibClass.cs' with the following code:

```
1  using System;
2
3  namespace csharp_sample_lib
4  {
5      public class sampleLibClass
6      {
7          public static string sayHelloTo(string name)
8          {
9              string result = String.Empty;
10
11             if (!String.IsNullOrEmpty(name))
12             {
13                 result = "Hello " + name;
14             }
15             else
16             {
17                 result = "Hello There";
18             }
19         }
20     }
21 }
```

- Fine Code Coverage:** A bottom panel showing coverage statistics for the 'csharp-sample-lib-test' project. It includes a table and a chart.

Name	Covered	Uncovered	Coverable	Total	Line coverage
csharp-sample-lib	17	3	20	37	85%
sampleLibClass	17	3	20	37	85%
csharp-sample-lib-test	16	2	18	47	88.8%
SampleLibraryTestClass	16	2	18	47	88.8%

**Bottom Navigation:** RTEU CE103 Week 4 | Fine Code Coverage | Error List | Output | Find Symbol Results | Test Explorer

The screenshot shows the Visual Studio Test Explorer window. At the top, there are status indicators: '144 %' (zoom level), 'No issues found' (green checkmark icon), and file navigation buttons. On the right, there are settings for 'Ln: 1 Ch: 1 SPC CRLF'. The main area displays a table of test results with columns: Test, Duration, Traits, and Error Message. The 'Error Message' column contains detailed failure messages for failing tests. To the right of the table is a 'Group Summary' panel showing the total count of tests (5), total duration (197 ms), and outcomes (3 Passed, 2 Failed). Below the main window are tabs for 'Fine Code Coverage', 'Error List', 'Output', 'Find Symbol Results', and 'Test Explorer' (which is selected).

Test	Duration	Traits	Error Message
✗ csharp-sample-lib-test (5)	197 ms		
✗ csharp_sample_lib_test (5)	197 ms		
✗ SampleLibraryTestClass (5)	197 ms		
✓ testMultiply	54 ms		
✓ testSayHelloTo	< 1 ms		
✗ testSayHelloToWrong	136 ms		Regular say hello won't work. Expected string length 9 but was 14. Strings differ at index 6. Expected: "Hello All" But was:...
✓ testSumCorrect	< 1 ms		
✗ testSumWrong	7 ms		Regular sum shouldn't work. Expected: 10 But was: 9

**Group Summary**

csharp-sample-lib-test

Tests in group: 5

Total Duration: 197 ms

Outcomes

3 Passed

2 Failed

Inline code highlight is part of enterprise visual studio edition



Analyzing code coverage in Visual Studio - DEV Community

RTEU CE103 Week-4

# TL;DR

Additional information you can use OpenCover + Nunit Runner + Report Generator together to setup a code coverage report but it has complex batch running process. After a few try I decided to use fine code coverage but here is the usage not tested well.

First unit test runner tool doesn't support .Net Core

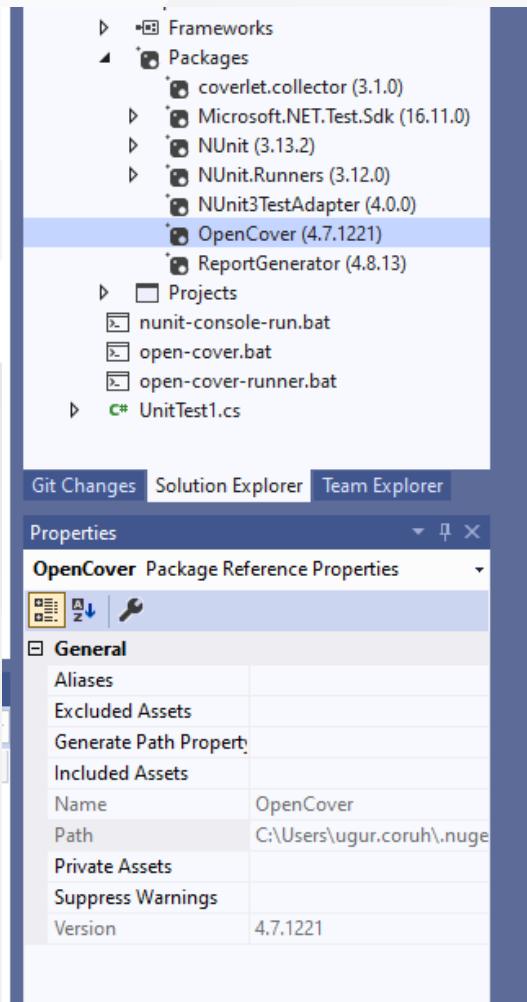
c# - The NUnit 3 driver encountered an error while executing reflected code  
([NUnit.Engine.NUnitEngineException](#)) - Stack Overflow

Follow the instructions on the link

[CMD OpenCover · sukhoi1/Useful-Notes Wiki · GitHub](#)

Install OpenCover, ReportGenerator, Nunit,Runners packages then use the package installation folder to get tools that you need

Here is a sample for open cover, select package and copy path



## Goto path and tools

```
C:\Users\ugur.coruh\.nuget\packages\opencover\4.7.1221
```

You need to setup some batch similar with following

run-test-coverage.bat

```
set pathA=C:\Users\ugur.coruh\.nuget\packages\opencover\4.7.1221\tools
set pathB=C:\Users\ugur.coruh\.nuget\packages\nunit.consolerunner\3.12.0\tools
set pathC=C:\Users\ugur.coruh\.nuget\packages\reportgenerator\4.8.13\tools\netcoreapp3.0
set dllpath=C:\Users\ugur.coruh\Desktop\csharp-sample-lib\csharp-sample-lib-test\bin\Debug\netcoreapp3.1

"%pathA%\OpenCover.Console.exe" ^
-targetargs:"%dllpath%\csharp-sample-lib-test.dll" ^
-filter:"+[csharp-sample-lib]* -[*test]*" ^
-target:"%pathB%\nunit3-console.exe" ^
-output:"%dllpath%\coverReport.xml" ^
-skipautoprops -register:user && "%pathC%\ReportGenerator.exe" -reports:"%dllpath%\coverReport.xml" -targetdir:"%dllpath%\coverage"
pause
```

but nunit3-console.exe gives error

```
    at NUnit.Engine.Runners.ProcessRunner.RunTests(ITestEventListener listener, TestFilter filter)
-->
InvalidCastException
Unable to cast transparent proxy to type 'System.Web.UI.ICallbackEventHandler'.
    at NUnit.Framework.Api.FrameworkController.LoadTestsAction..ctor(FrameworkController controller, Object handler)

Test Run Summary
Overall result: Failed
Test Count: 0, Passed: 0, Failed: 0, Warnings: 0, Inconclusive: 0, Skipped: 0
Start time: 2021-10-24 23:01:09Z
End time: 2021-10-24 23:01:17
```



for this compatibility issues I prefer to use fine code coverage extension.

OpenCover related studies

[Code coverage of manual or automated tests with OpenCover for .NET applications – Automation Rhapsody](#)

[Code coverage of .NET Core unit tests with OpenCover – Automation Rhapsody](#)

Sample OpenCover report

[Summary - Coverage Report](#)

## **Download and Setup OpenCover, NUnit Console, Report Generator without Package Manager**

You can also download the tools from github project pages and install on your operating system,

# OpenCover

[Releases · OpenCover/opencover · GitHub](#)

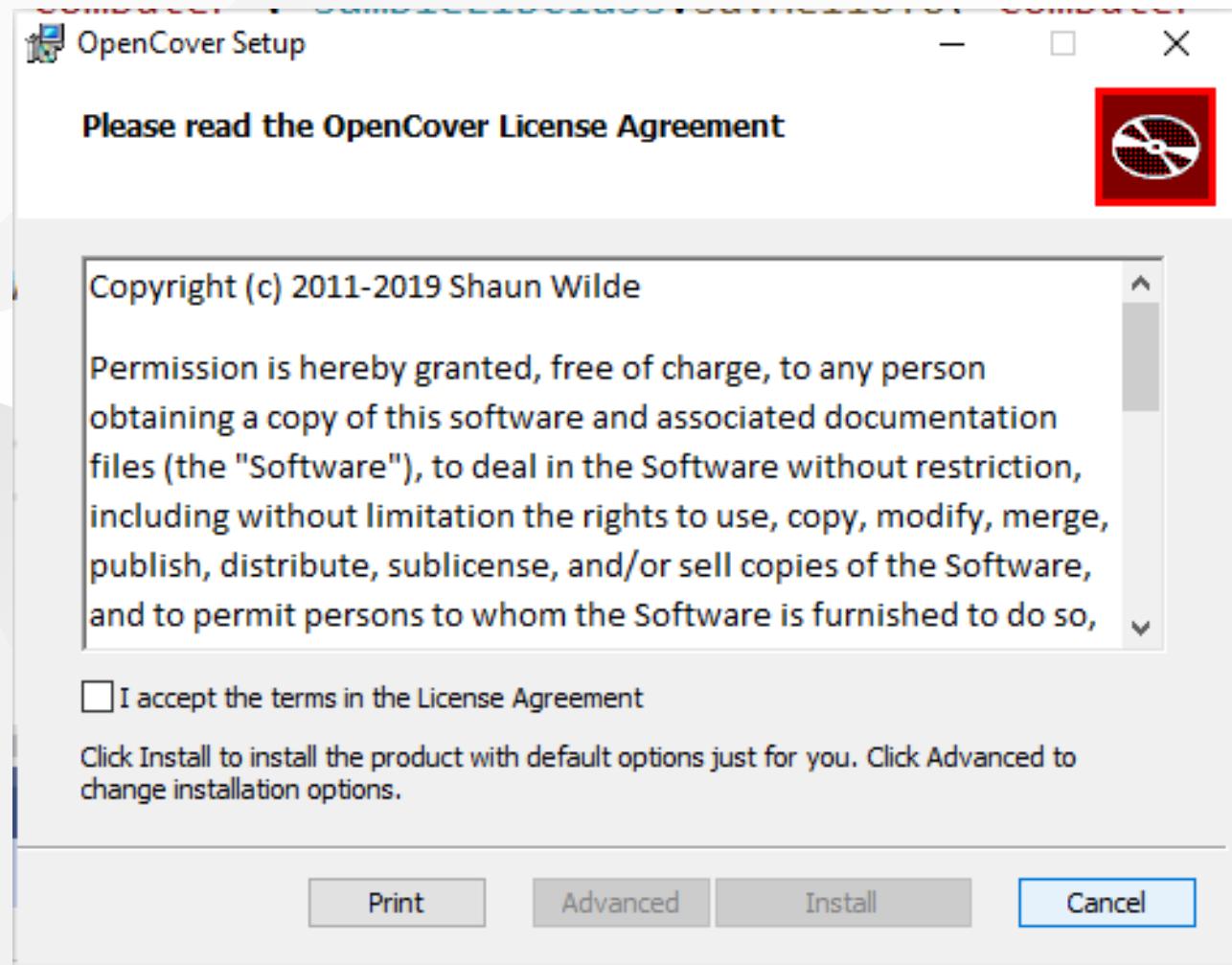
**OpenCover (Release) 4.7.1221** Latest

Merge pull request #1040 from sawilde/big/1029\_chocolatey\_dependency  
add dependency to dotnet 4.7.2 to chocolatey packages

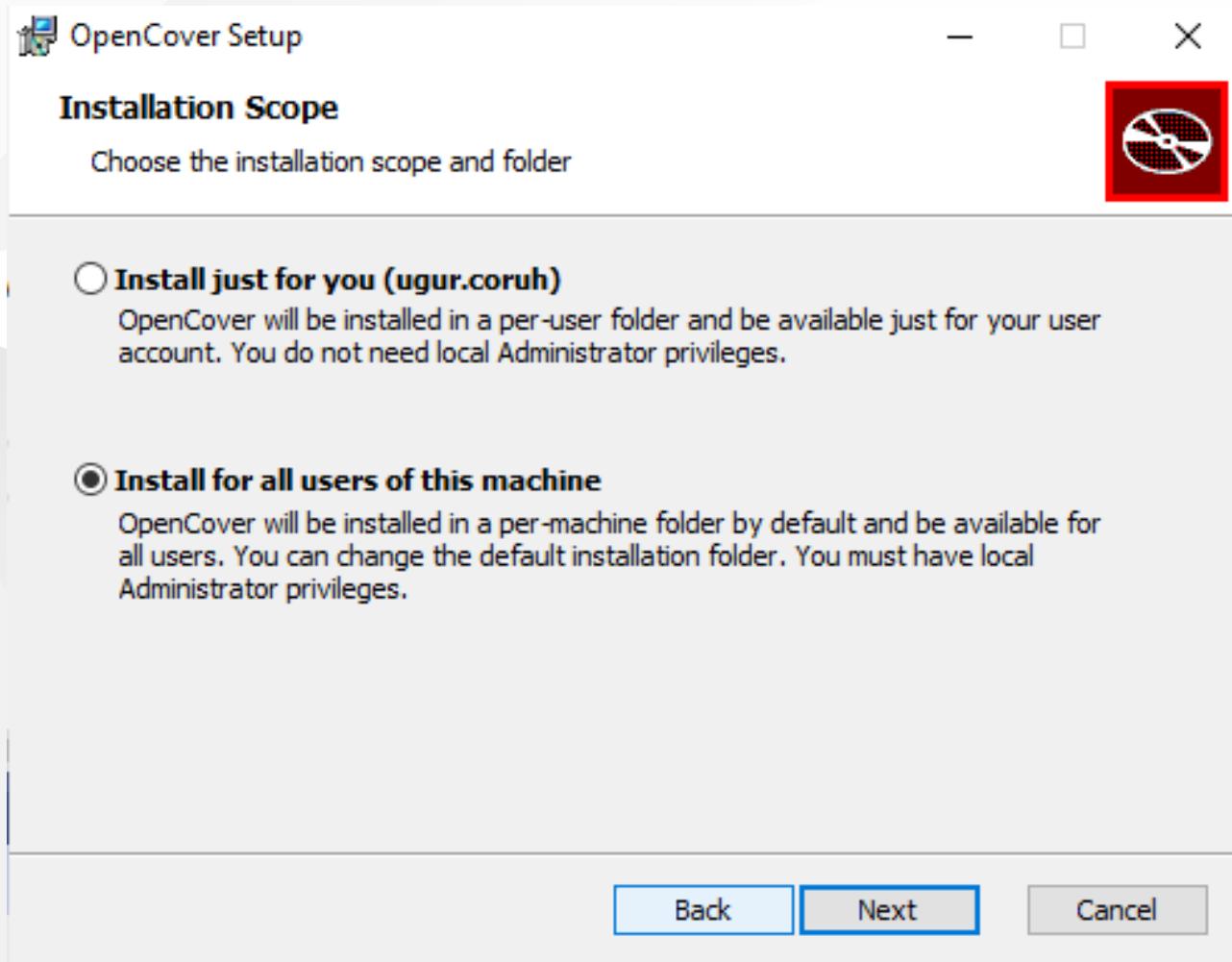
▼ Assets 6

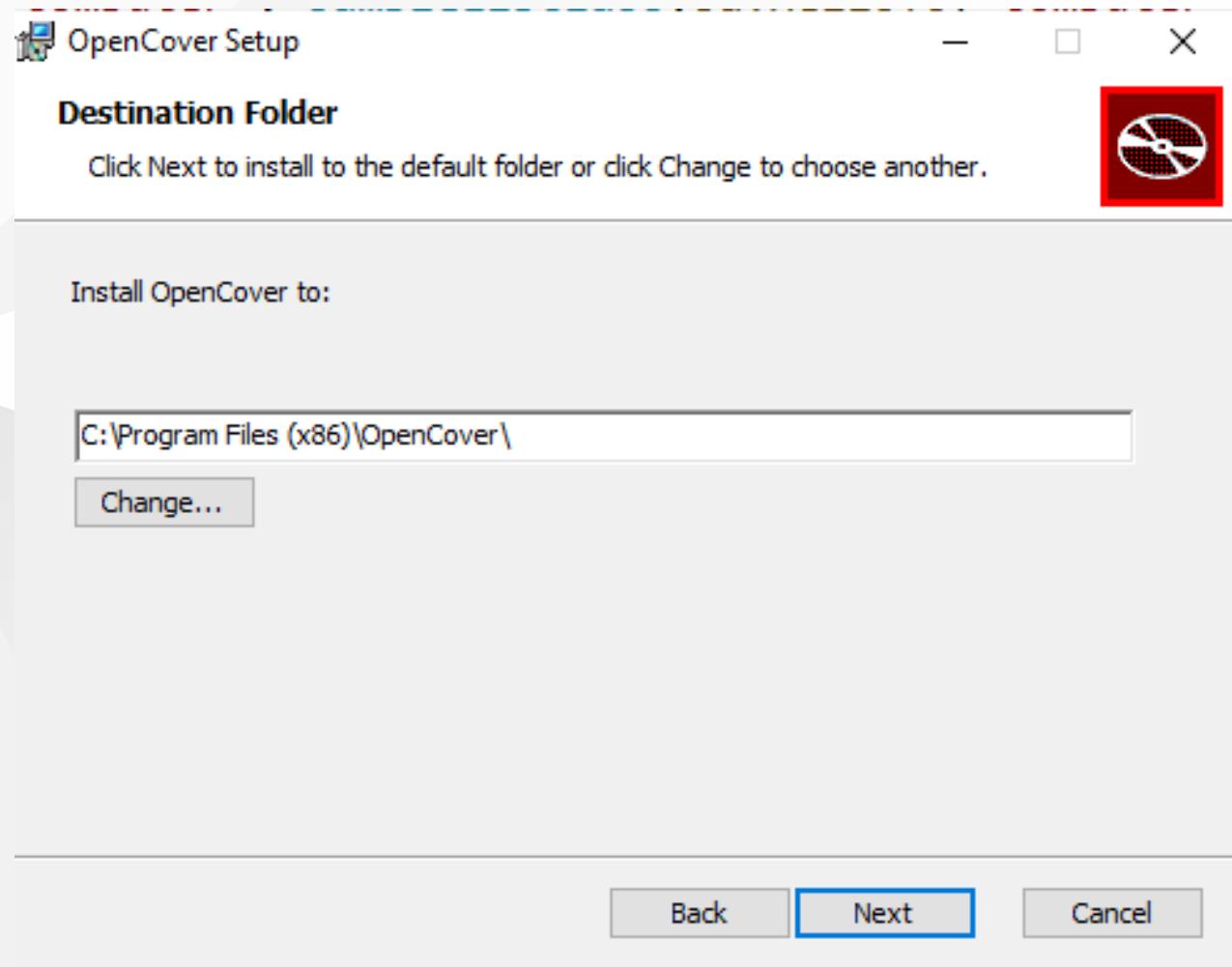
 <a href="#">checksum.installer.txt</a>	66 Bytes
 <a href="#">checksum.zip.txt</a>	66 Bytes
 <a href="#">opencover.4.7.1221.msi</a>	8.09 MB
 <a href="#">opencover.4.7.1221.zip</a>	7.76 MB
 <a href="#">Source code (zip)</a>	
 <a href="#">Source code (tar.gz)</a>	

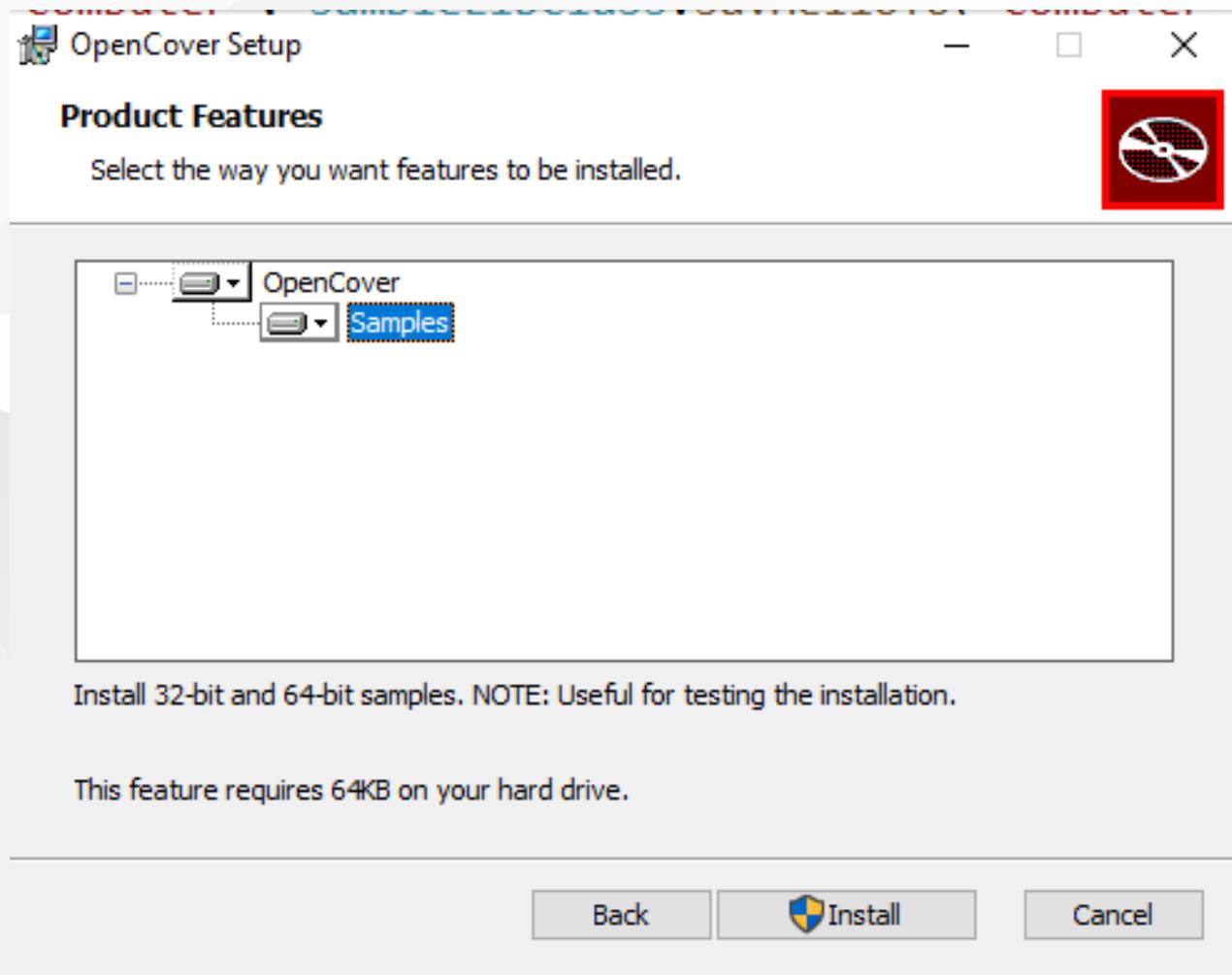


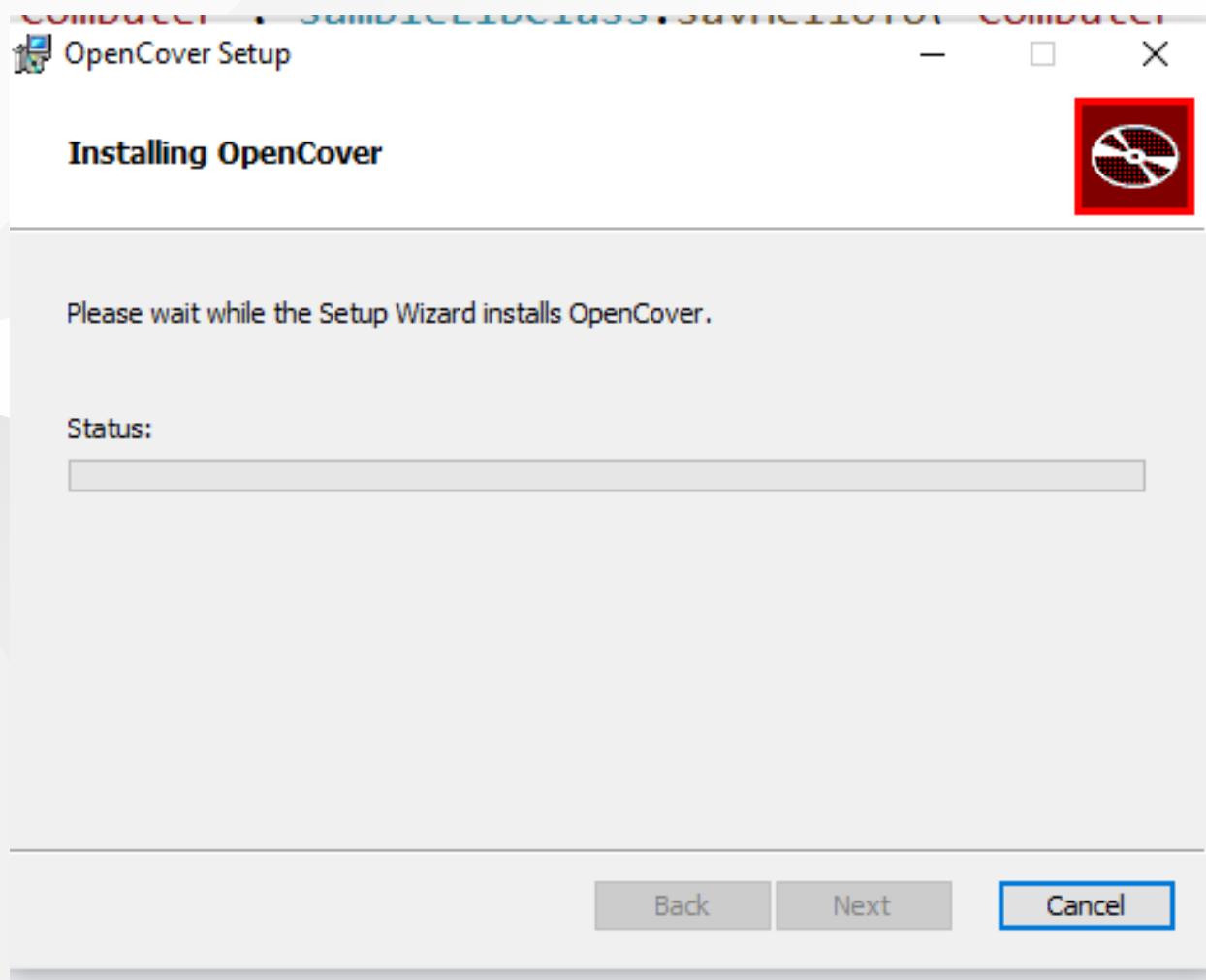


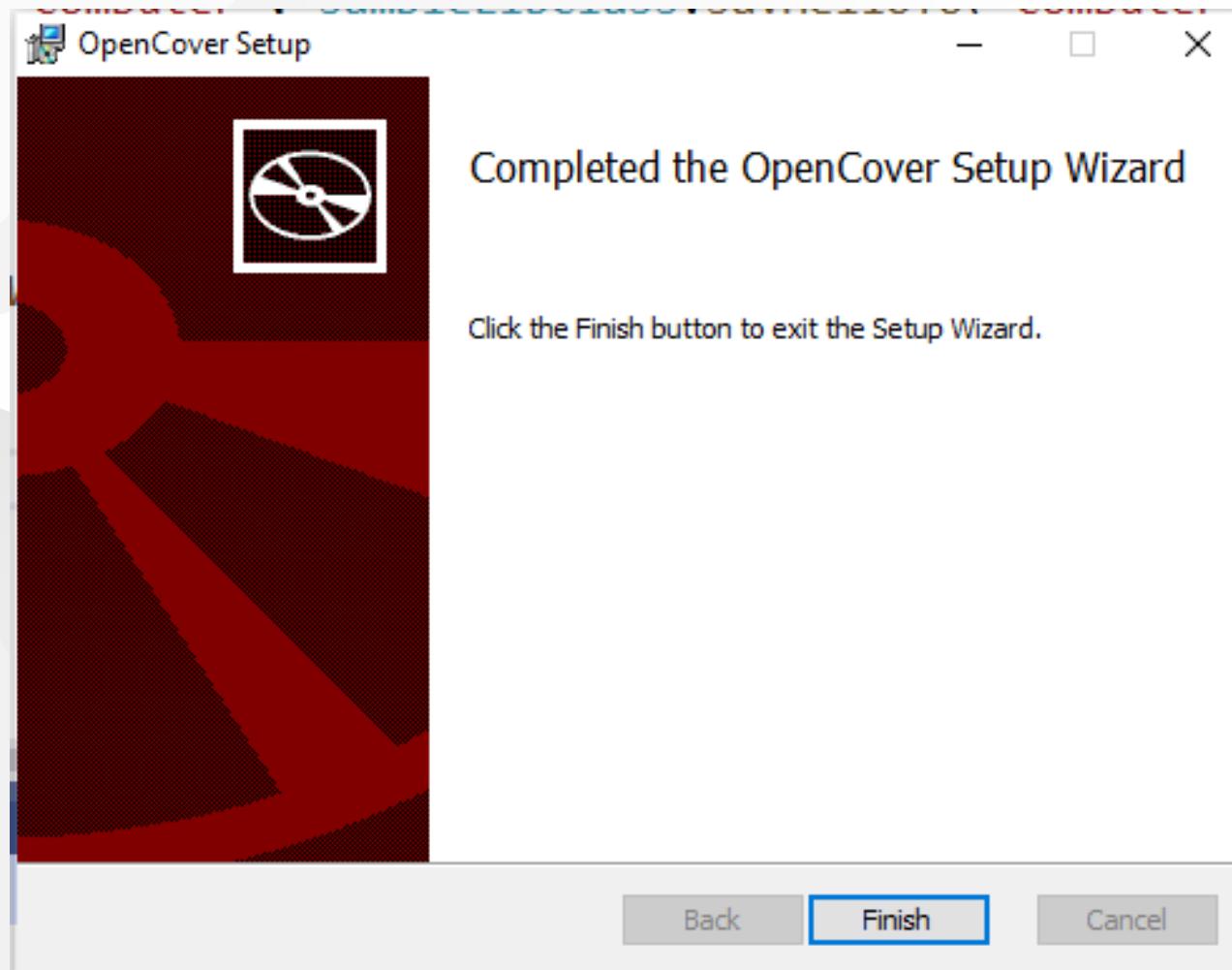
## Select advanced and then install for all users











	<a href="#">Mono.Cecil.Mono.dll</a>	9/13/2021
	<a href="#">Mono.Cecil.Pdb.dll</a>	9/15/2021
	<a href="#">Mono.Cecil.Rocks.dll</a>	9/15/2021
	<a href="#">Newtonsoft.Json.dll</a>	11/9/2019
	<a href="#">OpenCover.Console.exe</a>	6/19/2021
	<a href="#">OpenCover.Console.exe.config</a>	6/19/2021
	<a href="#">OpenCover.Console.pdb</a>	6/19/2021
	<a href="#">OpenCover.Extensions.dll</a>	6/19/2021
	<a href="#">OpenCover.Extensions.pdb</a>	6/19/2021

## System variables

Variable	Value
OPCH	C:\Program Files (x86)\OpenCover
OS	Windows_NT
Path	C:\Program Files\Java\jdk-16.0.1\bin;C:\Program Files\Java\jre1.8.0... .COM;.EXE;.BAT;.CMD;.VBS;.VBE;.JS;.JSE;.WSF;.WSH;.MSC;.PY;.PYW
PROCESSOR_ARCHITECTURE	AMD64
PROCESSOR_IDENTIFIER	Intel64 Family 6 Model 94 Stepping 3, GenuineIntel
PROCESSOR_LEVEL	6

New... Edit... Delete

OK Cancel

C:\Program Files\LLVM\bin
C:\Program Files\Git\cmd
%OPCH%

OK Cancel

[...]\Windows\system32\cmd.exe

Microsoft Windows [Version 10.0.19043.1288]  
© Microsoft Corporation. All rights reserved.

C:\Users\ugur.coruh>OpenCover.Console  
Launching OpenCover 4.7.1221.0

Incorrect Arguments: The target argument is required

Usage:

```
["]-target:<target application>["]
[[["]-targetdir:<target directory>["]]
[[["]-searchdirs:<additional PDB directory>[;<additional PDB
[["]-targetargs:<arguments for the target process>["]]
```

# ReportGenerator

Release ReportGenerator\_4.8.13 · danielpalme/ReportGenerator · GitHub

## ReportGenerator\_4.8.13 Latest

github-actions released this 27 days ago · 4 commits to master since this release · v4.8.13 · e552cc6

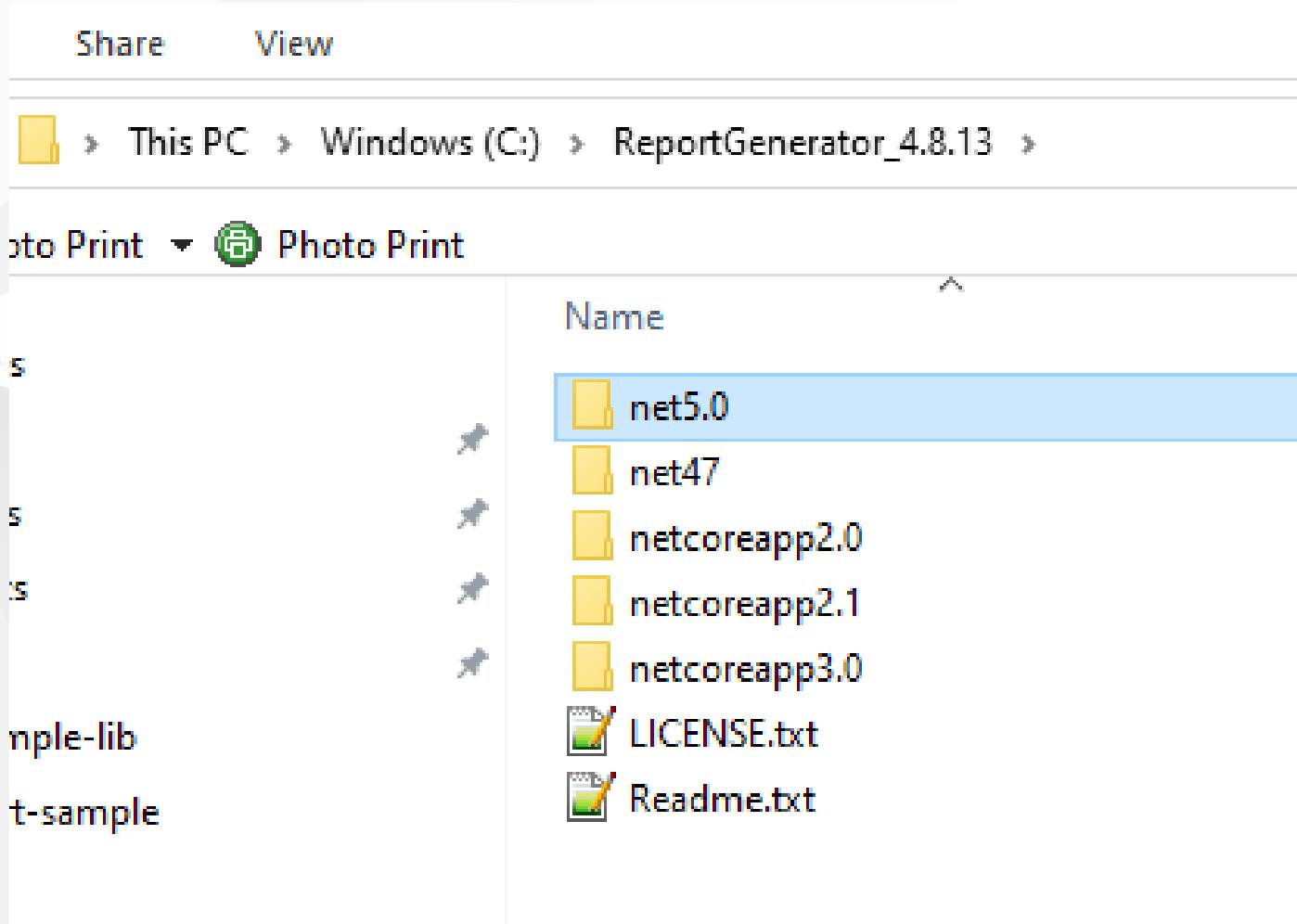
This release requires .NET 4.7 or .NET Core 2.x/3.x/5.x.

### Changes:

- #441: Added method coverage to reports
- #445: Added support for better custom logging
- #450: Conditional file numbers in class report

### Assets 3

<a href="#">ReportGenerator_4.8.13.zip</a>	13.2 MB
<a href="#">Source code (zip)</a>	
<a href="#">Source code (tar.gz)</a>	



# NUnit Console

## Downloads

The screenshot shows the 'Downloads' section of the NUnit website. At the top, there's a navigation bar with links for News, Download, Documentation, Contact, Twitter, Slack, and GitHub. Below that is a header with the title 'Downloads' and a download icon.

**Download Types**

The preferred way to download NUnit is through the [NuGet package manager](#). The latest releases can always be found on the relevant [GitHub releases pages](#).

**Latest NUnit 3 Releases**

<a href="#">NUnit 3.13.2</a>	April 27, 2021
<a href="#">NUnit Console 3.12</a>	January 17, 2021
<a href="#">NUnit Test Adapter 3.17</a>	July 11, 2020
<a href="#">NUnit Test Generator 2.3</a>	September 20, 2019
<a href="#">NUnit 3 Template for dotnet new CLI</a>	

**Latest NUnit 2 Release**

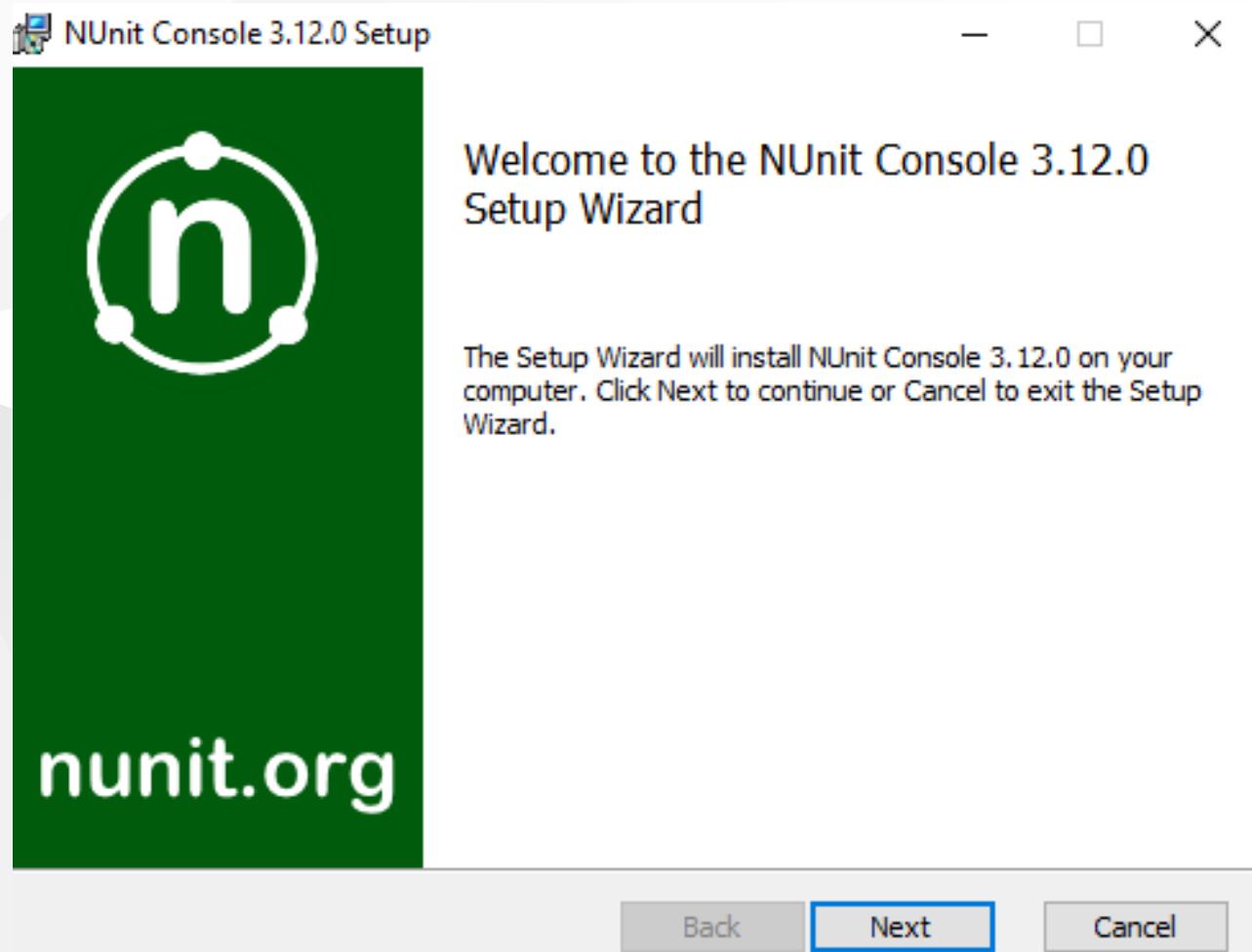
<a href="#">NUnit 2.7.1</a>	August 19, 2019
<a href="#">NUnit Test Adapter 2.2</a>	June 5, 2019

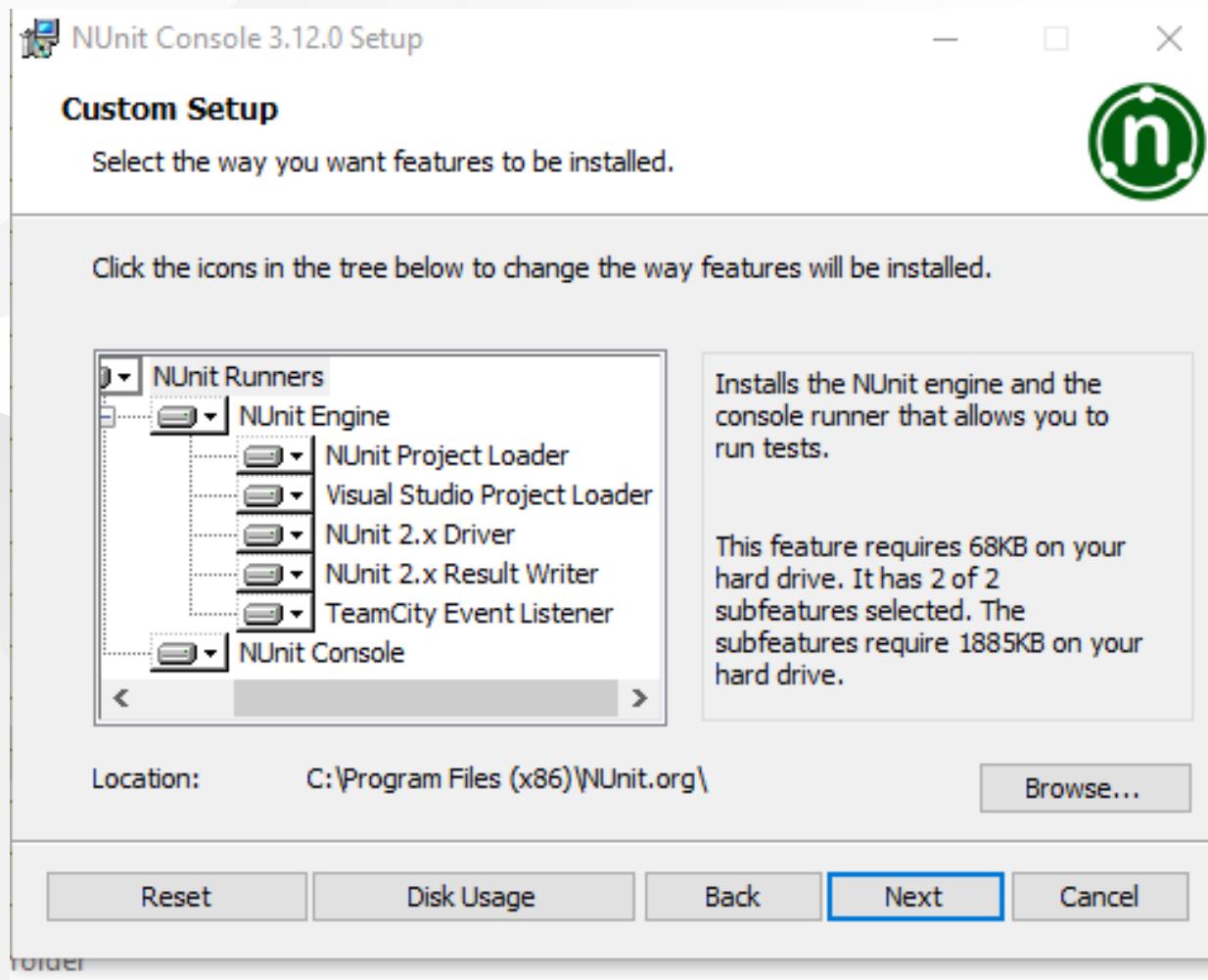
## Older Releases

These releases are needed by many people for legacy work, so we keep them around for download. Bugs are accepted on older releases only if they can be reproduced on a current release.

▼ Assets 10

	<a href="#">nunit-console-runner.3.12.0.nupkg</a>	733 KB
	<a href="#">NUnit.Console-3.12.0.msi</a>	1.04 MB
	<a href="#">NUnit.Console-3.12.0.zip</a>	14.4 MB
	<a href="#">NUnit.Console.3.12.0.nupkg</a>	19.2 KB
	<a href="#">NUnit.ConsoleRunner.3.12.0.nupkg</a>	746 KB
	<a href="#">NUnit.Engine.3.12.0.nupkg</a>	1 MB
	<a href="#">NUnit.Engine.Api.3.12.0.nupkg</a>	42.8 KB
	<a href="#">NUnit.Runners.3.12.0.nupkg</a>	19.3 KB
	<a href="#">Source code (zip)</a>	
	<a href="#">Source code (tar.gz)</a>	





	Name	Date modified	Type	Size
	addin	10/24/2021 11:30 PM	File folder	
	agents	10/24/2021 11:30 PM	File folder	
	nunit.bundle.addins	4/2/2018 2:18 PM	ADDINS File	1 KB
	nunit.engine.api.dll	1/23/2021 3:03 PM	Application exten...	18 KB
	nunit.engine.api.xml	1/23/2021 3:03 PM	XML File	55 KB
	nunit.engine.core.dll	1/23/2021 3:03 PM	Application exten...	91 KB
	nunit.engine.dll	1/23/2021 3:03 PM	Application exten...	54 KB
	nunit3-console.exe	1/23/2021 3:04 PM	Application	163 KB
	nunit3-console.exe.config	12/27/2020 3:39 PM	Configuration Sou...	2 KB
	testcentric.engine.metadata.dll	9/3/2020 6:49 PM	Application exten...	173 KB

## NUnit + MSTest Batch Report Generation (Not Tested)

[OpenCover and ReportGenerator Unit Test Coverage in Visual Studio 2013 and 2015 – CodeHelper.Net](#)

[OpenCover and ReportGenerator Unit Test Coverage in Visual Studio 2013 and 2015 - CodeProject](#)

## Java Unit Tests



## Eclipse IDE (JUnit4 , JUnit5)

In this sample we will create two example for similar library

Please check the following links

[JUnit 5 tutorial - Learn how to write unit tests](#)

[JUnit 5](#)

[JUnit 5 User Guide](#)

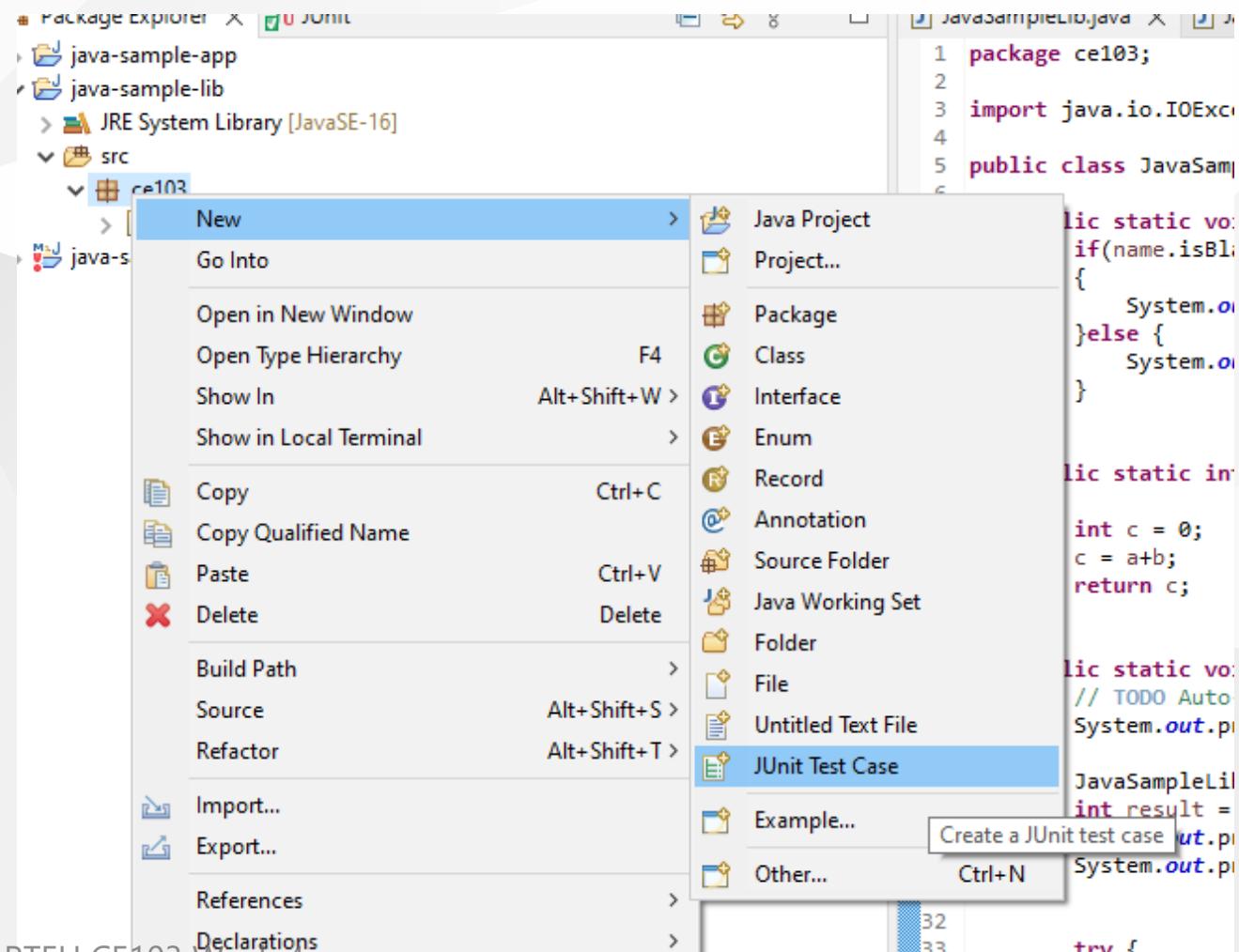
<https://www.eclemma.org/>

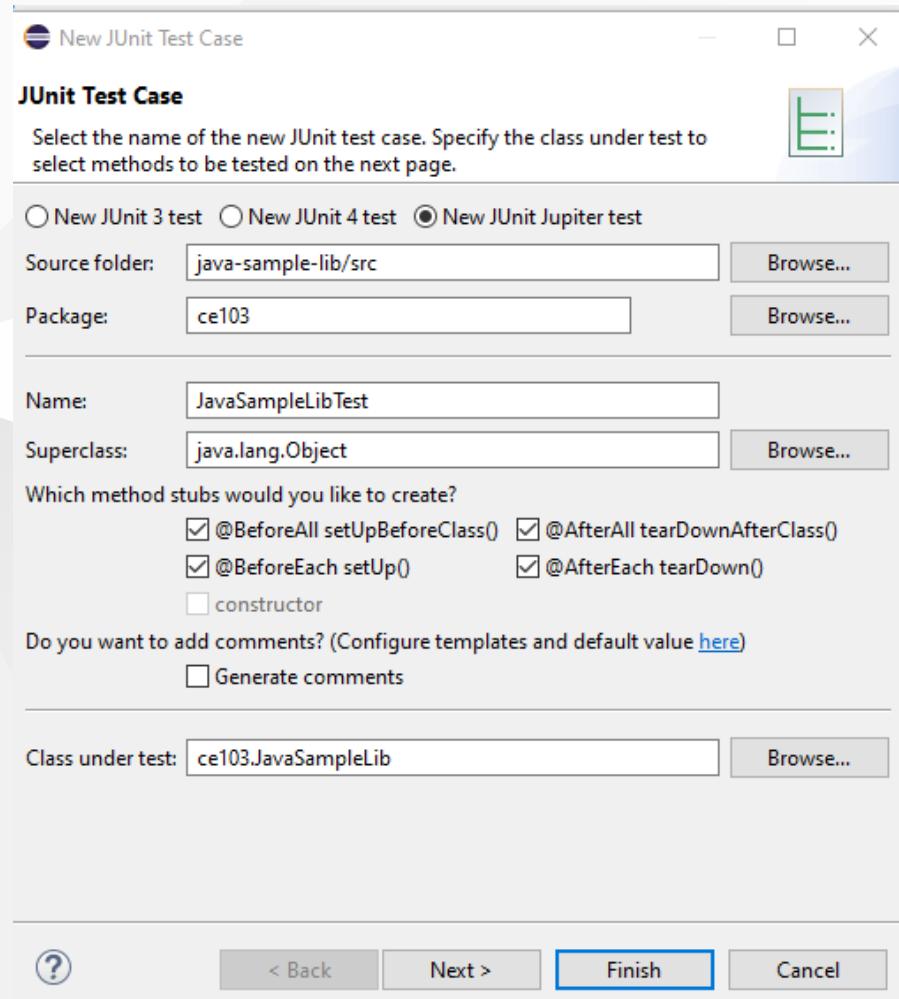
[JUnit Hello World Example - Examples Java Code Geeks - 2021](#)

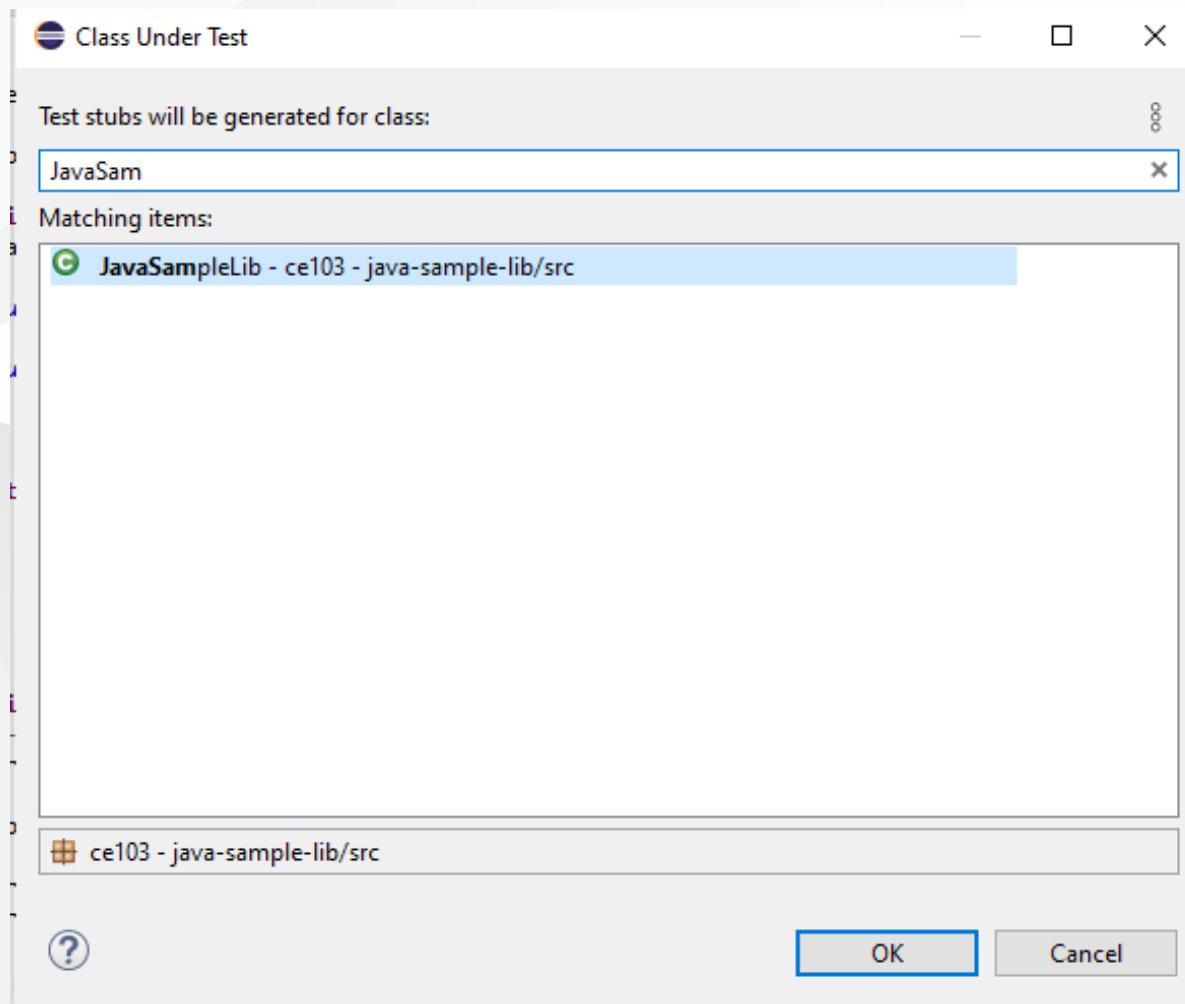
<https://yasinmemic.medium.com/java-ile-unit-test-yazmak-birim-test-ca15cf0d024b>

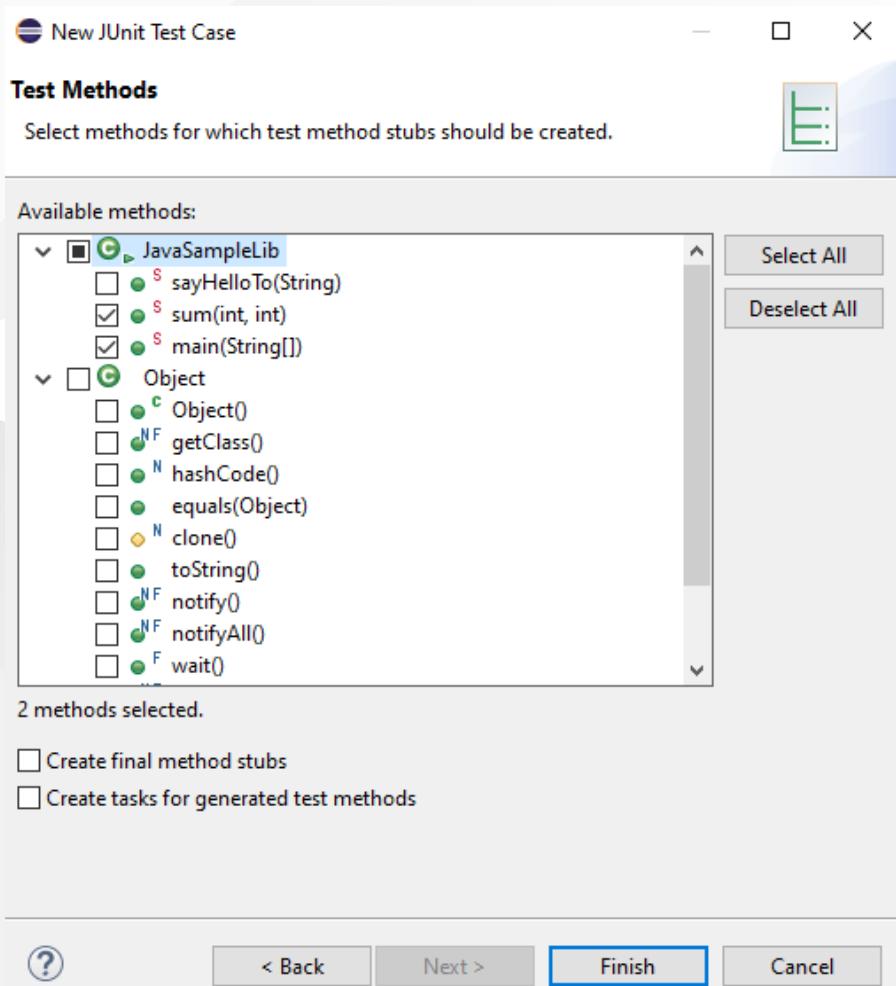
# Java Application + JUnit

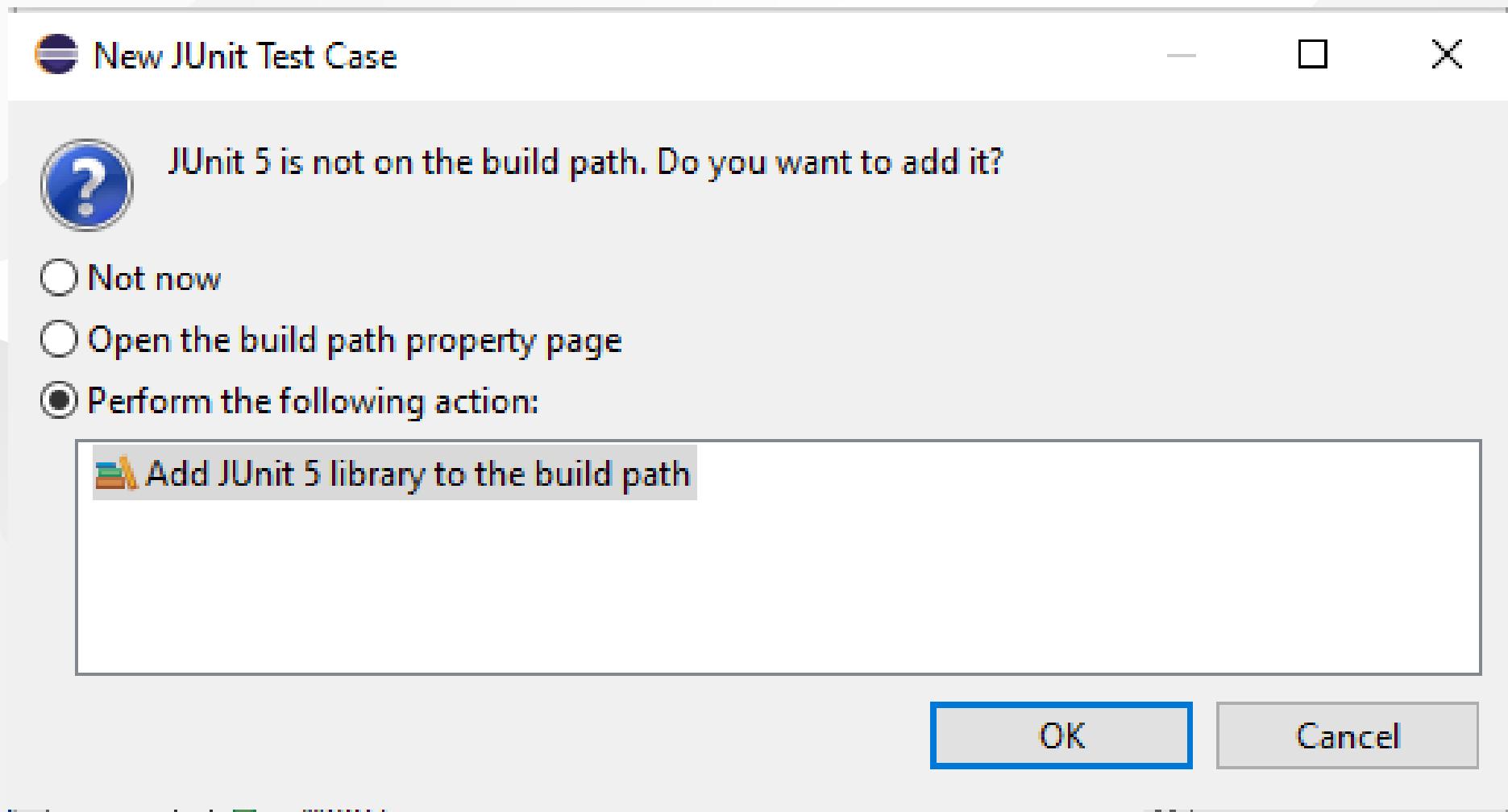
In normal java application we can right click the project java-sample-lib and add Junit case



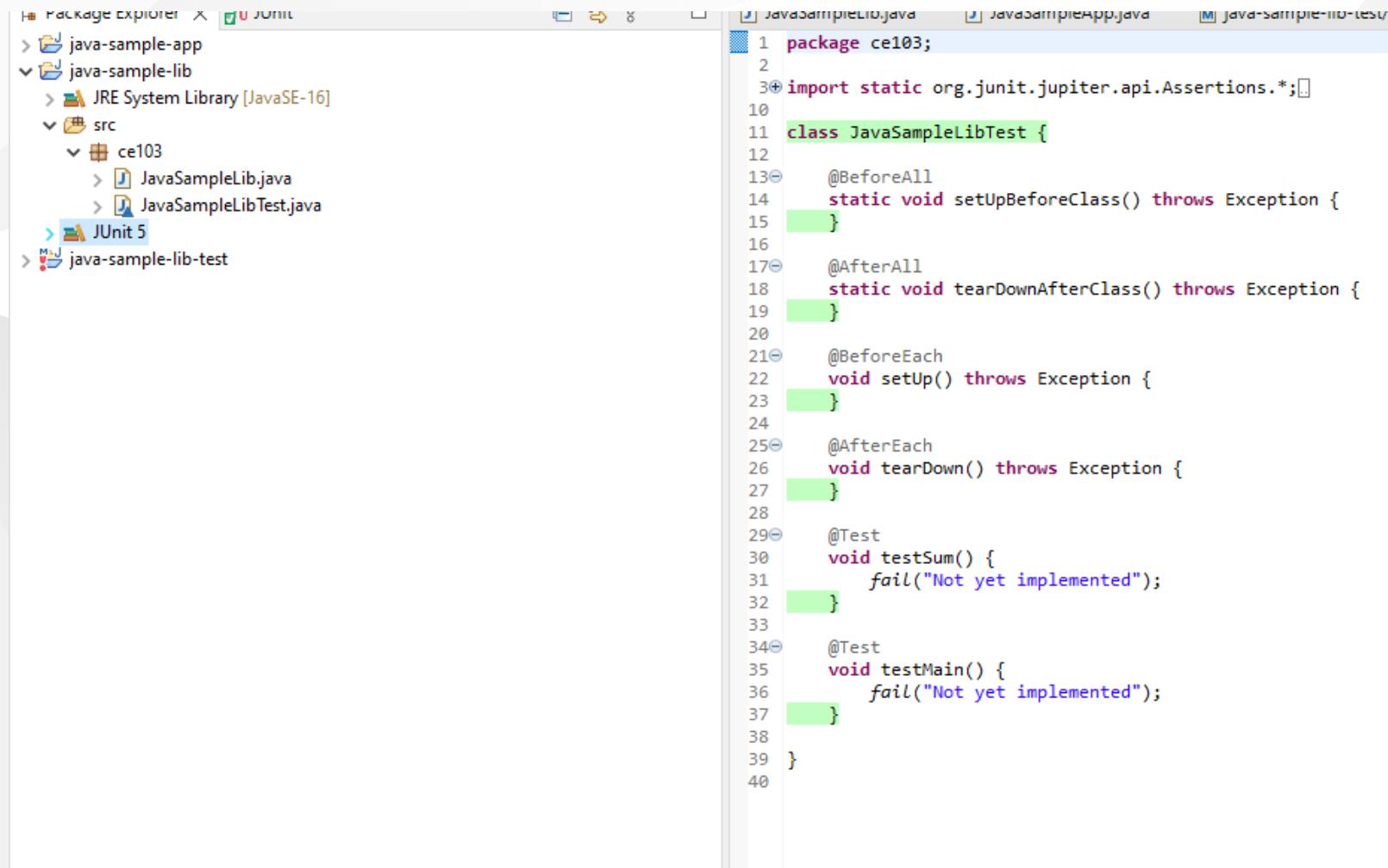








and you will have the following test class



The screenshot shows a Java development environment with the following structure:

- Package Explorer:** Shows the project tree:
  - java-sample-app
  - java-sample-lib
    - JRE System Library [JavaSE-16]
    - src
      - ce103
        - JavaSampleLib.java
        - JavaSampleLibTest.java
  - JUnit 5
  - java-sample-lib-test

The **JavaSampleLibTest.java** file is open in the editor, displaying the following code:

```
1 package ce103;
2
3 import static org.junit.jupiter.api.Assertions.*;
4
5 class JavaSampleLibTest {
6
7     @BeforeAll
8     static void setUpBeforeClass() throws Exception {
9
10    }
11
12     @AfterAll
13     static void tearDownAfterClass() throws Exception {
14
15    }
16
17     @BeforeEach
18     void setUp() throws Exception {
19
20    }
21
22     @AfterEach
23     void tearDown() throws Exception {
24
25    }
26
27     @Test
28     void testSum() {
29         fail("Not yet implemented");
30     }
31
32     @Test
33     void testMain() {
34         fail("Not yet implemented");
35     }
36
37 }
38
39 }
```

# We need to cover all code branches that we coded

I have updated JavaSampleLib.java as follow to check outputs

## JavaSampleLib.java

```
package ce103;

public class JavaSampleLib {

    public static String sayHelloTo(String name) {
        String output = "";
        if(!name.isBlank() && !name.isEmpty()){
            output = "Hello "+name;
        }else {
            output = "Hello There";
        }
        System.out.println(output);
        return output;
    }

    public static int sum(int a,int b)
    {
        int c = 0;
        c = a+b;
        return c;
    }

    public int multiply(int a, int b) {
        return a * b;
    }

    //  public static void main(String[] args) {
    //      // TODO Auto-generated method stub
    //      System.out.println("Hello World!");
    //  }

    //  JavaSampleLib.sayHelloTo("Computer");
    //  int result = JavaSampleLib.sum(5, 4);
    //  System.out.println("Results is" + result);
    //  System.out.printf("Results is %d \n", result);

    //  try {
    //      System.in.read();
    //  } catch (IOException e) {
    //      // TODO Auto-generated catch block
    //      e.printStackTrace();
    //  }

}
```



# and JavaSampleLibTest.java

```

package ce103;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.DisplayName;
import org.junit.jupiter.api.RepeatedTest;
import org.junit.jupiter.api.Test;
import org.junit.jupiter.params.ParameterizedTest;
import org.junit.jupiter.params.provider.MethodSource;

class JavaSampleLibTest {
    JavaSampleLib sampleLib;

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
    }

    @BeforeEach
    void setup() throws Exception {
        sampleLib = new JavaSampleLib();
    }

    @AfterEach
    void tearDown() throws Exception {
    }

    @Test
    @DisplayName("Simple Say Hello should work")
    void testSayHelloTo() {
        assertEquals("Hello Computer", JavaSampleLib.sayHelloTo("Computer"), "Regular say hello should work");
    }

    @Test
    @DisplayName("Simple Say Hello shouldn't work")
    void testSayHelloToWrong() {
        assertEquals("Hello All", JavaSampleLib.sayHelloTo("Computer"), "Regular say hello won't work");
    }

    @Test
    @DisplayName("Simple sum should work")
    void testSumCorrect() {
        assertEquals(9, JavaSampleLib.sum(4, 5), "Regular sum should work");
    }

    @Test
    @DisplayName("Simple sum shouldn't work")
    void testSumWrong() {
        assertEquals(10, JavaSampleLib.sum(4, 5), "Regular sum shouldn't work");
    }

    @Test
    @DisplayName("Simple multiplication should work")
    void testMultiply() {
        assertEquals(28, sampleLib.multiply(4, 5), "Regular multiplication should work");
    }

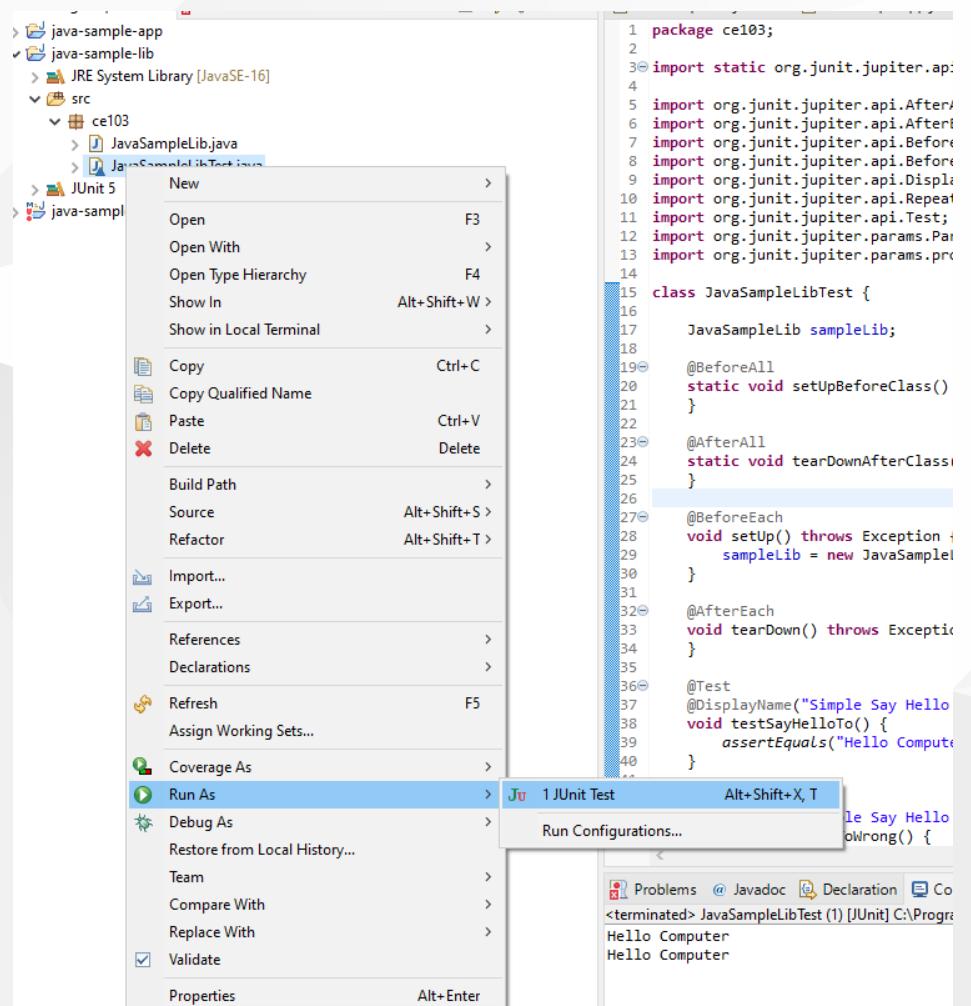
    @RepeatedTest(5)
    @DisplayName("Ensure correct handling of zero")
    void testMultiplyWithZero() {
        assertEquals(0, sampleLib.multiply(0, 5), "Multiple with zero should be zero");
        assertEquals(0, sampleLib.multiply(5, 0), "Multiple with zero should be zero");
    }

    public static int[][] data() {
        return new int[][] { { 1, 2, 2 }, { 5, 3, 15 }, { 121, 4, 484 }, { 2, 2, 2 } };
    }

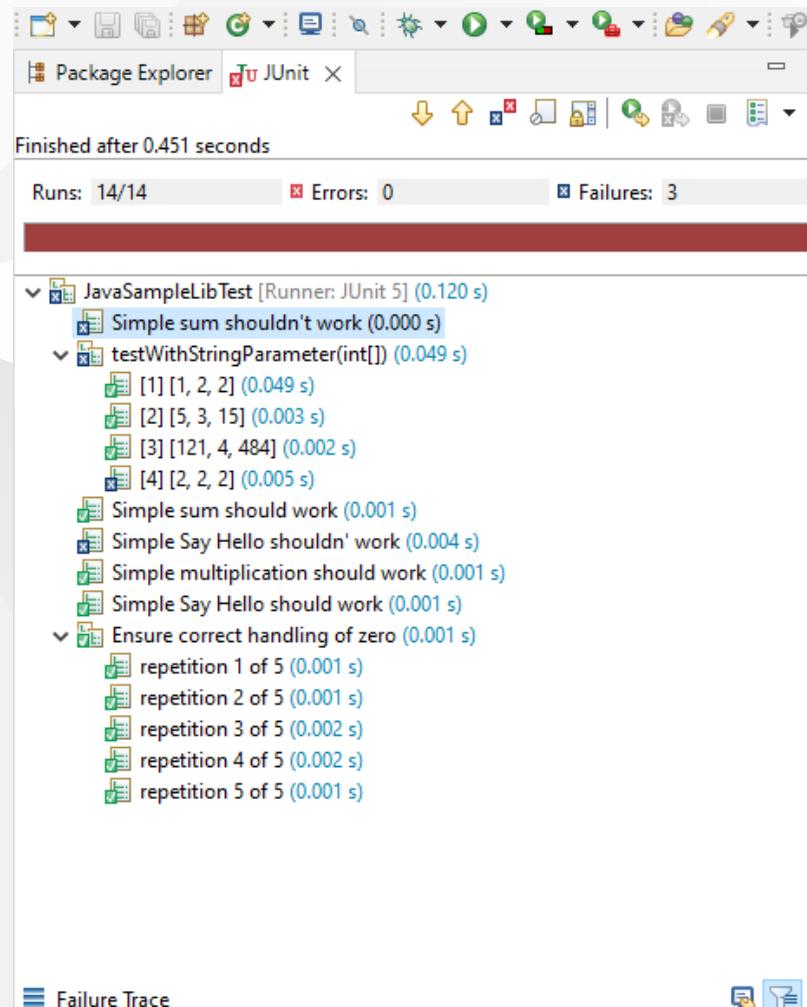
    @ParameterizedTest
    @MethodSource(value = "data")
    void testWithStringParameter(int[] data) {
        JavaSampleLib tester = new JavaSampleLib();
        int m1 = data[0];
        int m2 = data[1];
        int expected = data[2];
        assertEquals(expected, tester.multiply(m1, m2));
    }
}

```

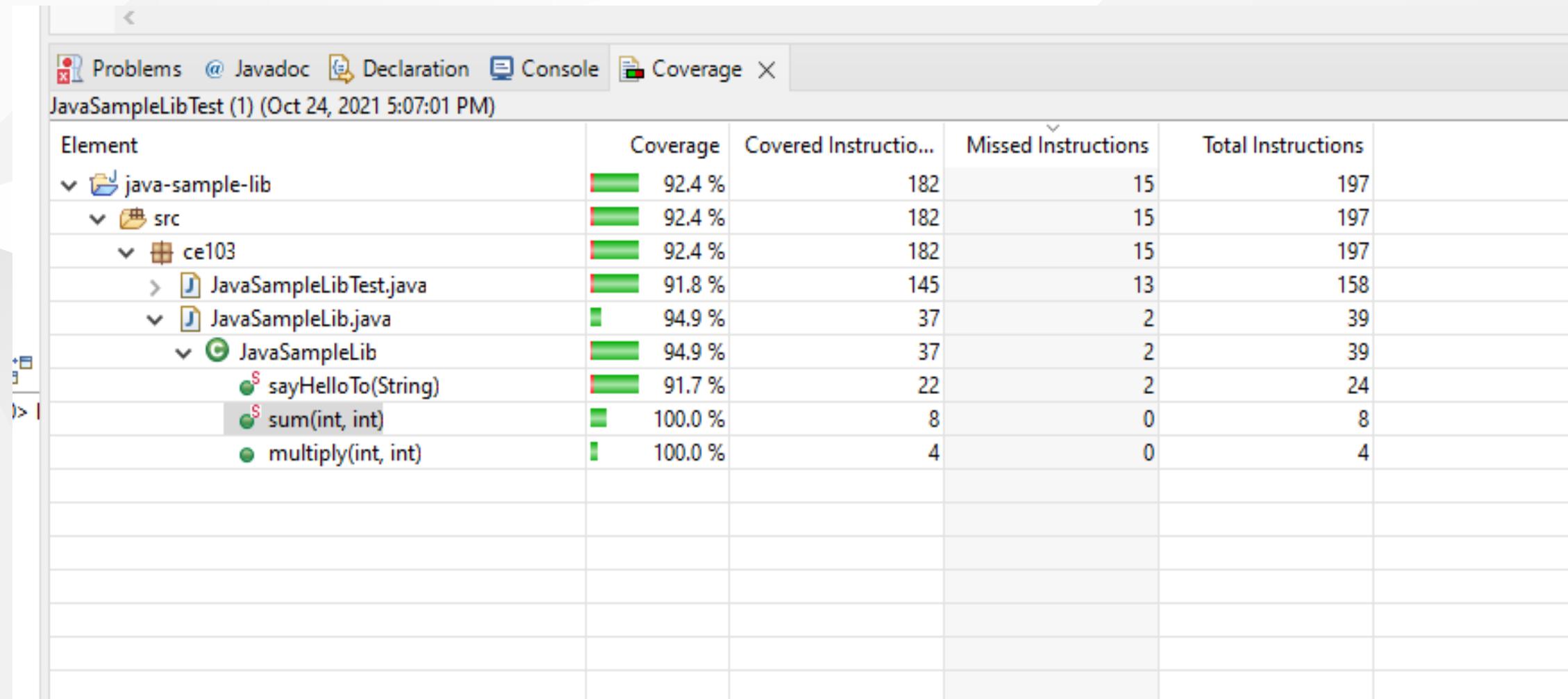
## if we run tests



we will see all results there



also we can see the code coverage of tests



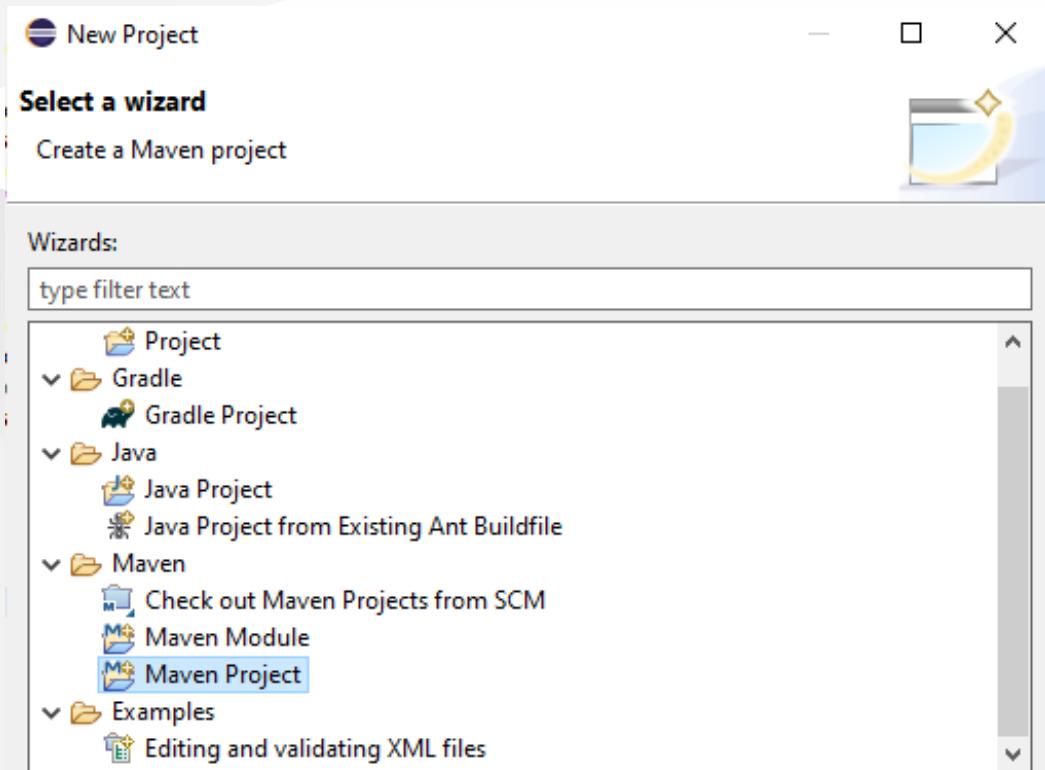
when we open our source code (just close and open again another case highlighting will not work) you will see tested part of your codes

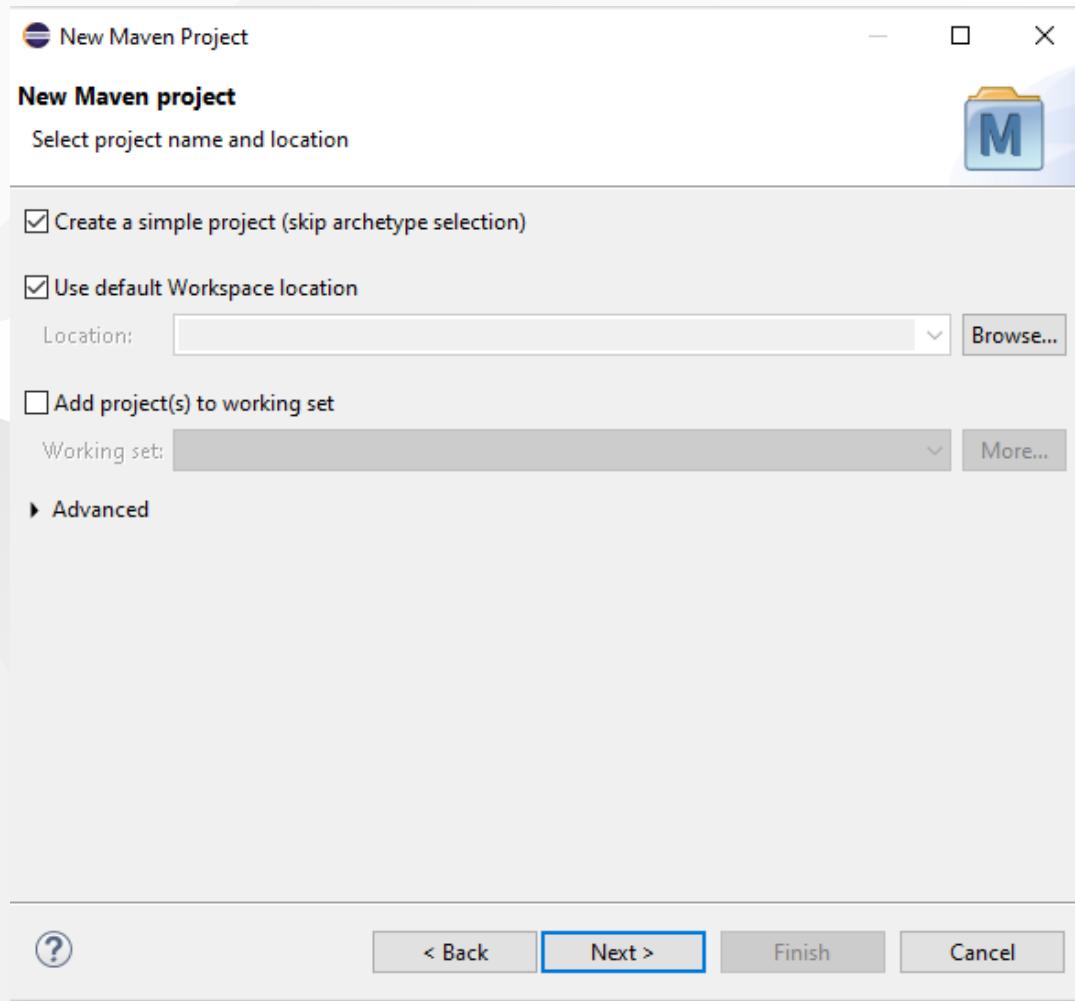
```
1 package ce103;
2
3 public class JavaSampleLib {
4
5     public static String sayHelloTo(String name) {
6
7         String output = "";
8
9         if(!name.isBlank() && !name.isEmpty()){
10             output = "Hello "+name;
11         }else {
12             output = "Hello There";
13         }
14
15         System.out.println(output);
16
17         return output;
18     }
19
20     public static int sum(int a,int b)
21     {
22         int c = 0;
23         c = a+b;
24         return c;
25     }
26
27     public int multiply(int a, int b) {
28         return a * b;
29     }
30 }
```

Lets create Maven project with tests

Create a maven project

*File -> New -> Maven Project*





Lets convert our sample java-sample-lib directories to standard folder structure for test and app division

## [Maven – Introduction to the Standard Directory Layout](#)

Also for intro you can use this

## [JUnit Hello World Example - Examples Java Code Geeks - 2021](#)

Eclipse

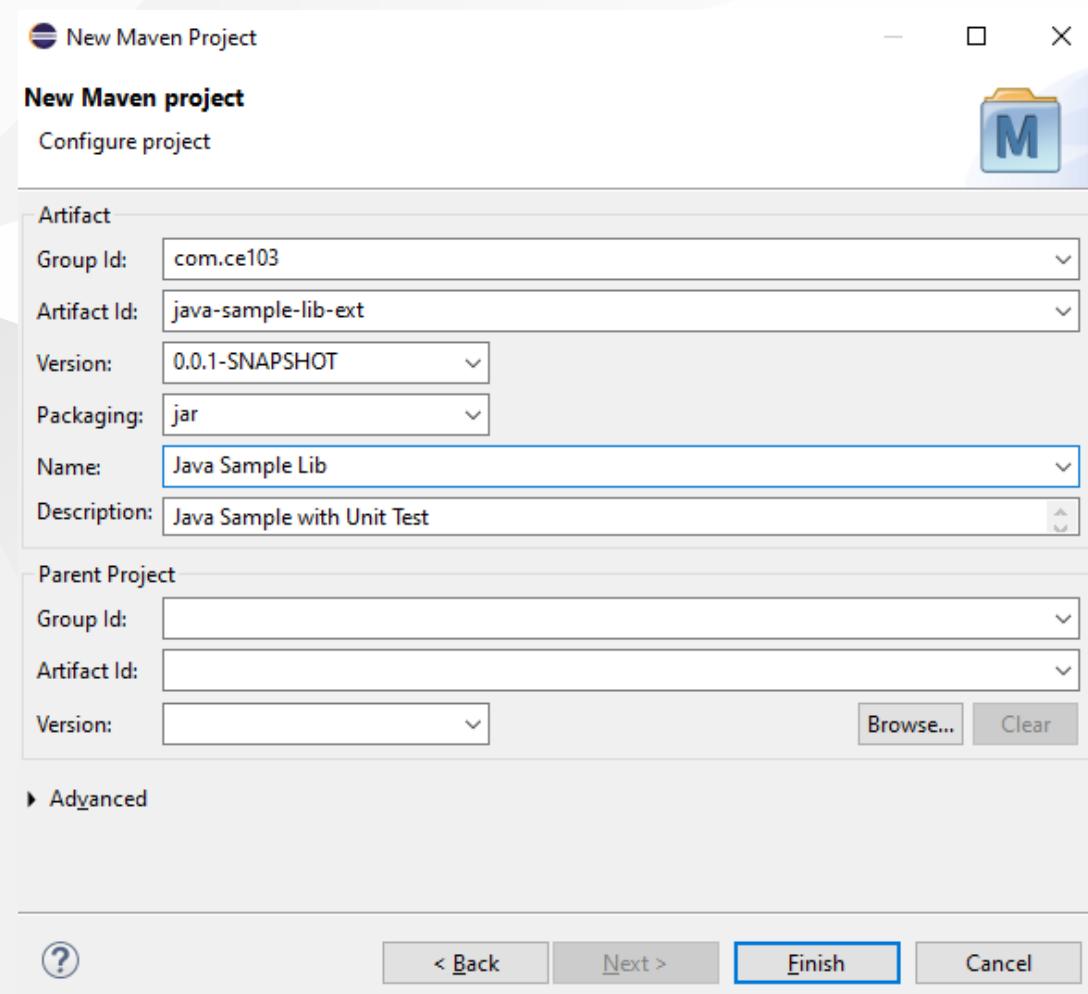
Maven

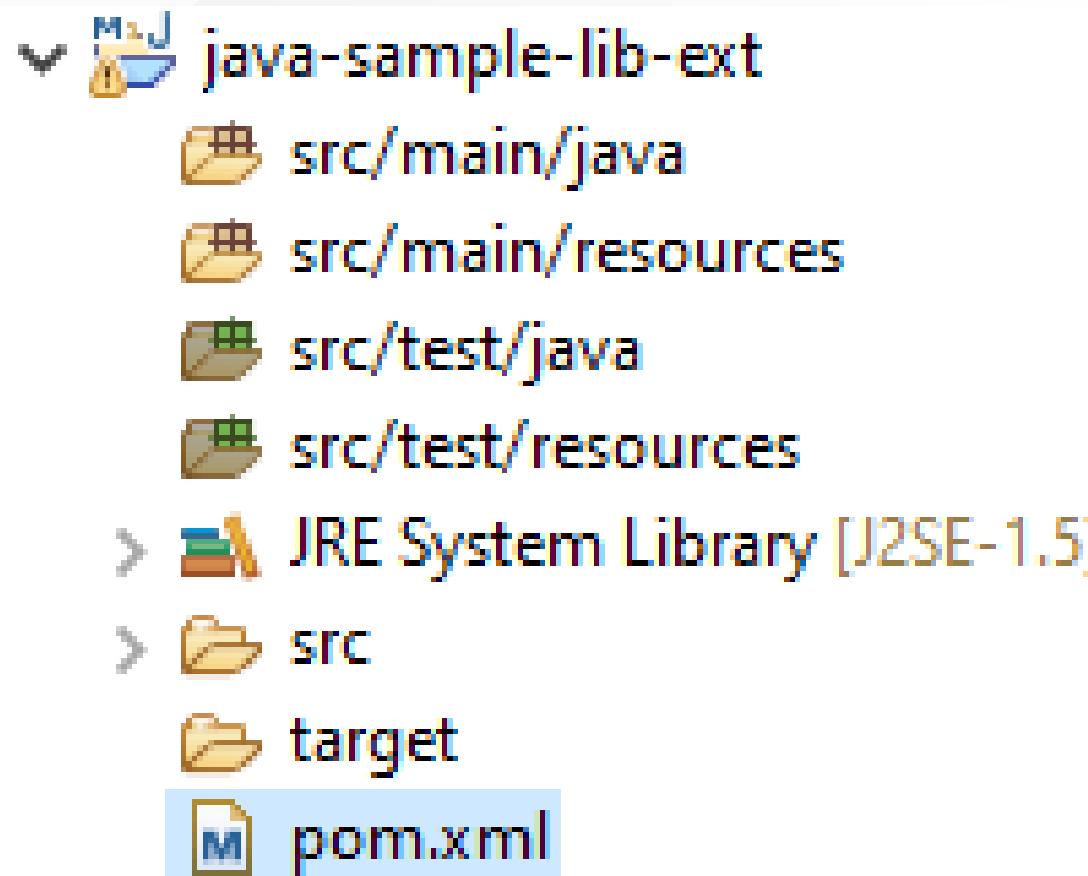
Java

JUnit 4.12 (pulled by Maven automatically)



Lets give new sample java-sample-lib-mvnbut in this time we will create a maven project





## pom.xml file

```
<project xmlns="http://maven.apache.org/POM/4.0.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
  <modelVersion>4.0.0</modelVersion>
  <groupId>com.ce103</groupId>
  <artifactId>java-sample-lib-ext</artifactId>
  <version>0.0.1-SNAPSHOT</version>
  <name>Java Sample Lib</name>
  <description>Java Sample with Unit Test</description>
</project>
```

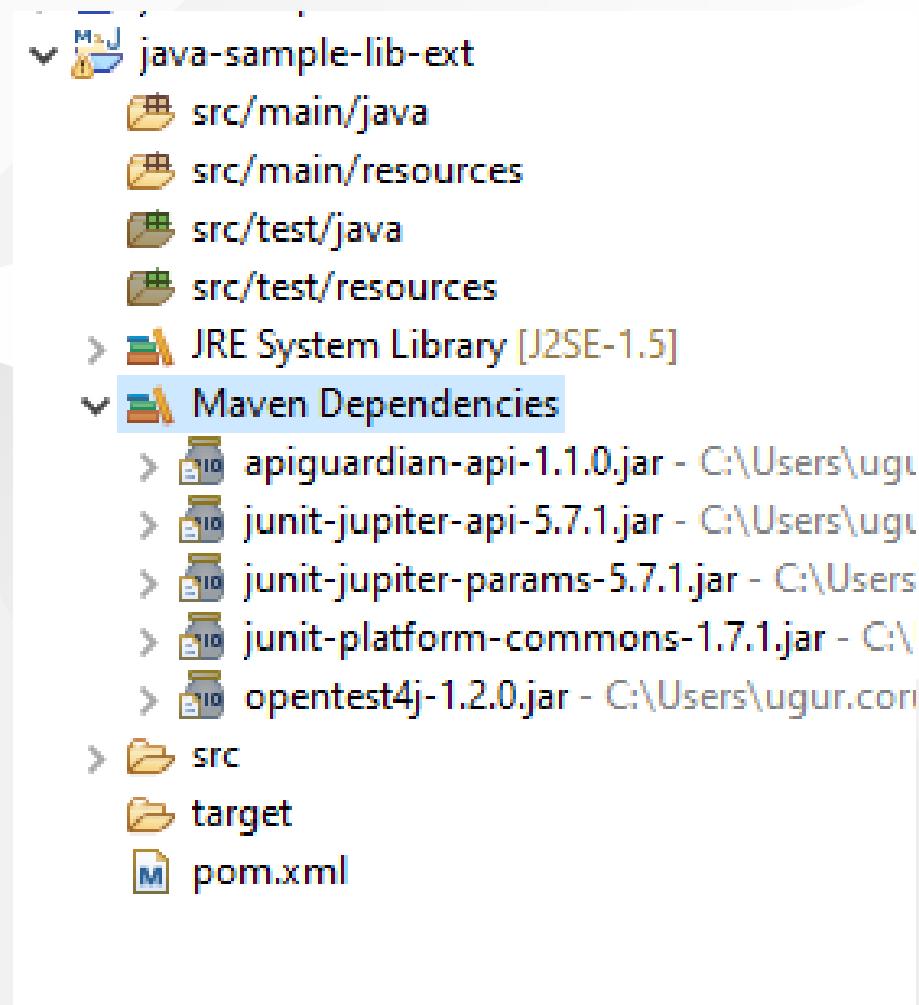
## we will add JUnit 5 for our project

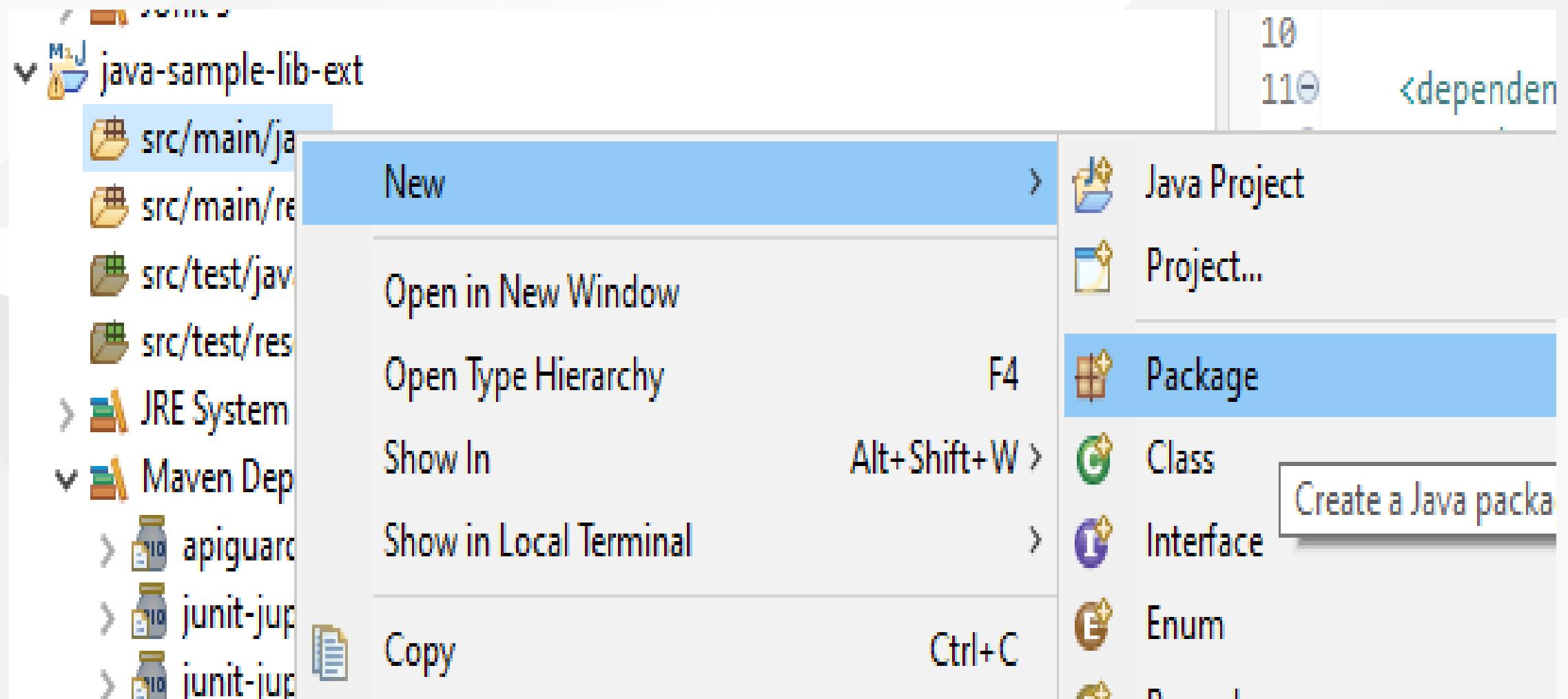
```
<project xmlns="http://maven.apache.org/POM/4.0.0"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 https://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>
    <groupId>com.ce103</groupId>
    <artifactId>java-sample-lib-ext</artifactId>
    <version>0.0.1-SNAPSHOT</version>
    <name>Java Sample Lib</name>
    <description>Java Sample with Unit Test</description>

    <dependencies>
        <dependency>
            <groupId>org.junit.jupiter</groupId>
            <artifactId>junit-jupiter-params</artifactId>
            <version>5.7.1</version>
            <scope>test</scope>
        </dependency>
    </dependencies>

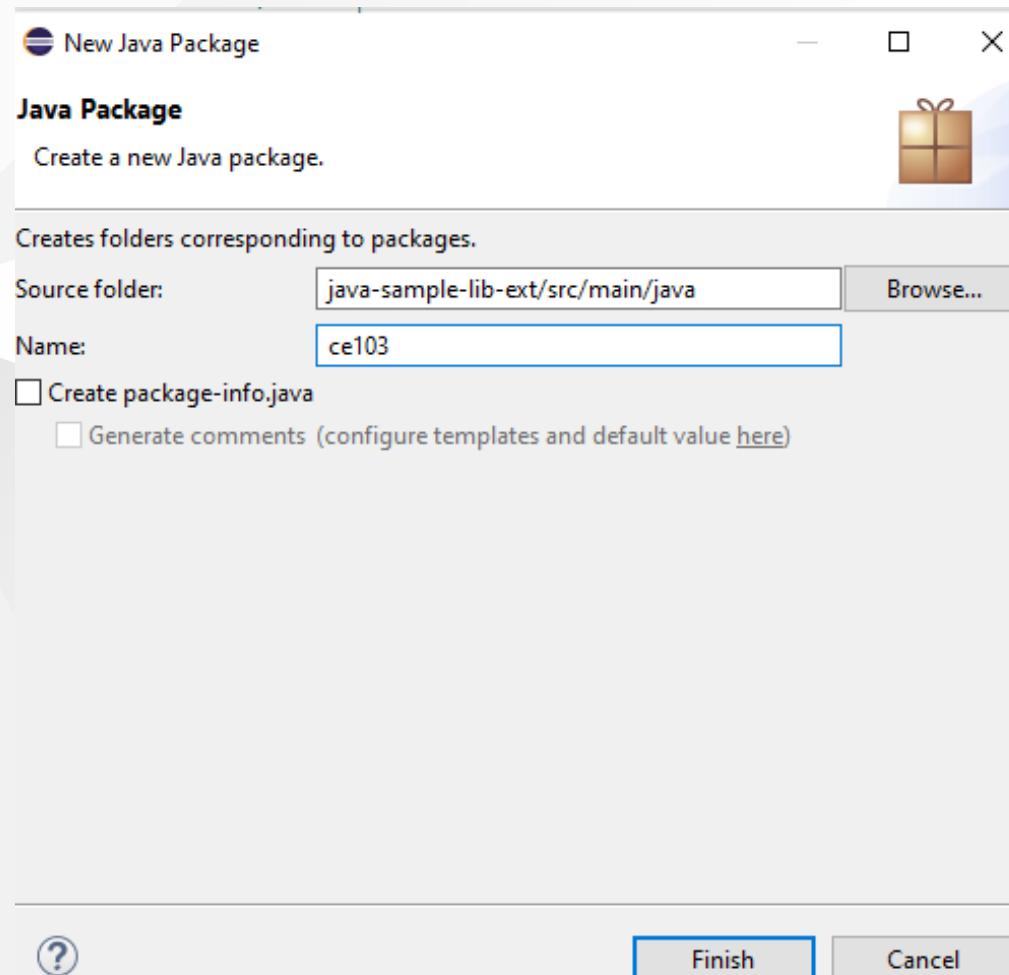
</project>
```

it will automatically download libraries

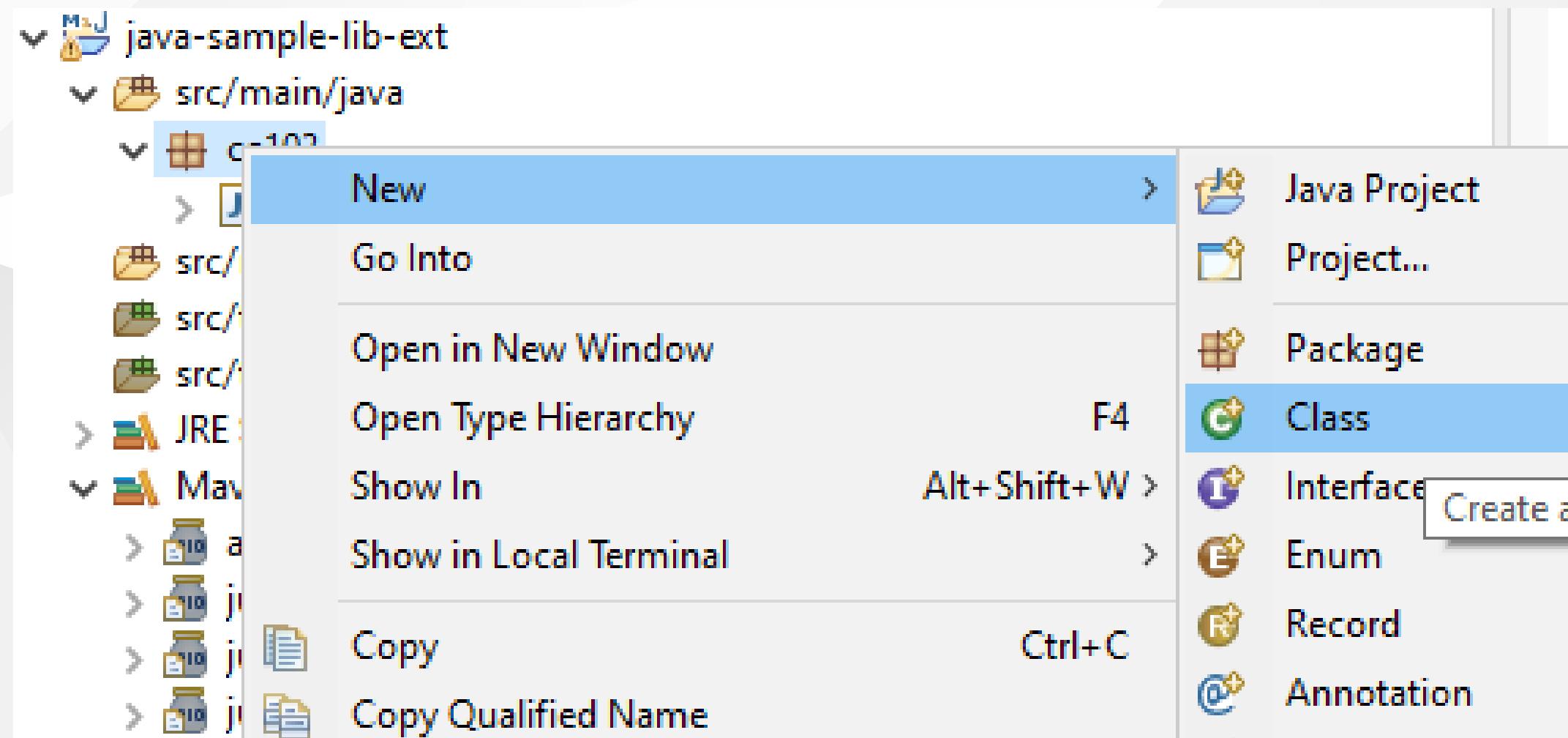


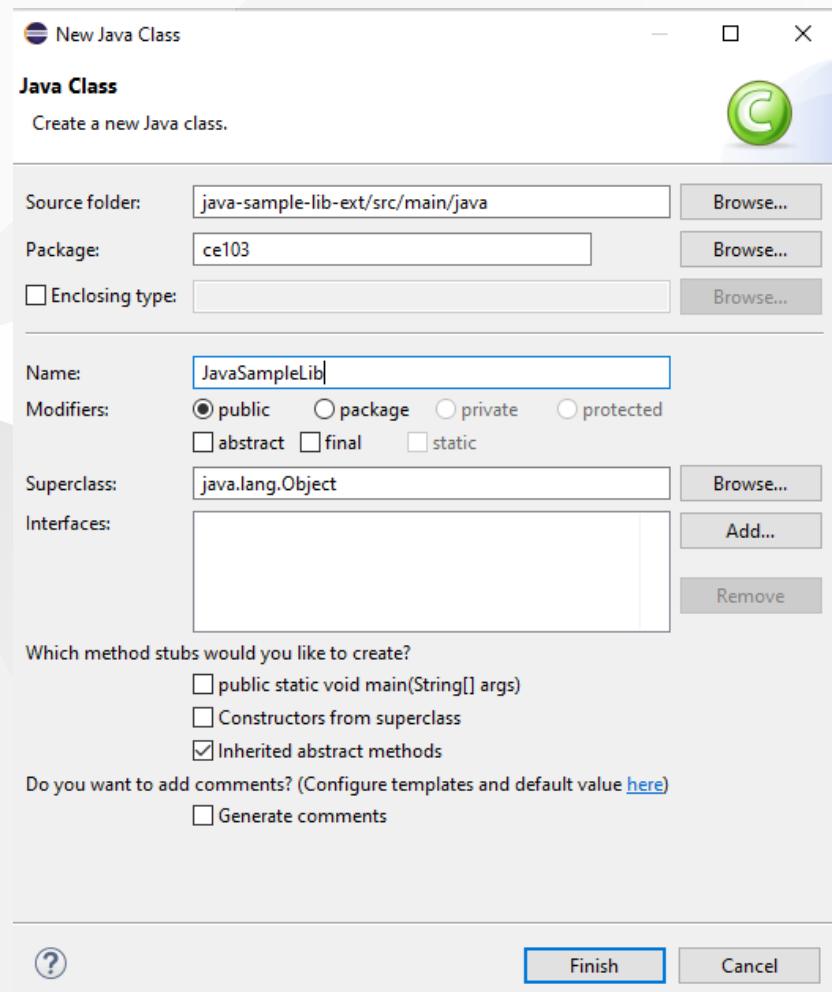


## Create java sample library in ce103 package, first create java package



In this package create library class





# copy content from other library

```
package ce103;

public class JavaSampleLib {

    public static String sayHelloTo(String name) {

        String output = "";

        if(!name.isBlank() && !name.isEmpty()){
            output = "Hello "+name;
        }else {
            output = "Hello There";
        }

        System.out.println(output);

        return output;
    }

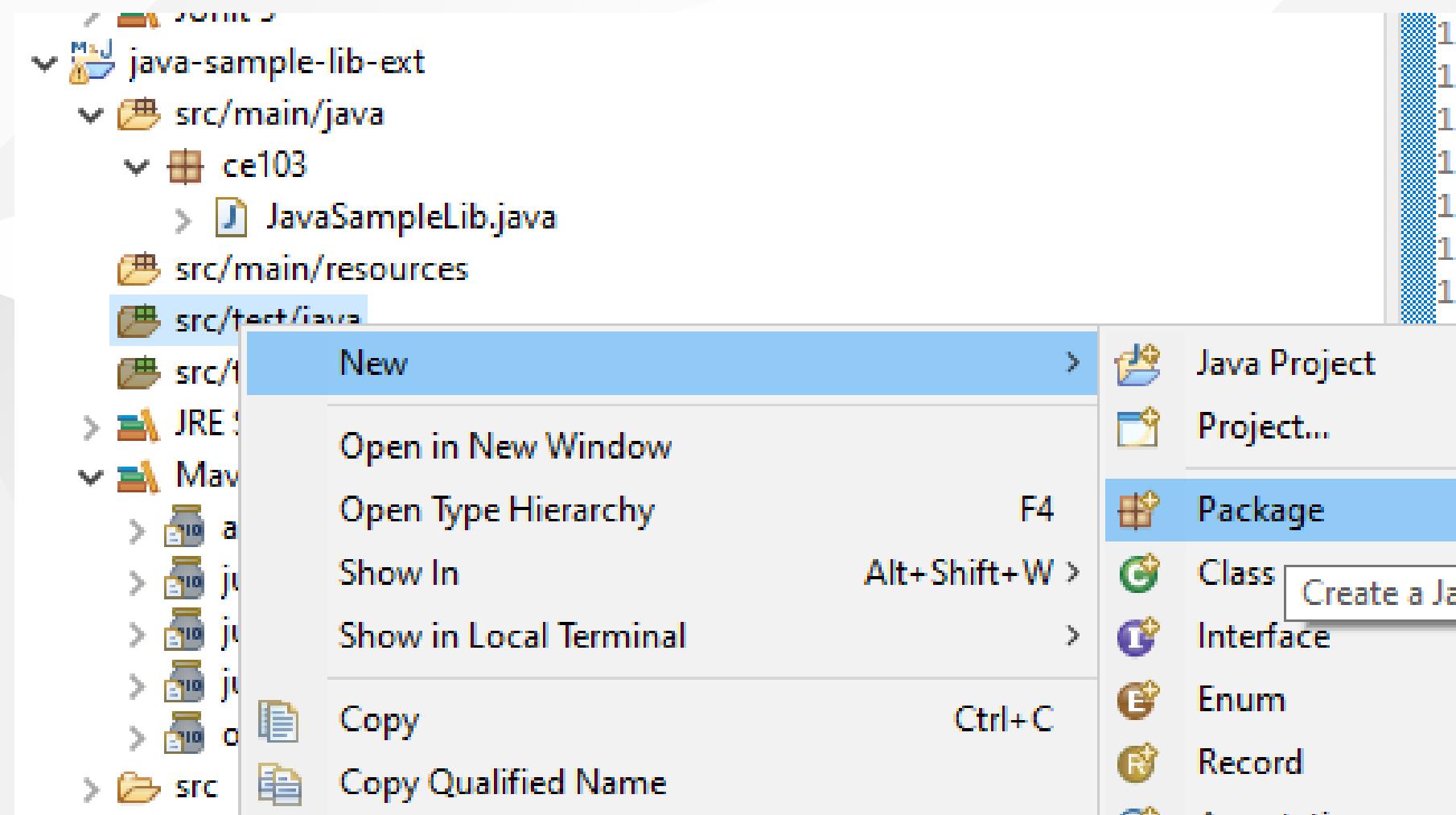
    public static int sum(int a,int b)
    {
        int c = 0;
        c = a+b;
        return c;
    }

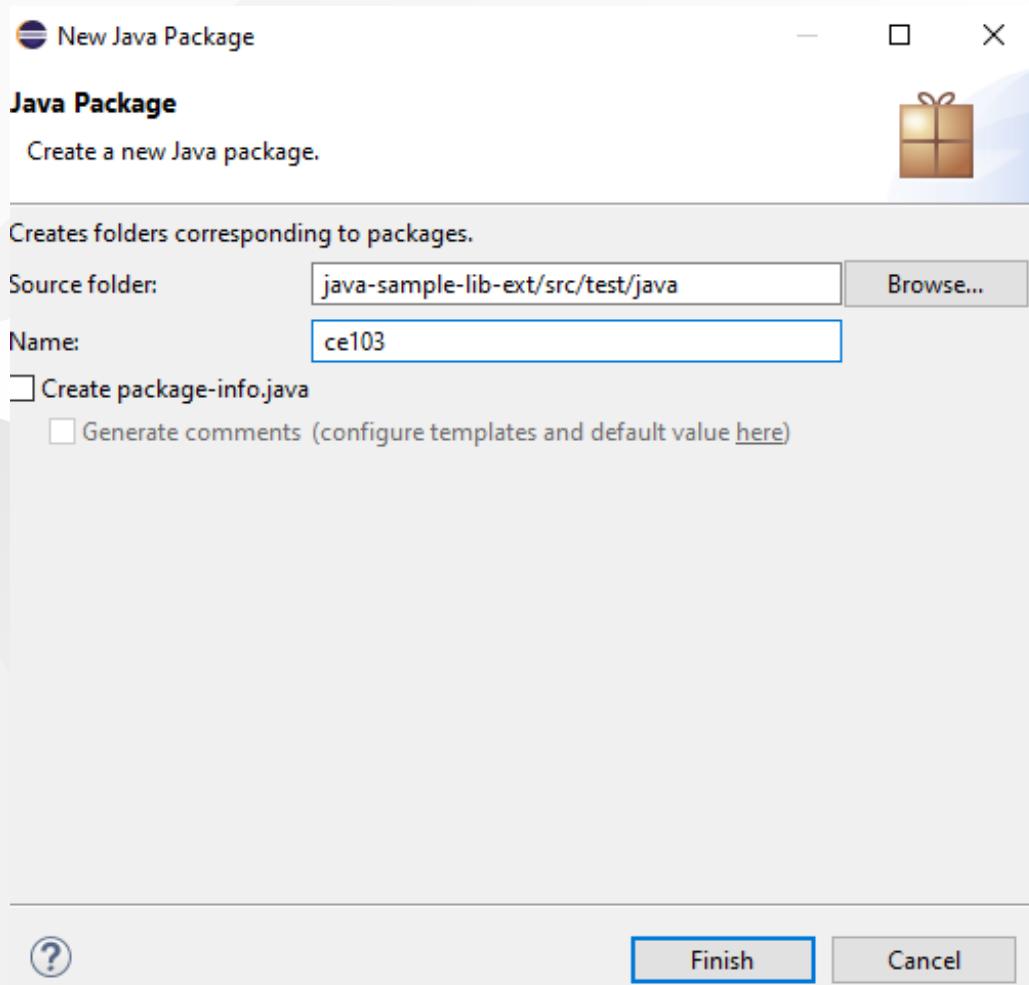
    public int multiply(int a, int b) {
        return a * b;
    }

}
```

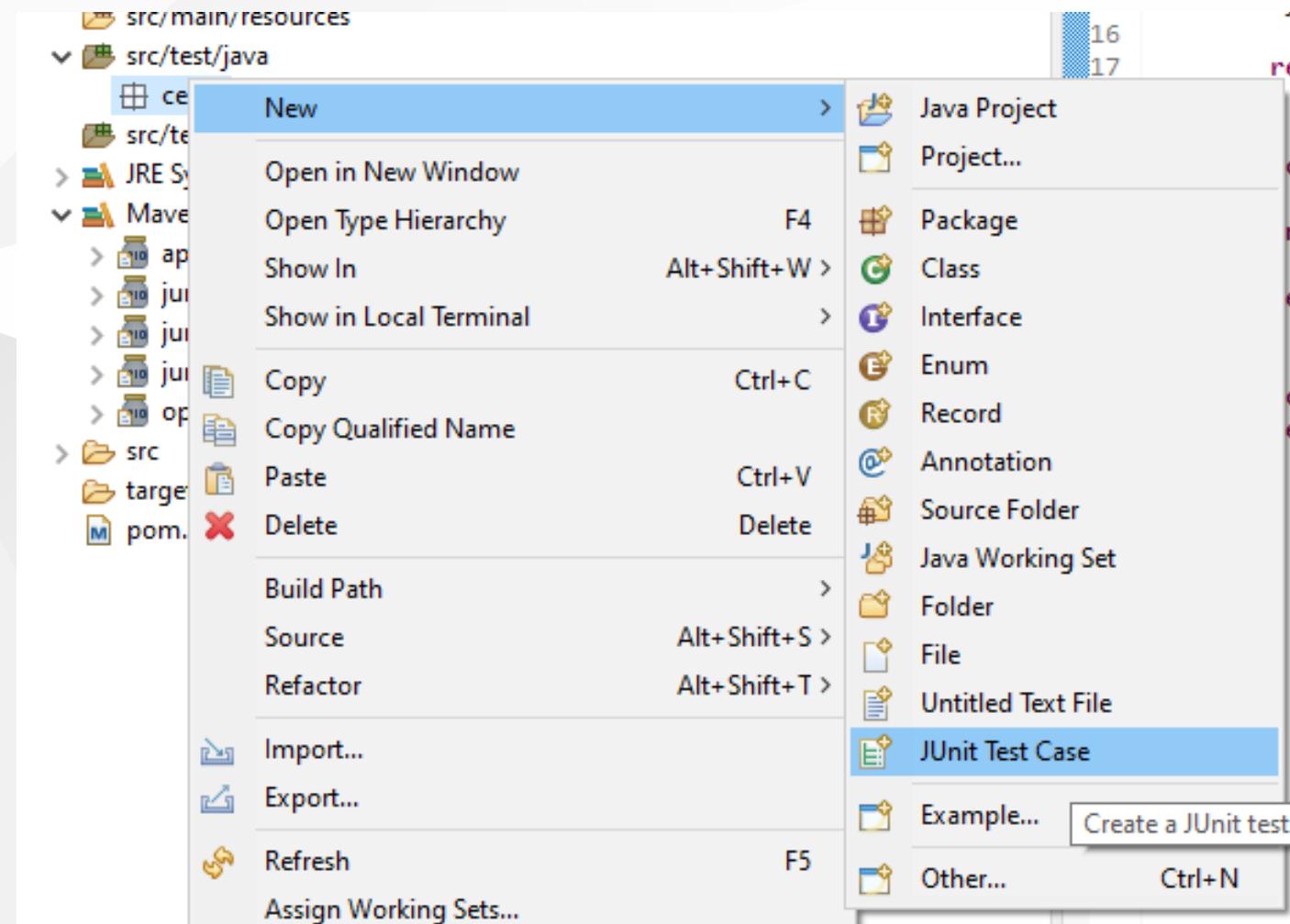


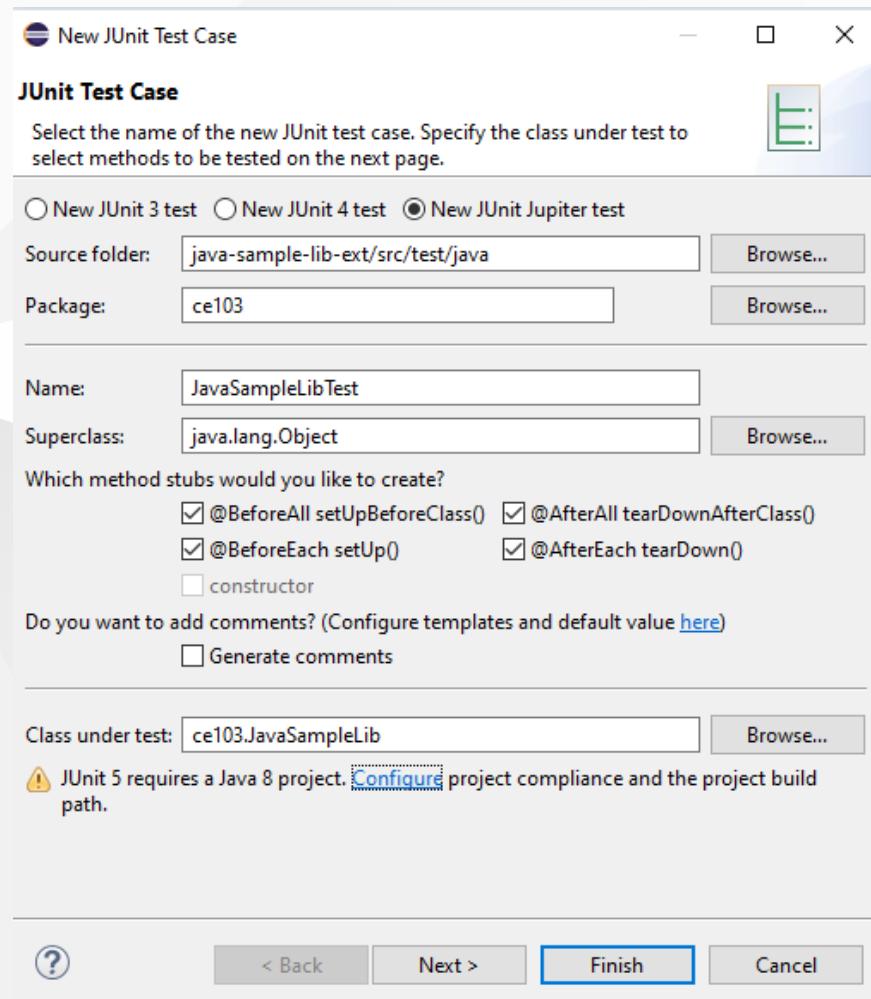
Now lets create tests inf src/test/java

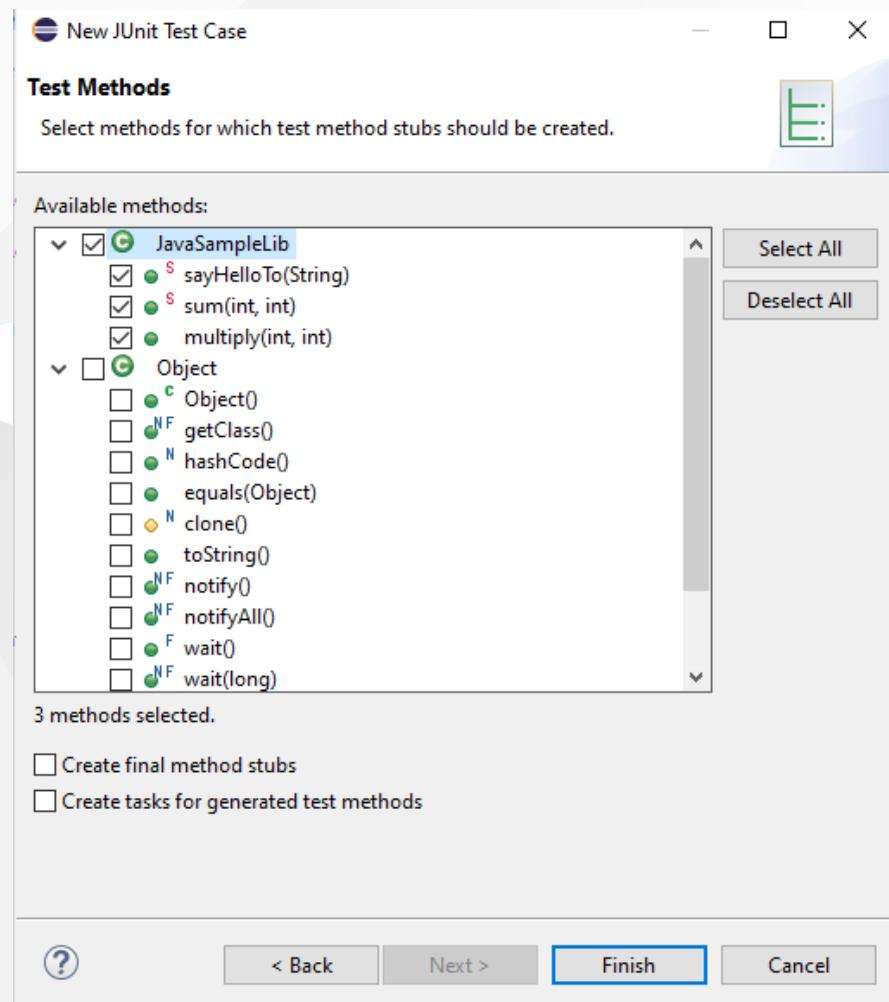


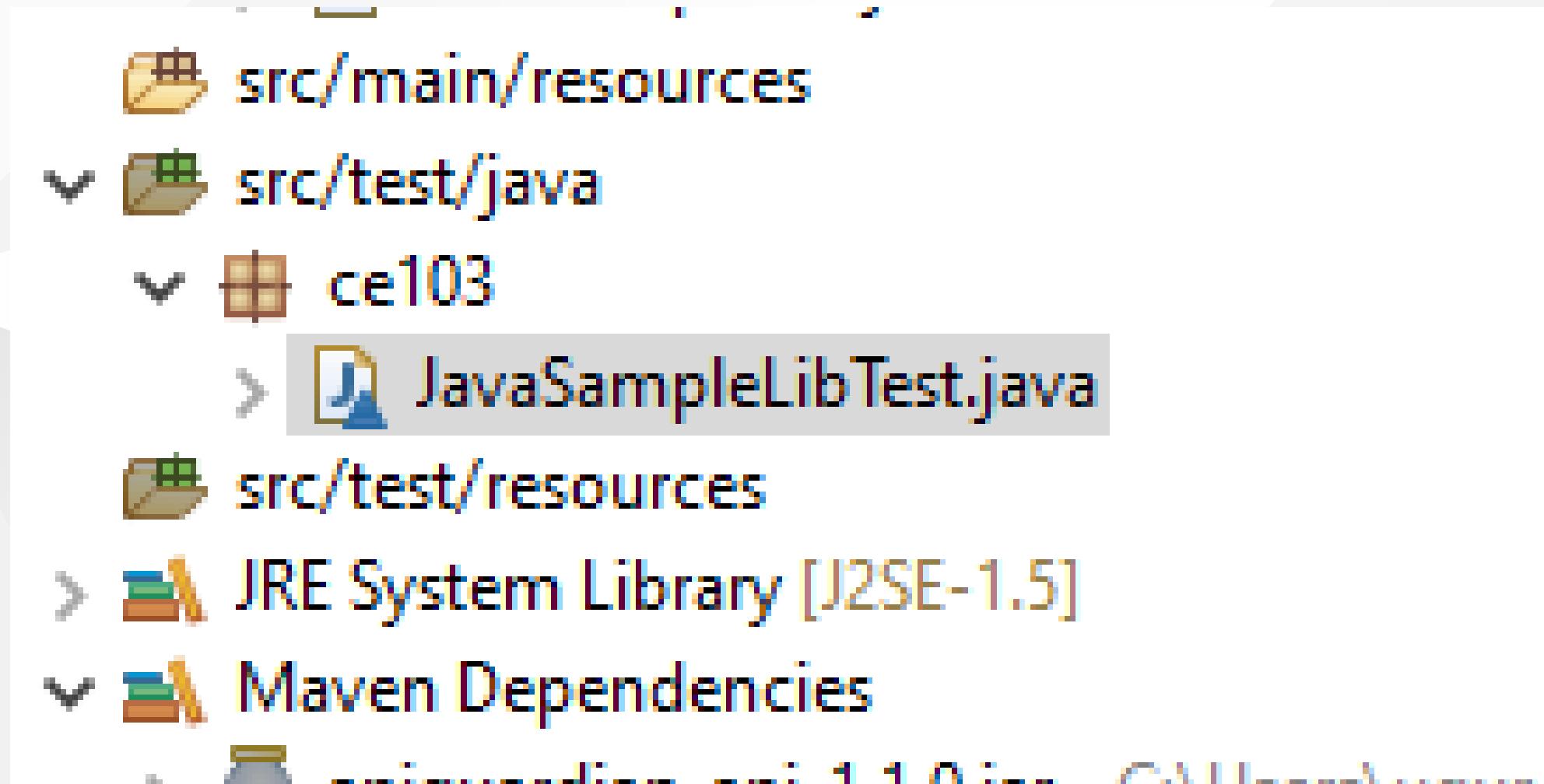


## create a JUnit Case









# you will simple template

```
package ce103;

import static org.junit.jupiter.api.Assertions.*;

import org.junit.jupiter.api.AfterAll;
import org.junit.jupiter.api.AfterEach;
import org.junit.jupiter.api.BeforeAll;
import org.junit.jupiter.api.BeforeEach;
import org.junit.jupiter.api.Test;

class JavaSampleLibTest {

    @BeforeAll
    static void setUpBeforeClass() throws Exception {
    }

    @AfterAll
    static void tearDownAfterClass() throws Exception {
    }

    @BeforeEach
    void setUp() throws Exception {
    }

    @AfterEach
    void tearDown() throws Exception {
    }

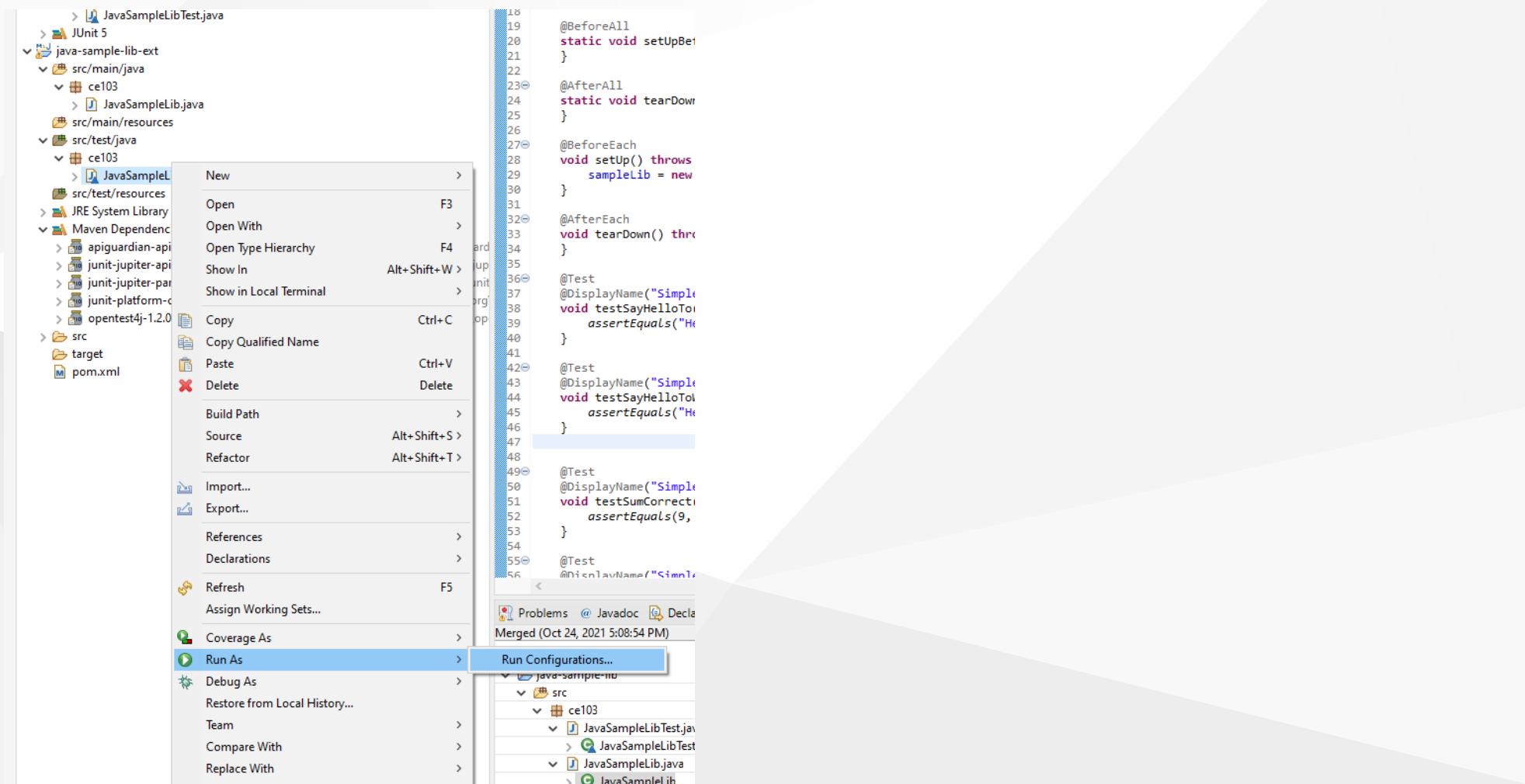
    @Test
    void testSayHelloTo() {
        fail("Not yet implemented");
    }

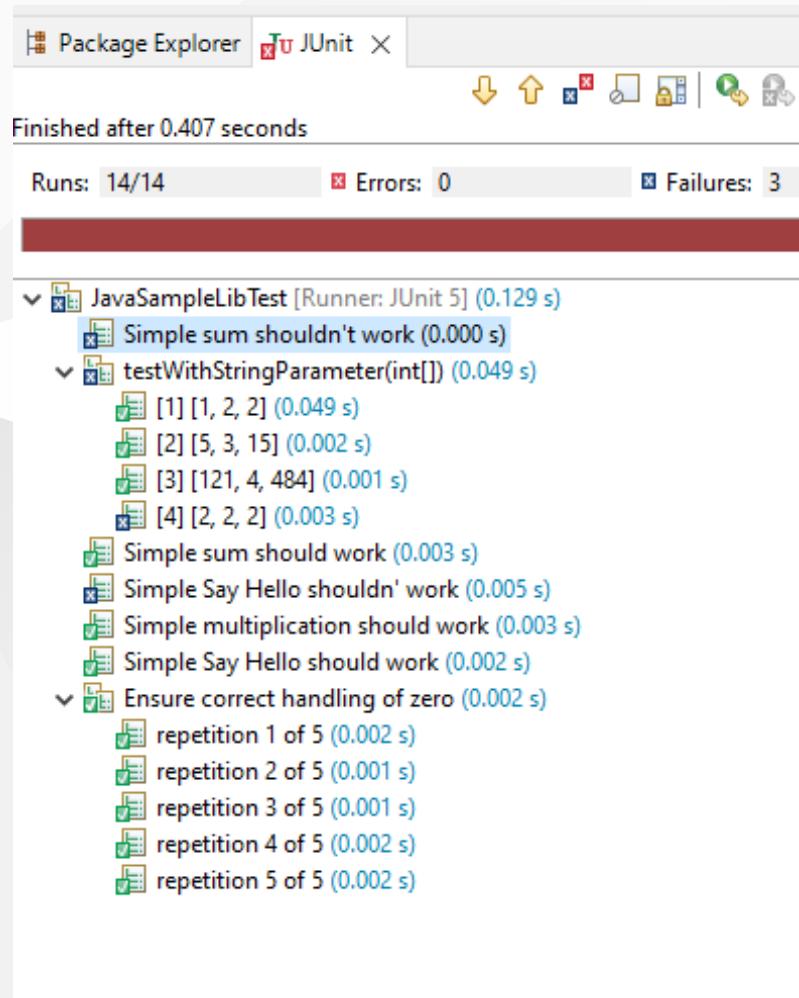
    @Test
    void testSum() {
        fail("Not yet implemented");
    }

    @Test
    void testMultiply() {
        fail("Not yet implemented");
    }
}
```



now lets copy tests from other projects





The screenshot shows the Eclipse IDE interface with the following components:

- Top Bar:** eclipse-workspace - java-sample-lib-ext/src/main/java/ce103/JavaSampleLib.java - eclipse IDE. Includes File, Edit, Source, Refactor, Navigate, Search, Project, Run, Window, Help.
- Left Sidebar:** Package Explorer (JavaSampleLibTest [Runner: JUnit 5] (0.278 s)) showing test results: Runs: 14/14, Errors: 0, Failures: 3. It lists several test cases like Simple sum shouldn't work, testWithStringParameter(int[]), Simple sum should work, etc.
- Central Area:** Java code editor for JavaSampleLib.java. The code defines a class JavaSampleLib with methods sayHelloTo, sum, and multiply.
- Bottom Left:** Failure Trace showing an assertion error: org.opentest4j.AssertionFailedError: Regular sum shouldn't work ==> expected: <10> | at ce103.JavaSampleLibTest.testSumWrong(JavaSampleLibTest.java:58).
- Bottom Right:** Coverage analysis table for JavaSampleLibTestExt (Oct 24, 2021 5:39:34 PM). The table details instruction coverage across various source files and methods.

Element	Coverage	Covered Instructions	Missed Instructions	Total Instructions
java-sample-lib-ext	92.4 %	182	15	197
src/test/java	91.8 %	145	13	158
ce103	91.8 %	145	13	158
src/main/java	94.9 %	37	2	39
ce103	94.9 %	37	2	39
JavaSampleLib.java	94.9 %	37	2	39

That's a part of java unit testing...



# TDD (Test Driven Development)

# Test and Deployment Automation Management

Travis-CI + C



## Travis-CI + Cpp



## Travis-CI + C#



## Travis-CI + Java



# References

[GitHub - MicrosoftDocs/cpp-docs: C++ Documentation](#)