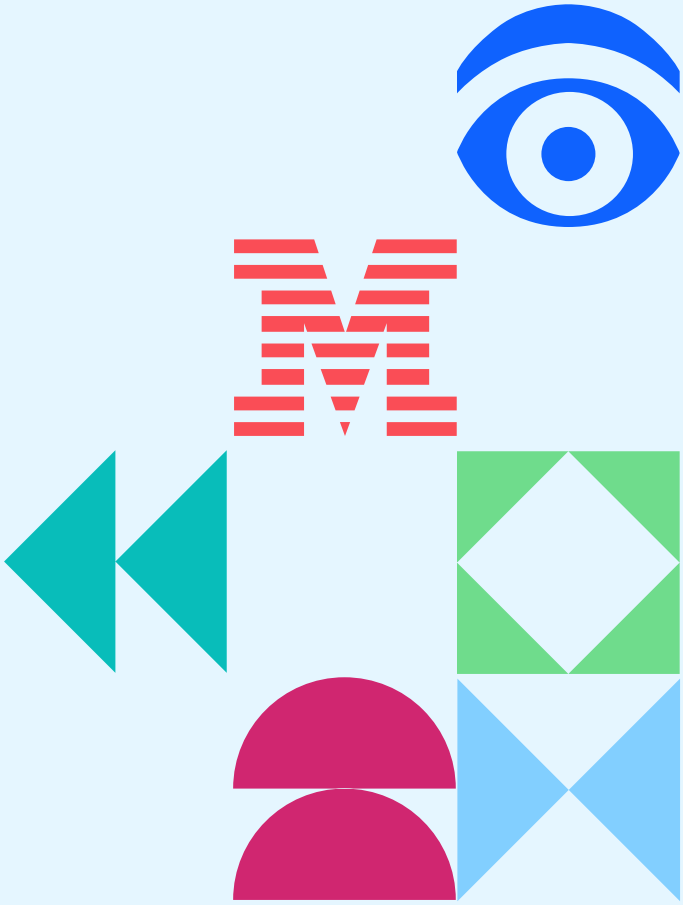
Using IBM Watsonx Code Assistant for Z for Application Discovery & COBOL Refactoring

Hands-on lab guide



David Hawreluk

dhawrel@us.ibm.com

Brand Technical Specialist - Z DevOps, ADDI

Frank Hernandez

Frank.Hernandez1@ibm.com

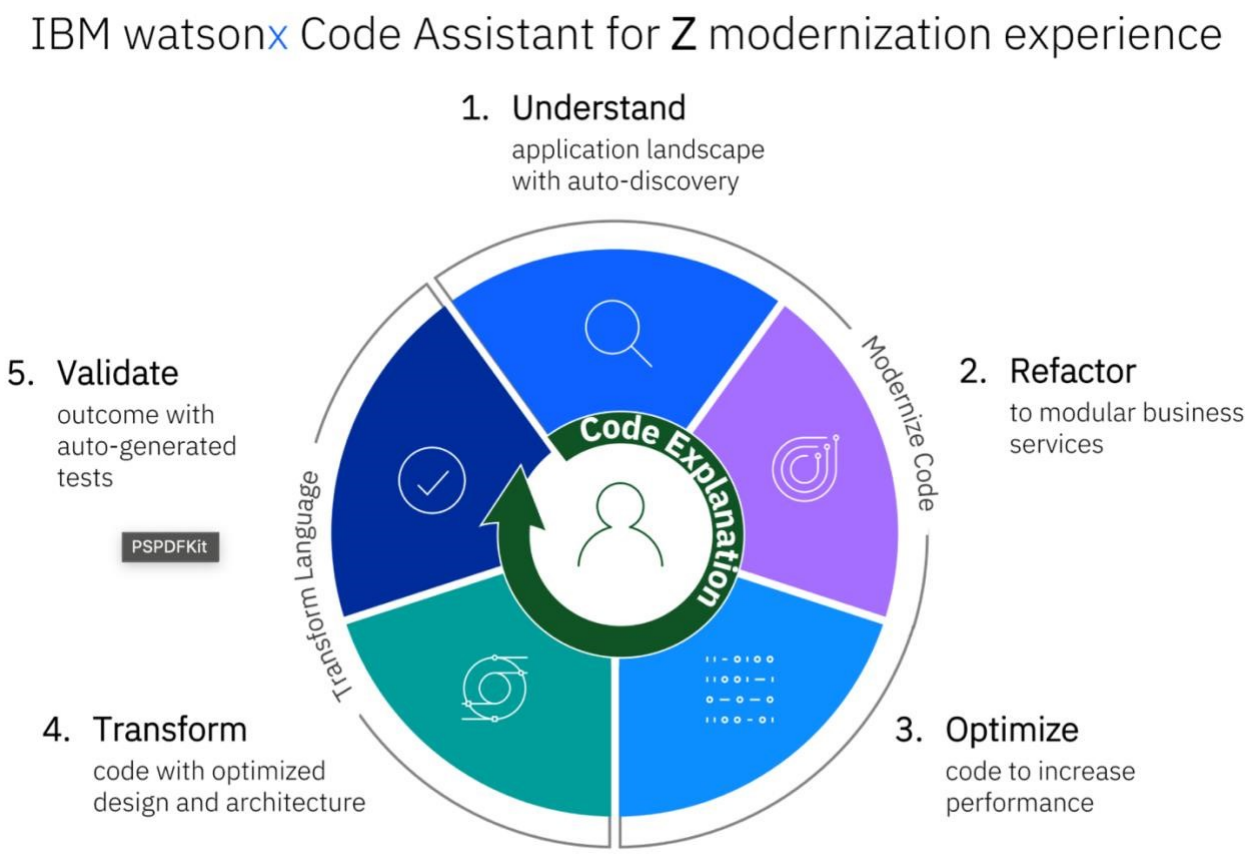
Brand Technical Specialist – Z AIOPS, WCA4Z

IBM

TABLE OF CONTENTS

2024 IBM Corporation

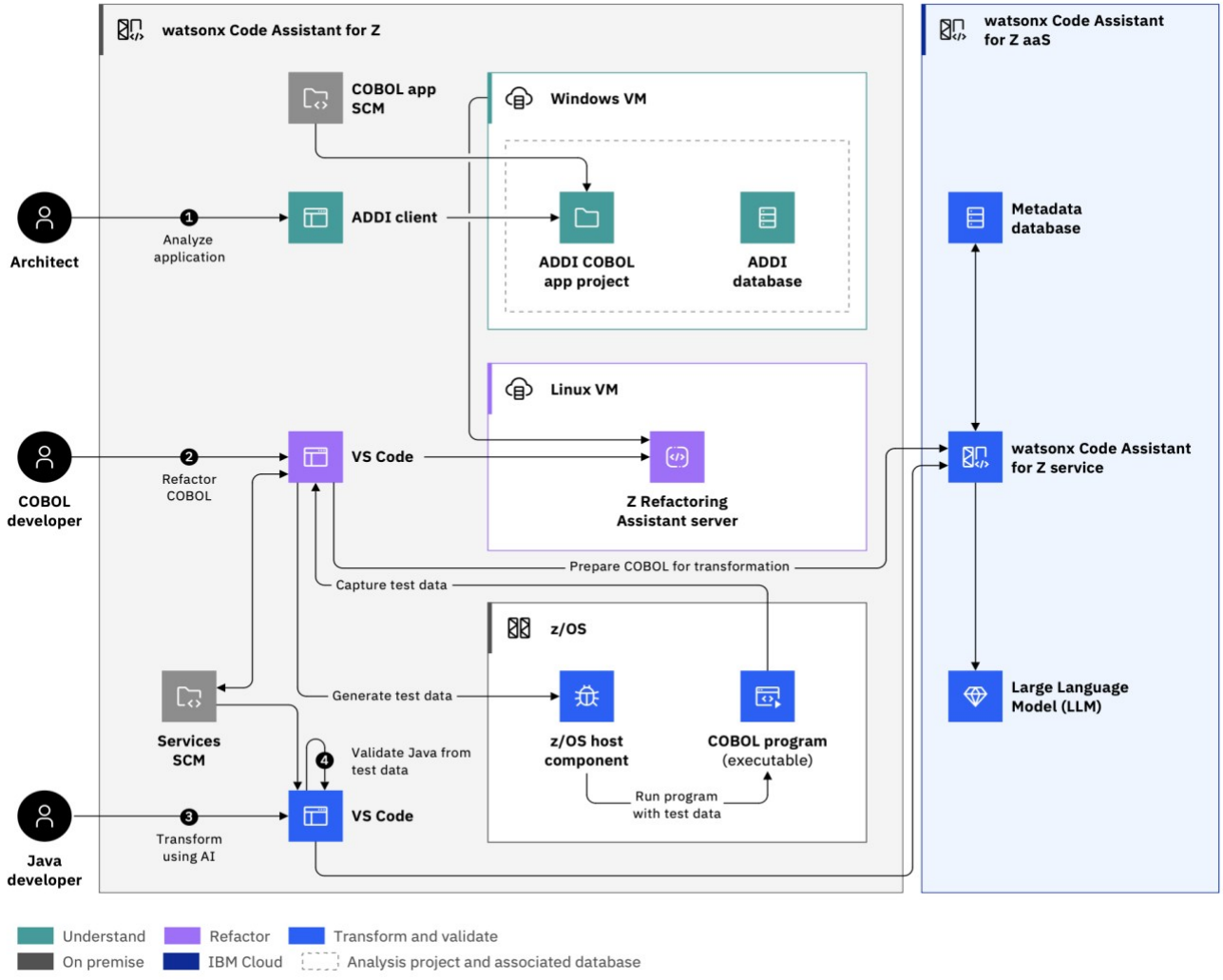
1 00 Overview



If the modernization strategy involves converting the COBOL service to Java, the COBOL developer can prepare the COBOL code for transformation, sending the necessary metadata to the Db2 database in IBM Cloud for watsonx Code Assistant for Z. After reviewing and compiling the COBOL code, the COBOL developer can use the IBM watsonx Code Assistant for Z Validation Assistant to generate COBOL test data. This test data can be used later in the process by the Java developer to validate that the Java service is semantically equivalent.

The Java developer uses the IBM watsonx Code Assistant for Z extension in Visual Studio Code (VS Code) to connect to the watsonx Code Assistant for Z service in IBM Cloud. This service is provisioned by using a Cloud account and provides access to the AI foundation model. This foundation model

also called a large language model (LLM), is trained on many programming languages and is fine-tuned for converting COBOL to Java. Each developer accesses the service by using an individual API key and connects securely to use the AI. The Java developer uses the COBOL service in the SCM repository to generate Javaclasses and business logic, and can review and compile the code to complete the Java service. The Java developer can then use the Z Validation Assistant to generate JUnit tests for each Java method to check that the code is semantically equivalent to the COBOL service.



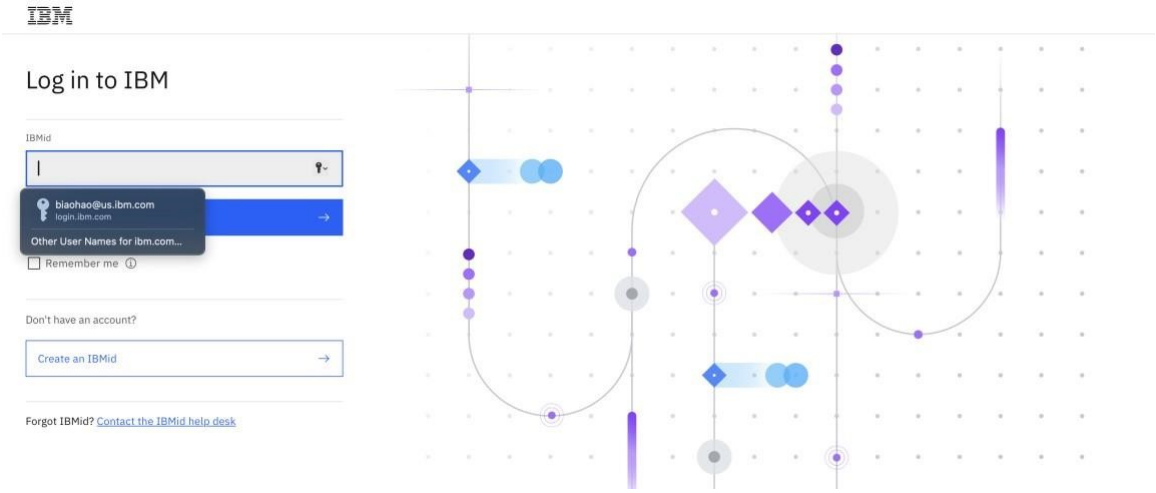
These images have been configured to communicate with each other, so you do not need to do that yourself. These configurations include:

- WCA4Z Service (1 of 3) – the LLM Service to connect to DB2 Meta Data for code transformation; the API Key for VS Code to access the LLM Service
- WCA4Z Discovery (2 of 3) – the network connection to WCA4Z Refactoring (3 of 3) via the private network interface; the SSH connection to WCA4Z Refactor (3 of 3) to start the Refactoring Assistant service; DB2 configuration to DB2 Meta Data on WCA4Z Service (1 of 3); and more

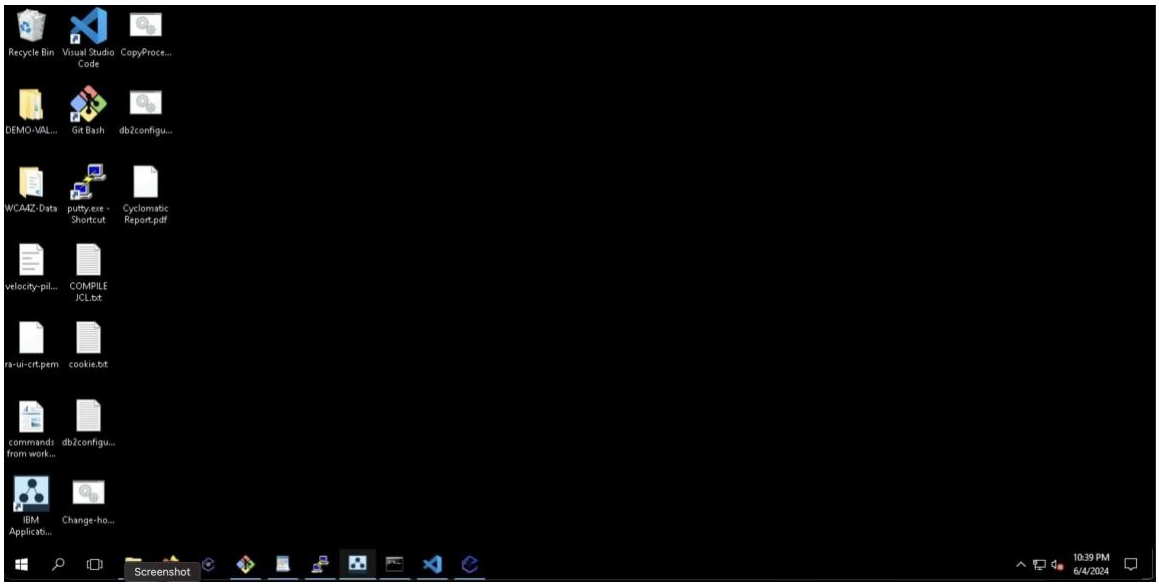
- WCA4Z Discovery (2 of 3) – the ADDI clientCOBOL project is built with either IBM sample code (or your own code reviewed by IBM); the ADDI refactoredCOBOL project is built and configured to use the DB2 for Meta Data
- WCA4Z Refactor (3 of 3) – the network connection to WCA4Z Discovery (2 of 3) via the private network interface
- WCA4Z Validation – Validate the outcome with auto-generated tests.
- VS Code – use the API key to invoke the LLM Service; and more

All the tasks will be performed for the training using the tools in the ADDI Server image.

You will log in using your IBM ID and the authentication credentials.



As indicated in the diagram below, you will be connected to the WCA4Z Discovery (2 of 3) or ADDI Server image. You will use the tools in the image, such as the ADDI Client and VS Code, to perform the tasks in the following Understand, Refactor, and Transform sections.



All the tasks will be performed for the training using the tools in the ADDI Server image.

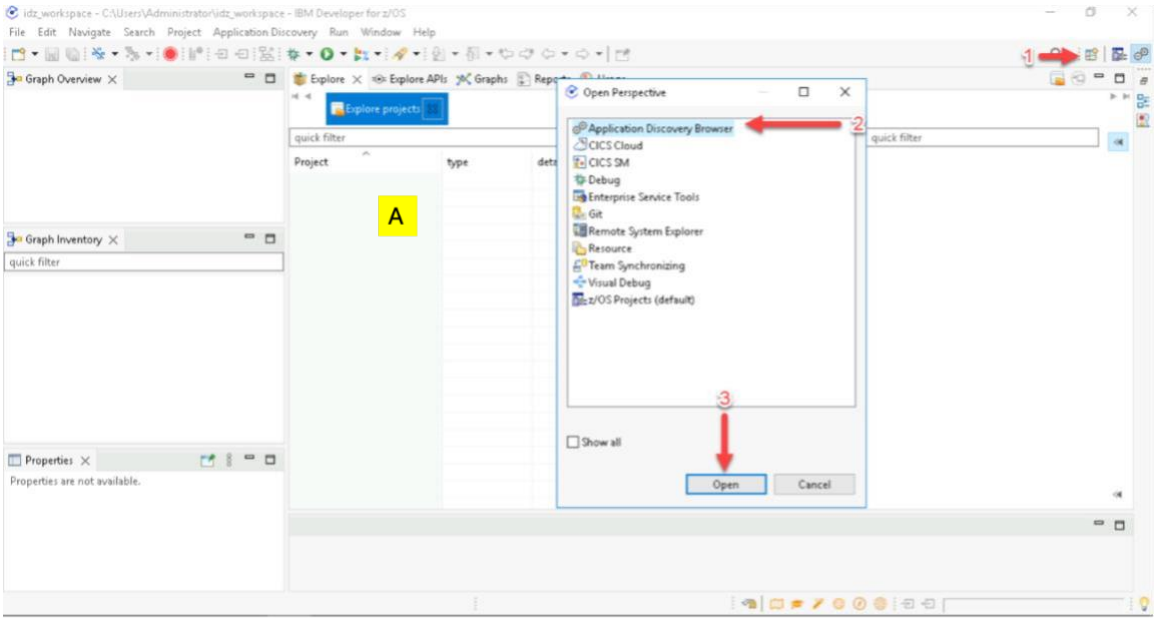
2 01 Understand

All the steps below are performed within the ADDI Server image.

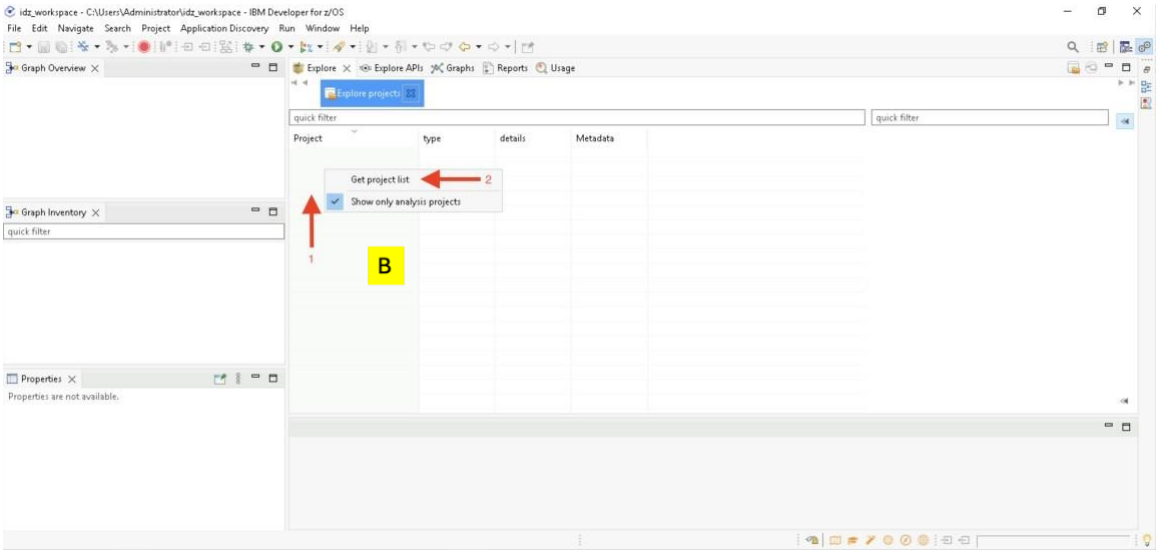
Step 1 – Open the ADDI Client

In your ADDI Server Image, open IBM Developer for z Systems.

Once opened, navigate to the Application Discovery Browser (A) perspective. Follow the instructions (1, 2, and 3, as indicated in the diagram below) to change to the Application Discovery Browser perspective if you are not in that perspective.

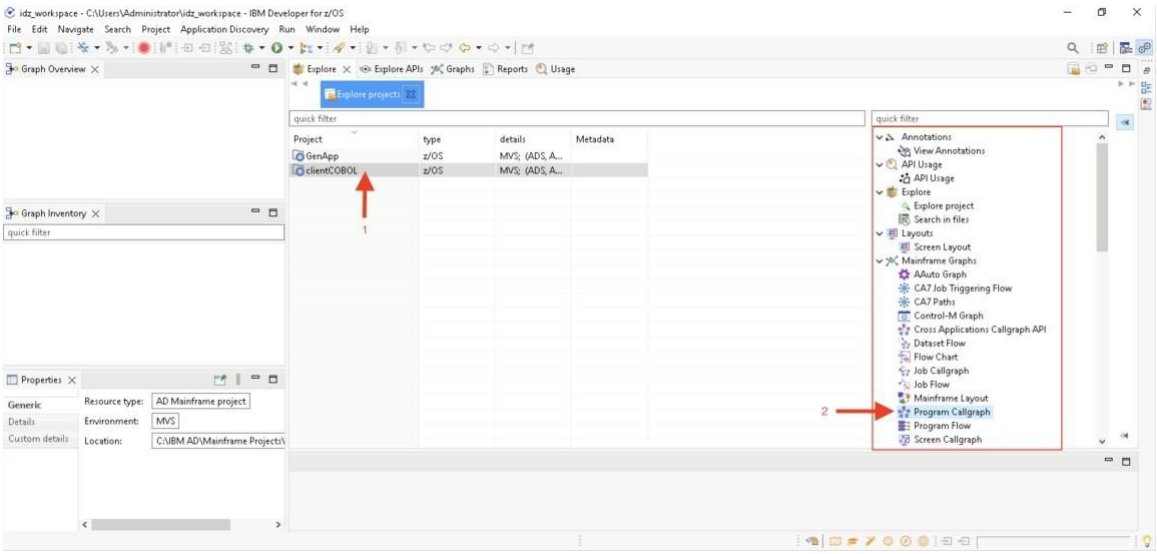


If no project is in the *Project* pane (B), right-click the empty space under *Project* to bring up the menu and select the menu item "*Get project list*", as indicated in the diagram below.

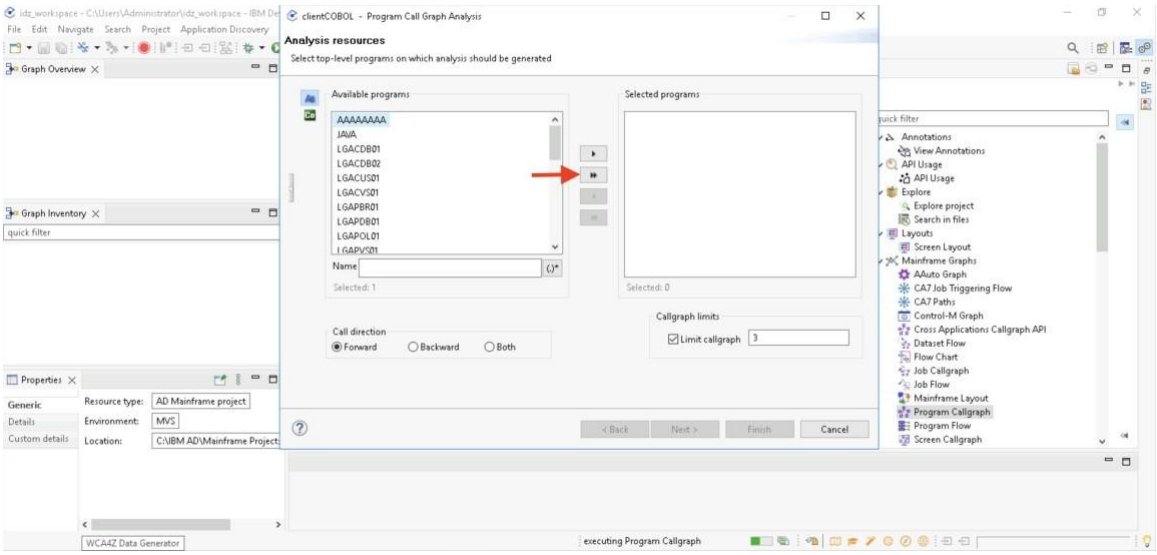


2.1 Step 2 – Build Program Callgraph

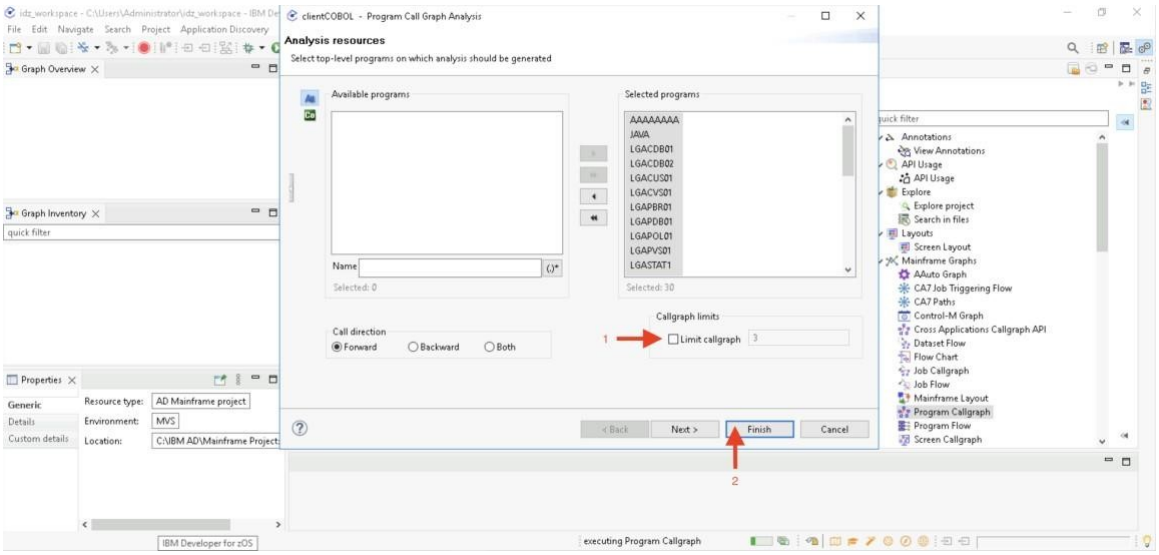
Select GenApp or *clientCOBOL* in the project list, the ADDI menu on the right side appears. Find and double-click *"Program Callgraph"* under *"Mainframe Graphs"*, as indicated in the diagram below.



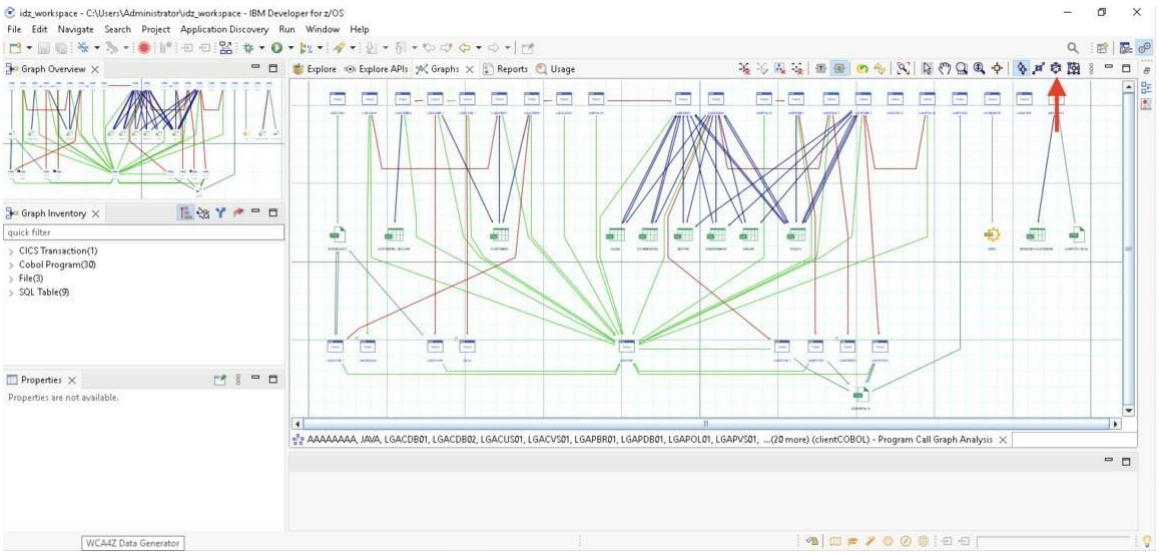
Click *">>"* to select all in *Available programs*, as indicated in the diagram below.



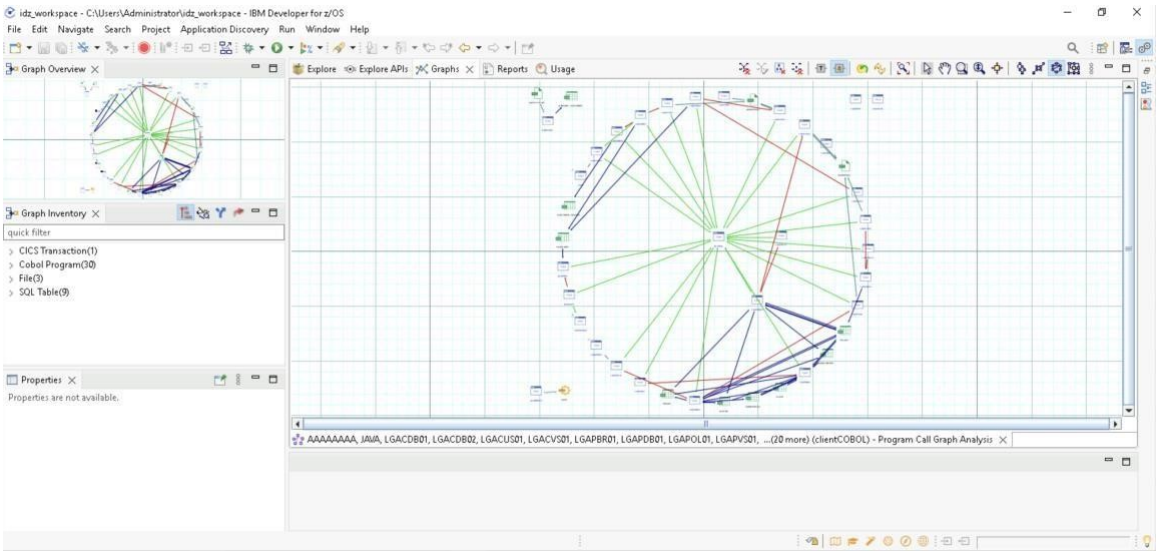
Uncheck the *Limit callgraph* and click *Finish*, as indicated in the diagram below.



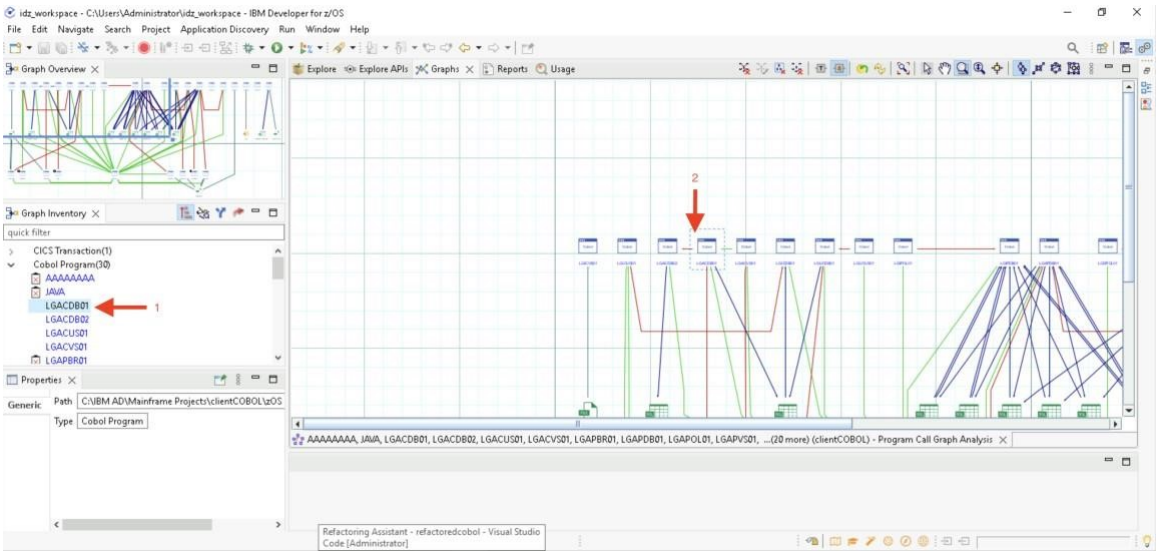
The diagram below is the program call graph. Click the icon "*Change layout to circular"*, as indicated in the diagram below.



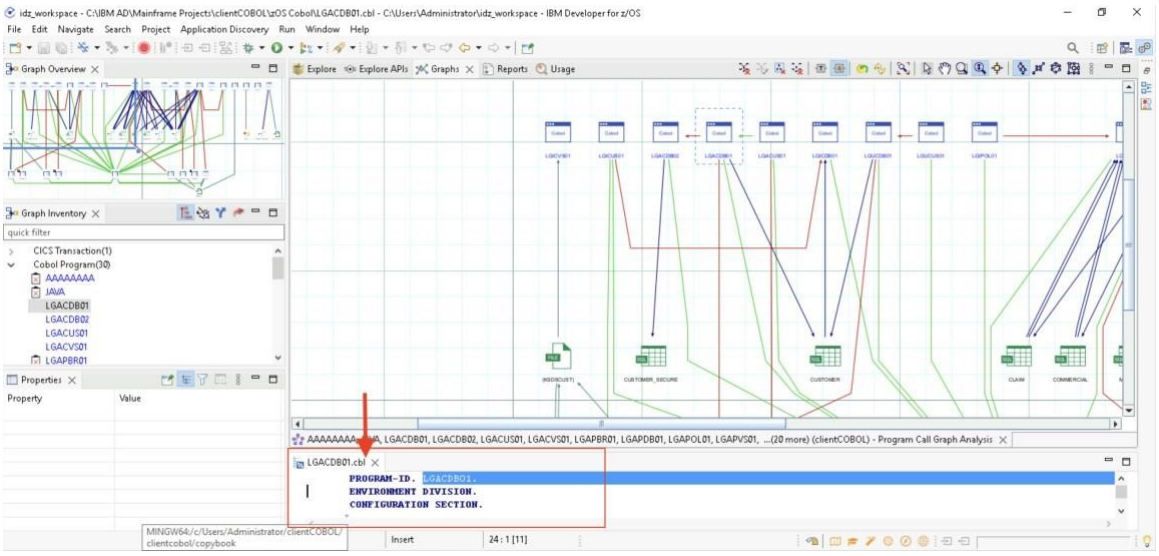
The diagram below is the circular program call graph. By clicking the different layout icons, you can change it to a different layout, such as *hierarchical*, *symmetric*, or *orthogonal*. Click the icon "*Change layout to hierarchical"* to revert back to the initial *hierarchical* layout.



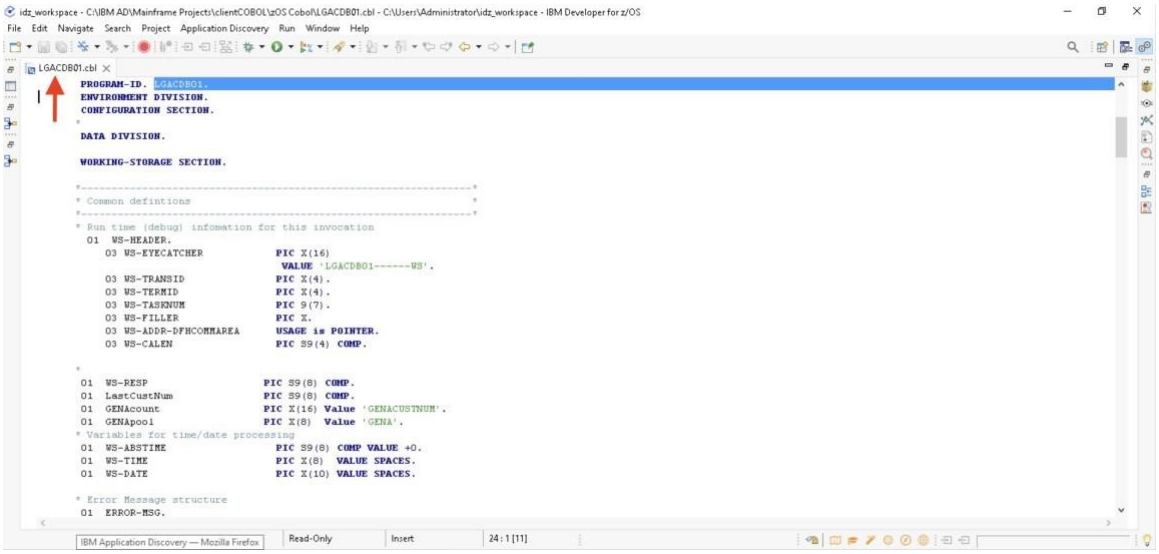
Expand on the *Cobol Programs* in the *Graph Inventory* pane on the left side and select *LGACDB01*. The diagram below shows the selected program centered in the main graph pane. Double-click the program icon to show the COBOL code in the pane below.



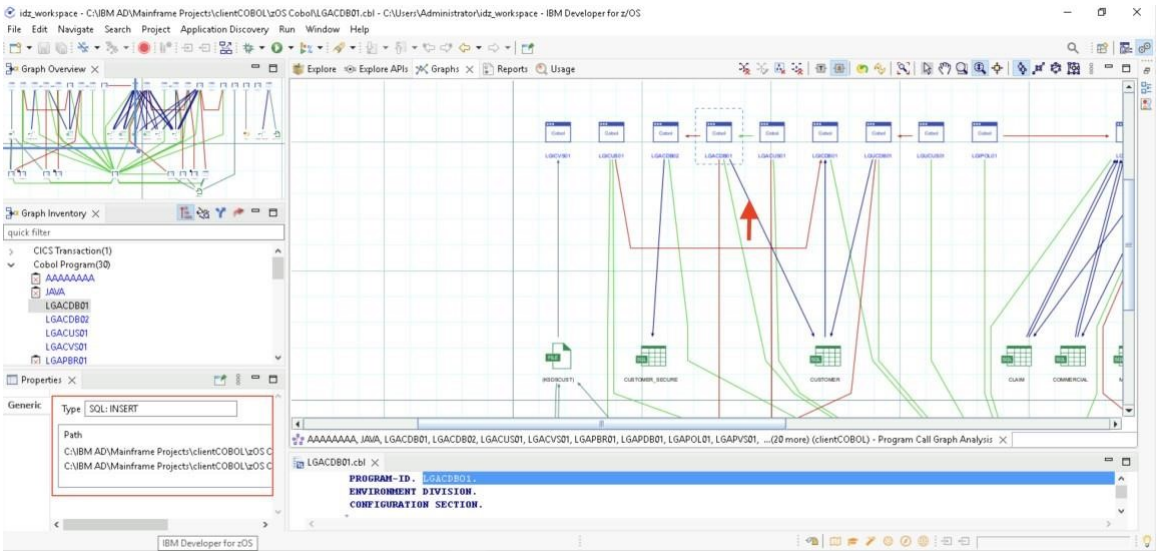
The diagram below shows the COBOL code. Double-click on the program tab *LGACDB01.cbl* to maximize the program pane.



Review the COBOL code. Double-click on the program tab *LGACDB01.cbl* to restore the program pane.



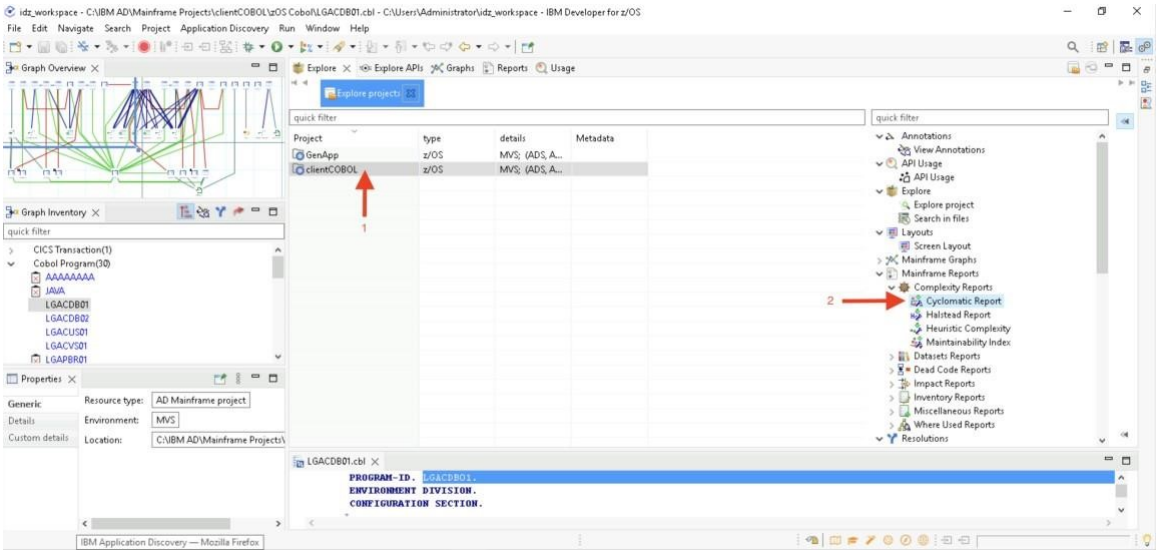
Click on the blue line between the program icon for *LGACDB01.cbl* and the table icon for *CUSTOMER* table; the *Properties* pane on the bottom left side shows it's an SQL INSERT statement, as indicated in the diagram below.



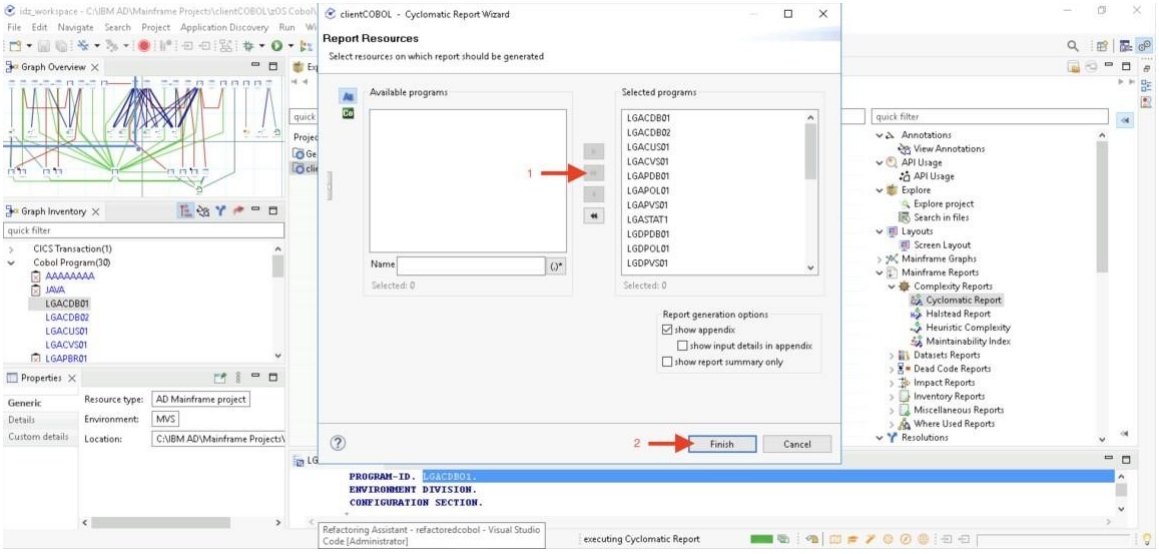
2.2 Step 3 – Create Mainframe Report

Back to *Explore projects*, select the *clientCOBOL* project, then expand on "*Mainframe*

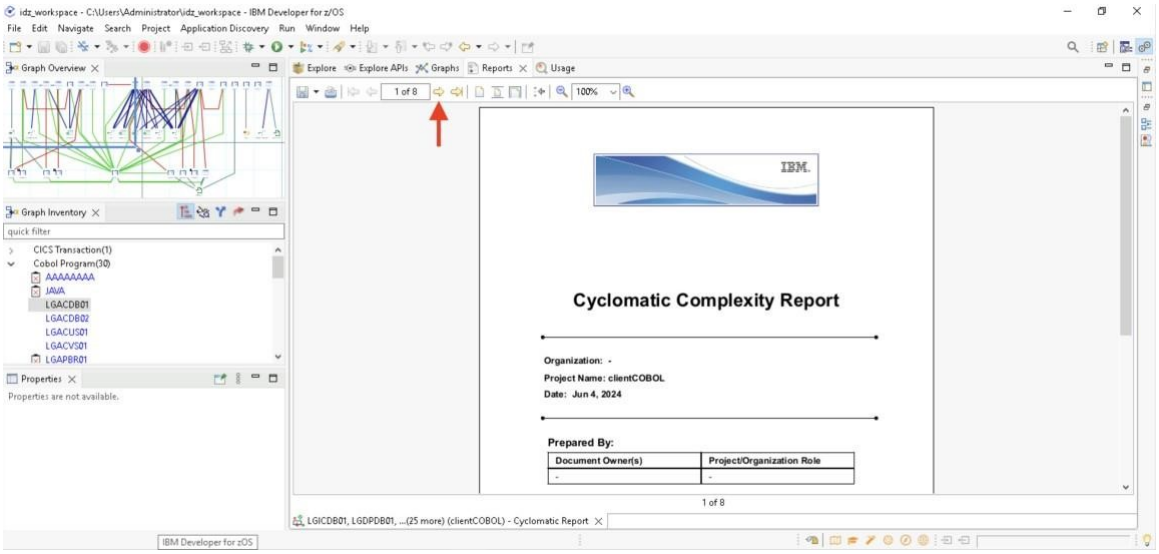
Reports > Complexity Reports" on the right side, and double-click the "*Cyclomatic Report"*, as indicated in the diagram below.



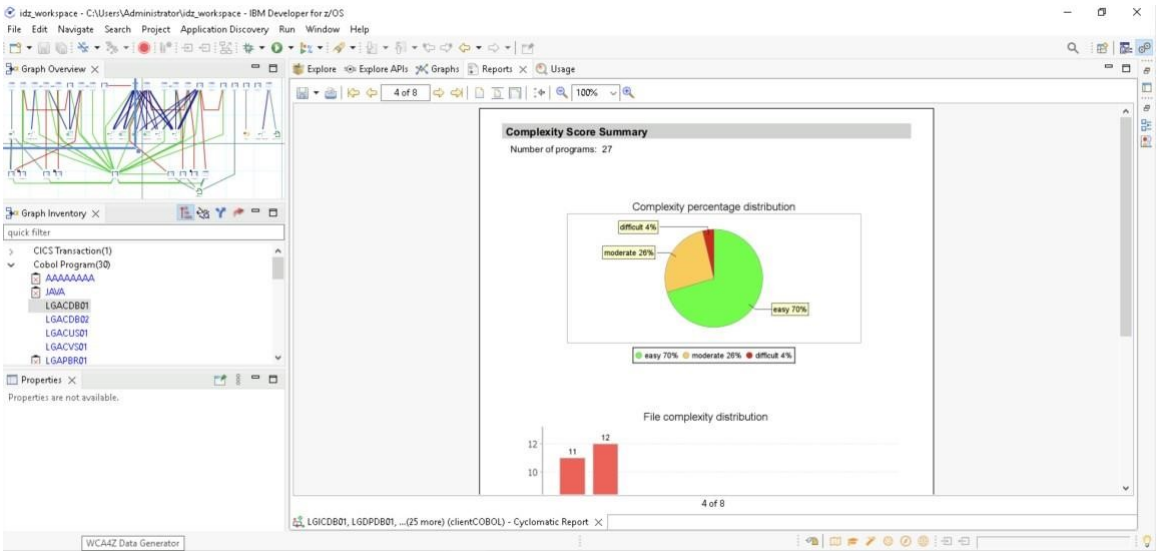
Click *">>"* to select all of the *Available programs* and then click *Finish*.



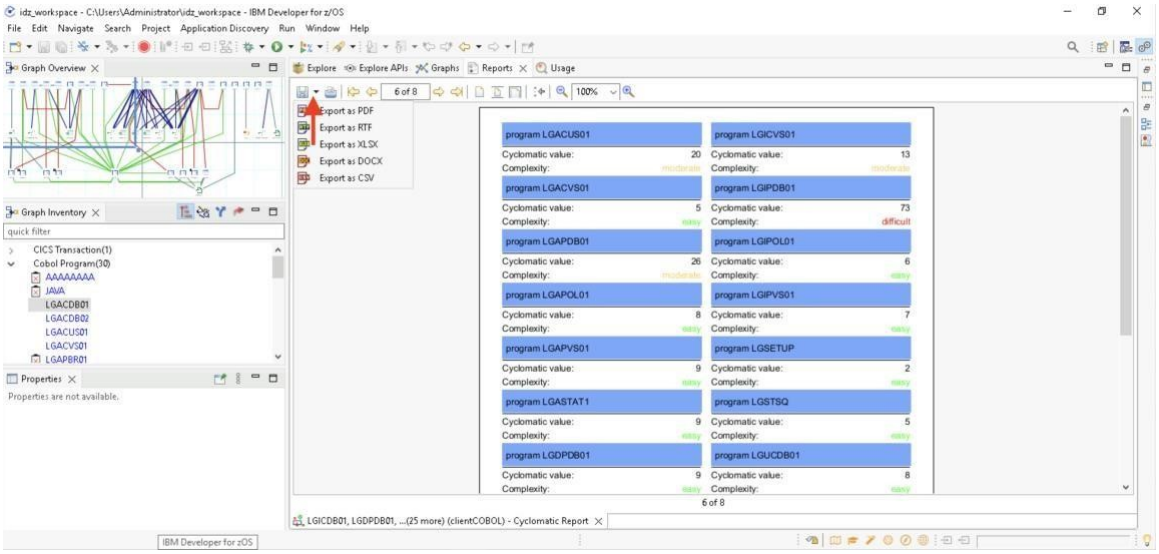
The Cyclomatic Complexity Report is shown in the diagram below. You can review it by clicking the page navigation icons, as indicated in the diagram below.



The diagram below shows another page of the report.



The diagram below shows another page of the report. You can export the report in various formats, as indicated in the diagram below.



You can generate other types of reports that are of interest to you.

2.3 Step 4 – More About ADDI

Program Callgraph and *Cyclomatic Report* above are just two examples of many ADDI functionalities. To learn more about ADDI, look at the ADDI documentation here:

https://www.ibm.com/docs/en/addi/6.1.2?topic=analyze-ui-reference

END OF UNDERSTAND SECTION

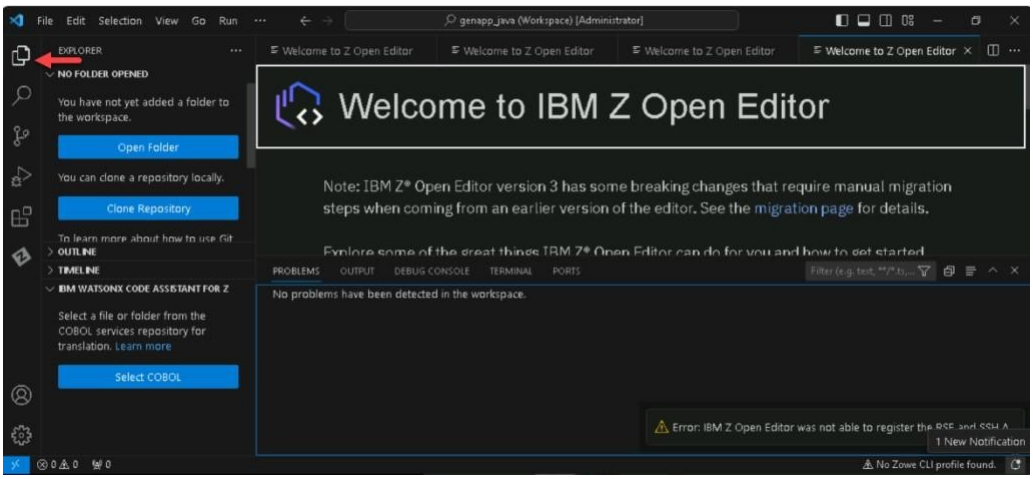
3 02 Refactor

All the steps below are performed within the ADDI Server image.

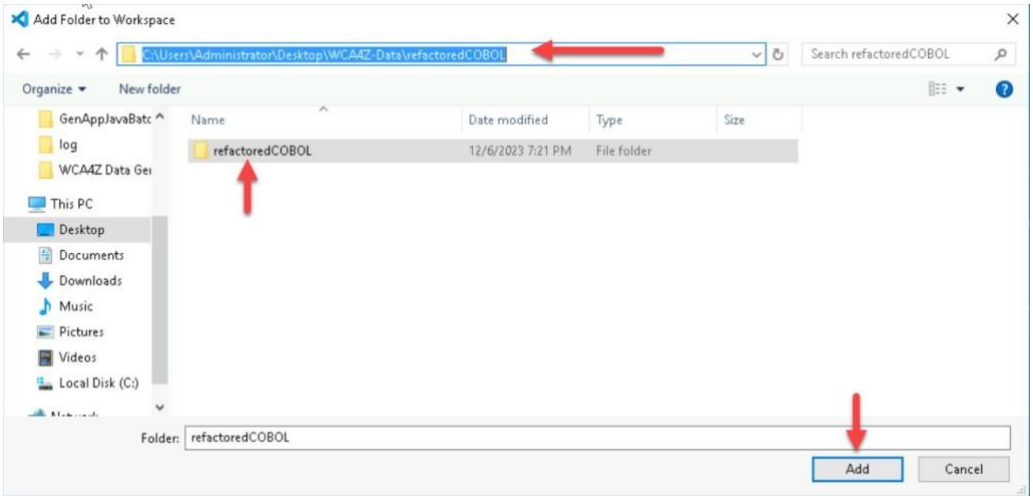
3.1 Step 1 – Select Folder for Refactored Code

You will now connect your refactored COBOL repo to VS Code, where you will later store your refactored COBOL code, ready for transformation to Java.

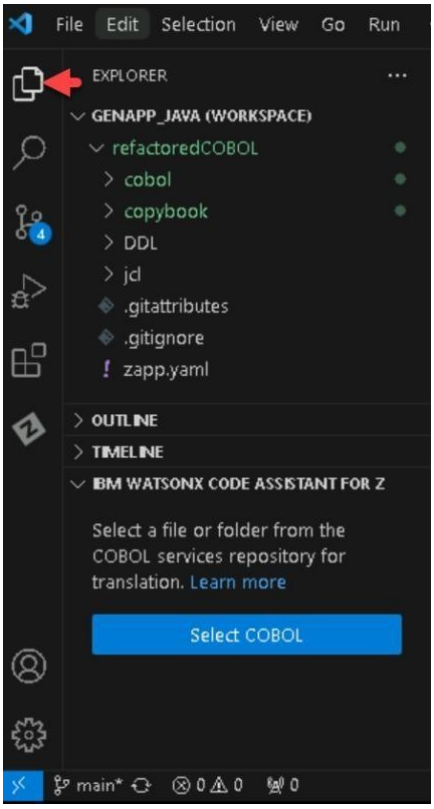
Navigate to the *Explorer* tab.

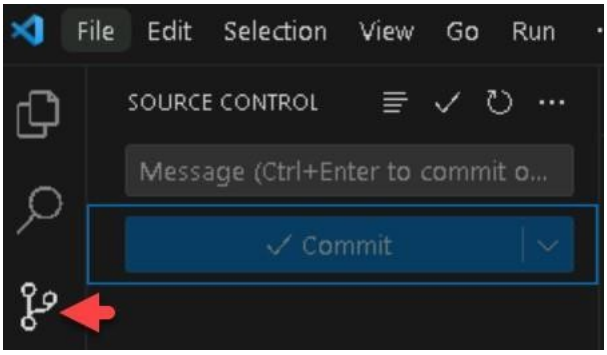


Select “*Open Folder”* and navigate to *C:\Users\Administrator\Desktop\WCA4ZData\refactoredCOBOL*, select the folder *refactoredCOBOL* and then *Add*.

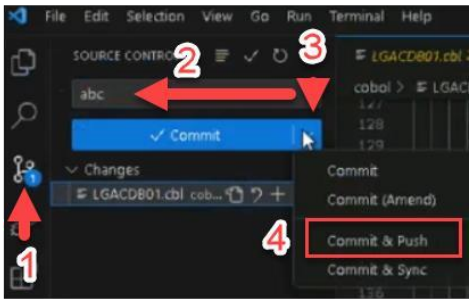


Confirm that you can see this in the *Explorer* tab and the *Git* tab in your VS Code environment.



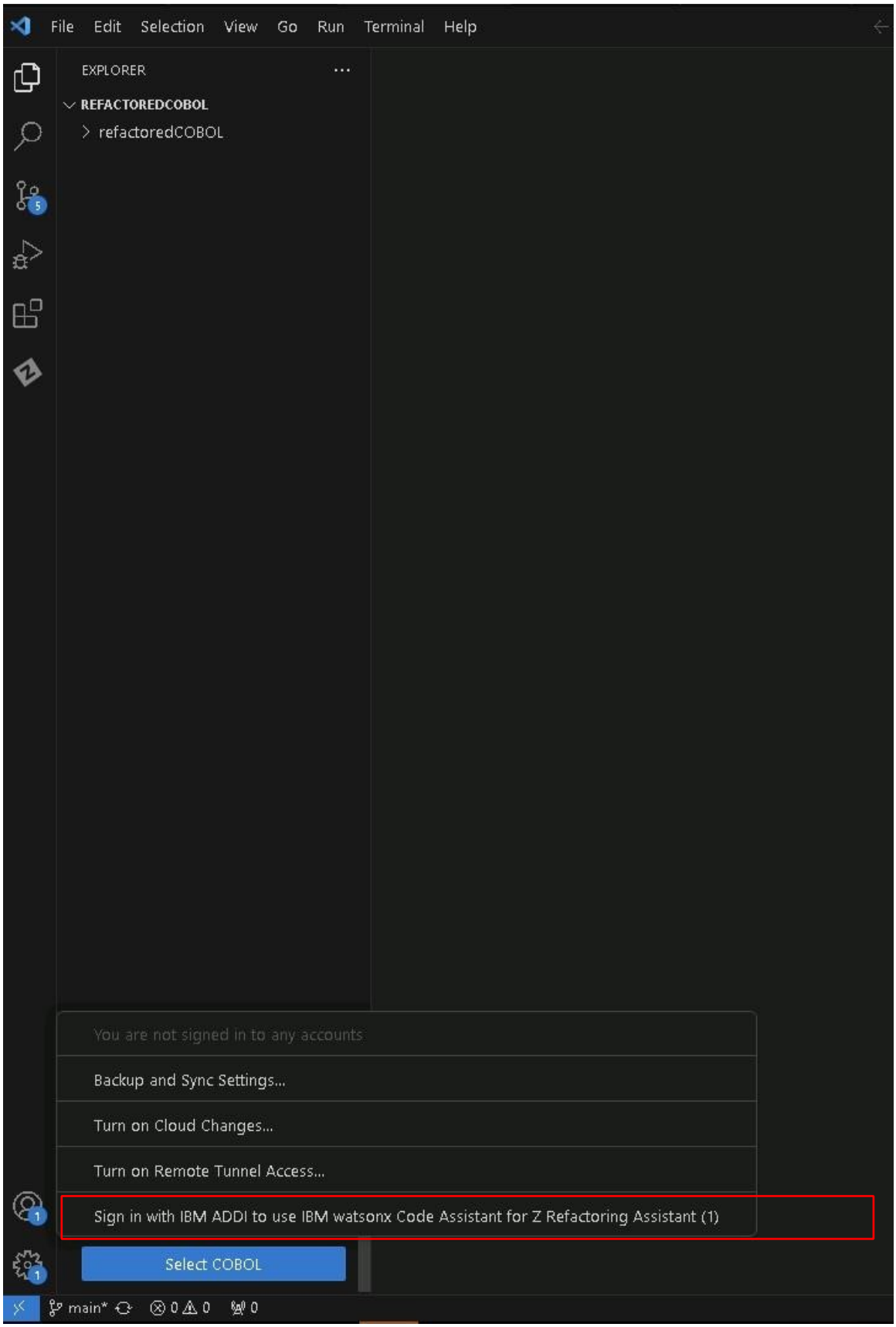


You may see changes that need to be staged and pushed to your Git repo. If so, add a commit message and then select *"Commit & Push"* to push your copybooks to your *refactoredCOBOL* Git repo.

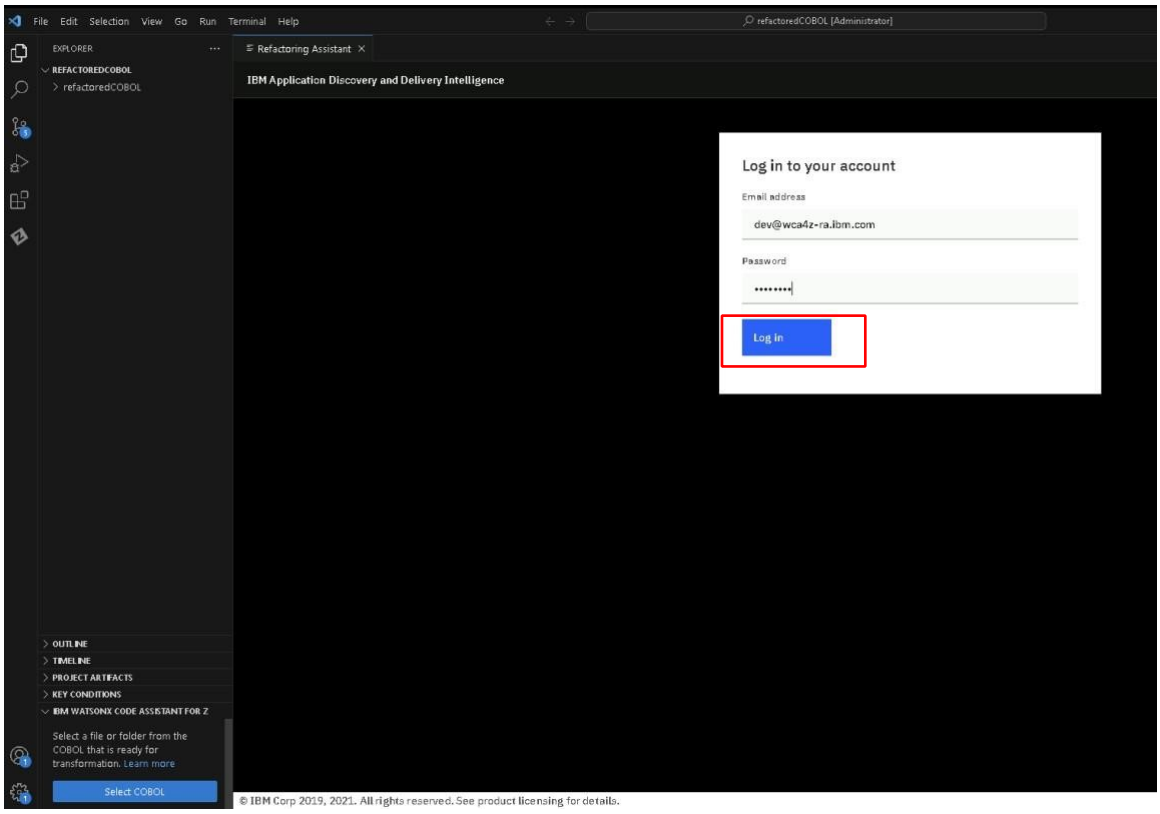


3.2 Step 2 – Sign in to Use Refactoring Assistant

You will use the new Refactoring Assistant interface via VS Code instead of the web browser. Launch VS Code and select *"Accounts > Sign in with IBM ADDI to use IBM watsonx Code Assistant for Z Refactoring Assistant"*, as indicated in the diagram below.


File Edit Selection View Go Run Terminal Help
EXPLORER
REFACTOREDCOBOL
refactoredCOBOL
You are not signed in to any accounts
Backup and Sync Settings...
Turn on Cloud Changes...
Turn on Remote Tunnel Access...
Sign in with IBM ADDI to use IBM watsonx Code Assistant for Z Refactoring Assistant (1)
Select COBOL
main*

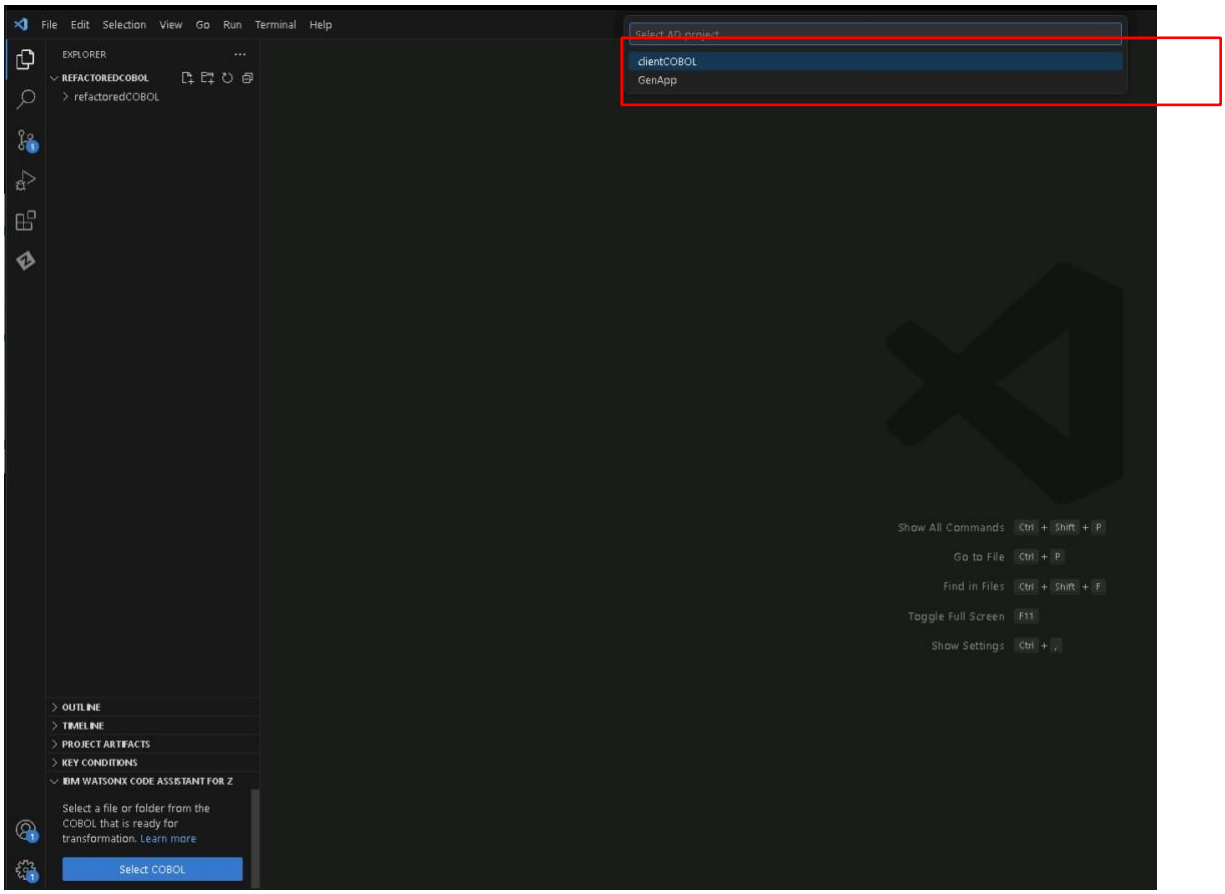
A new Refactoring Assistant window will open. Type in the credentials: dev@wca4zra.ibm.com for *userid* and password for *password*. Click "*Log In*".



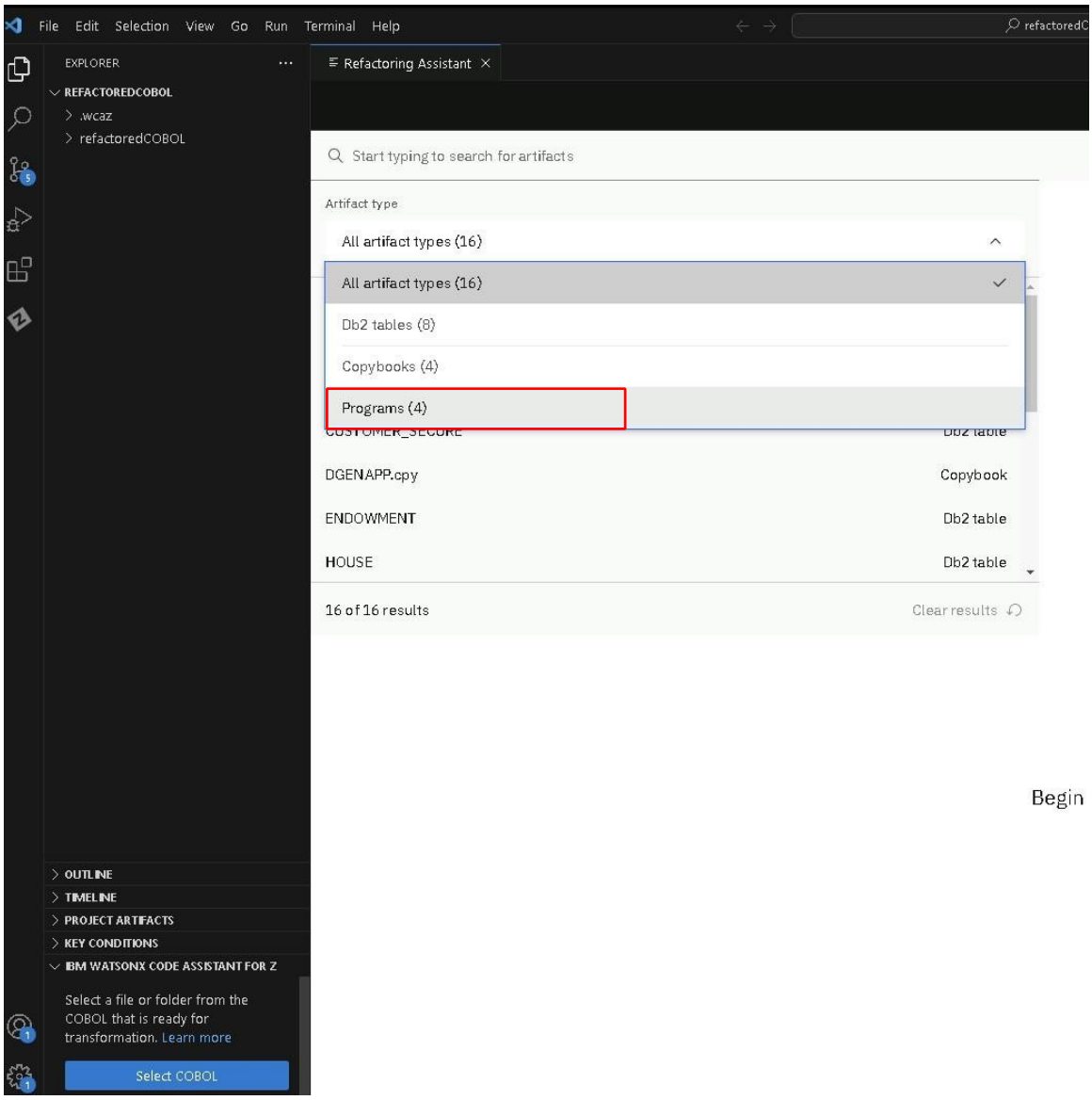
3.3 Step 3 – Refactor COBOL Code

At the top of a window you will see a list of ADDI projects, including GenApp and *clientCOBOL*.

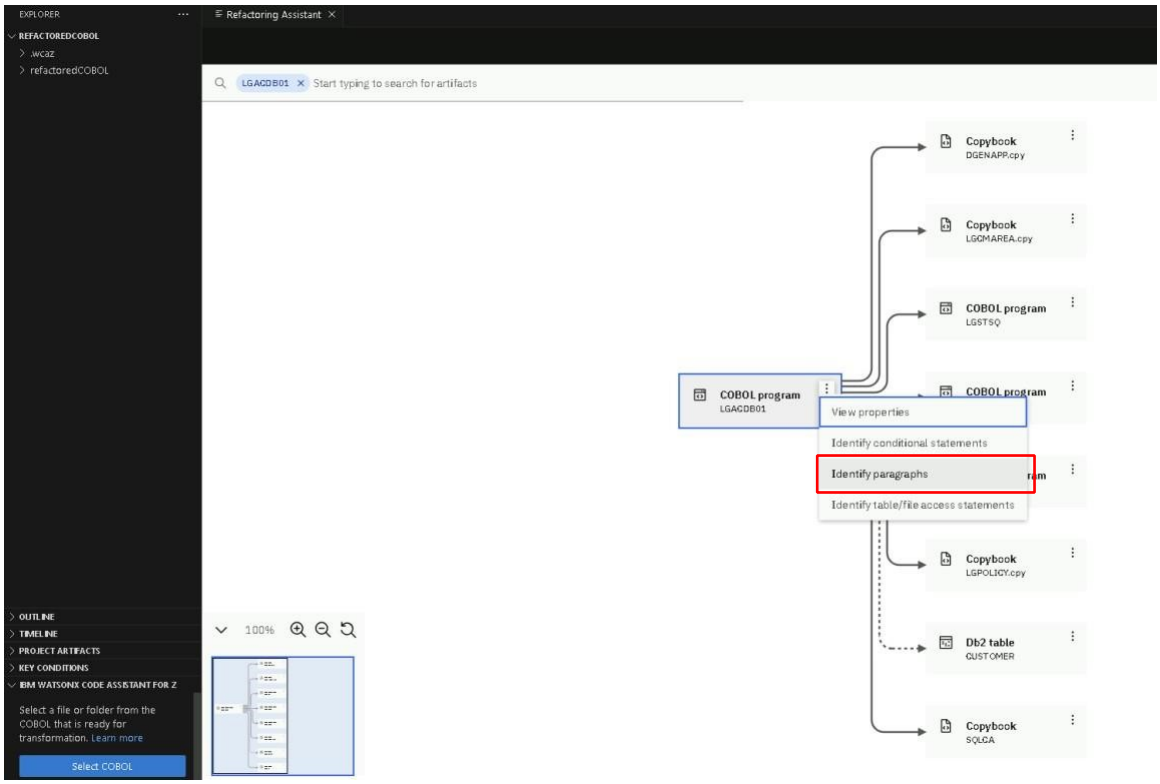
Select *GenApp* to refactor the COBOL code.



Select "*Programs"* as the artifact type and then the name of the program that you will refactor. In this guide, you select the *LGACDB01* program. The program inserts a customer into the *CUSTOMER* table, among other things.

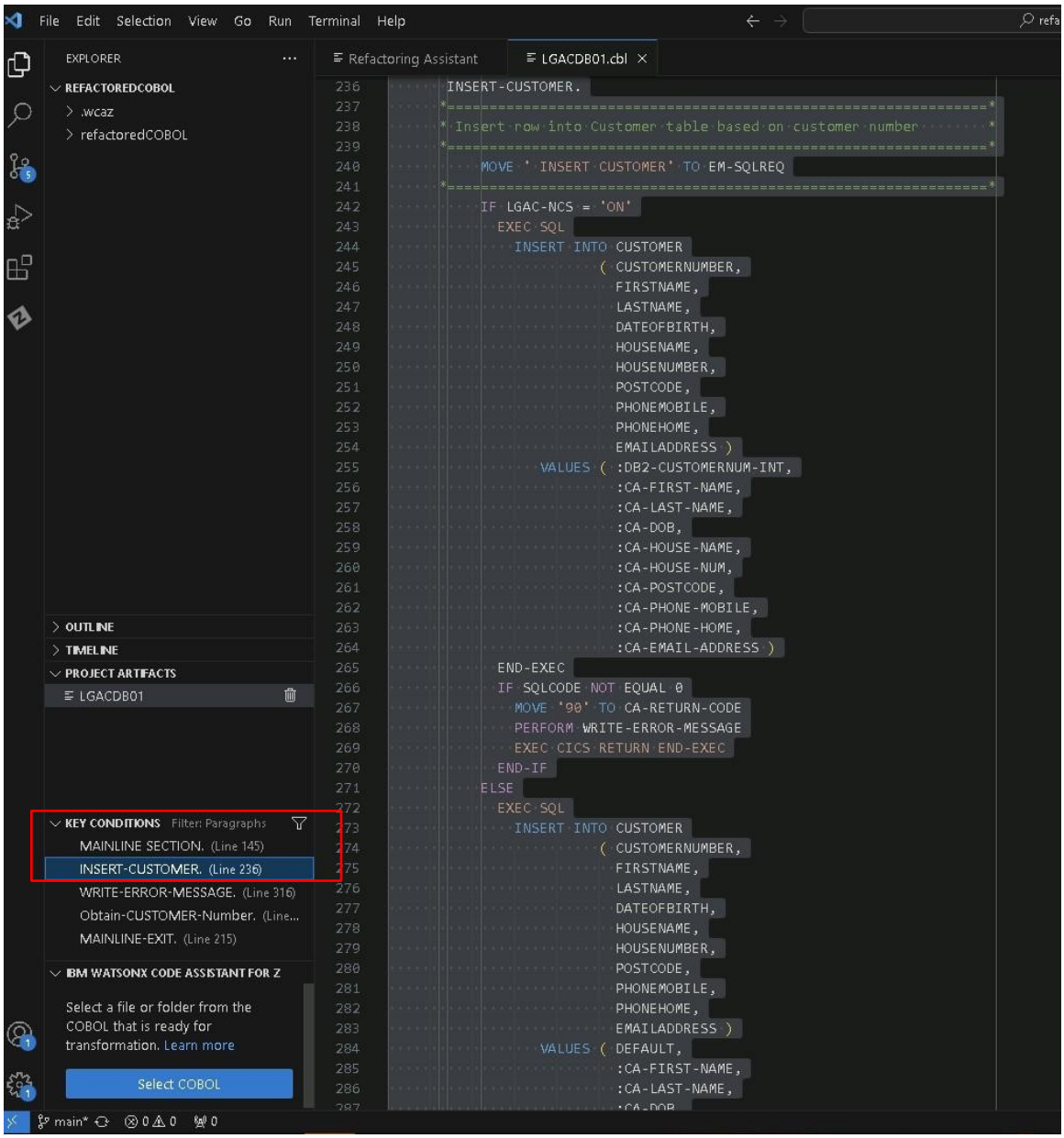


Click on the three dots next to the program that you want to refactor and select "*Identify Paragraphs*".

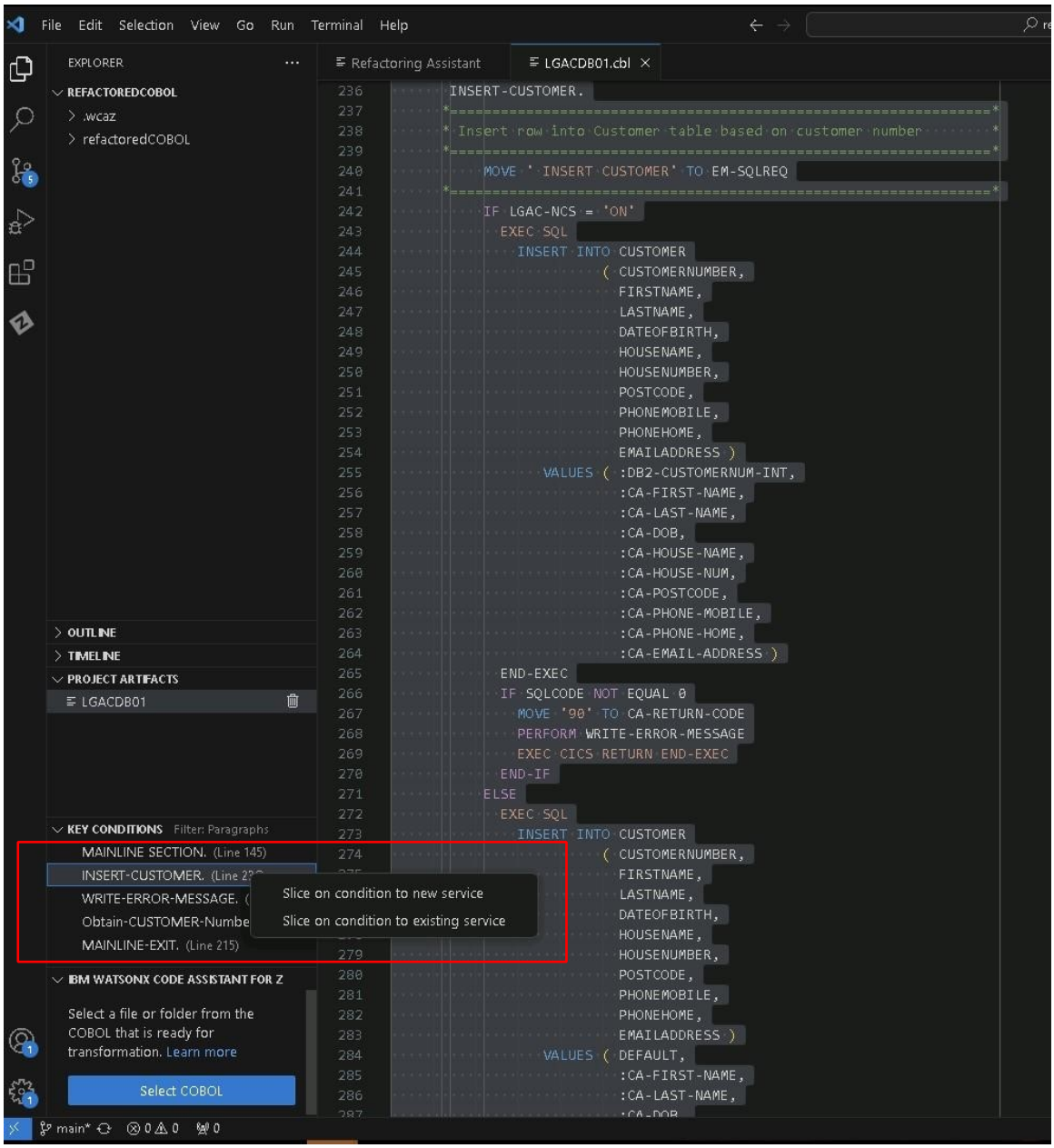


Identified paragraphs will appear in the menu on the left under "*Key Conditions*". Select the paragraph *INSERT-CUSTOMER* that you would like to work on. The selected paragraph will appear on the right side.

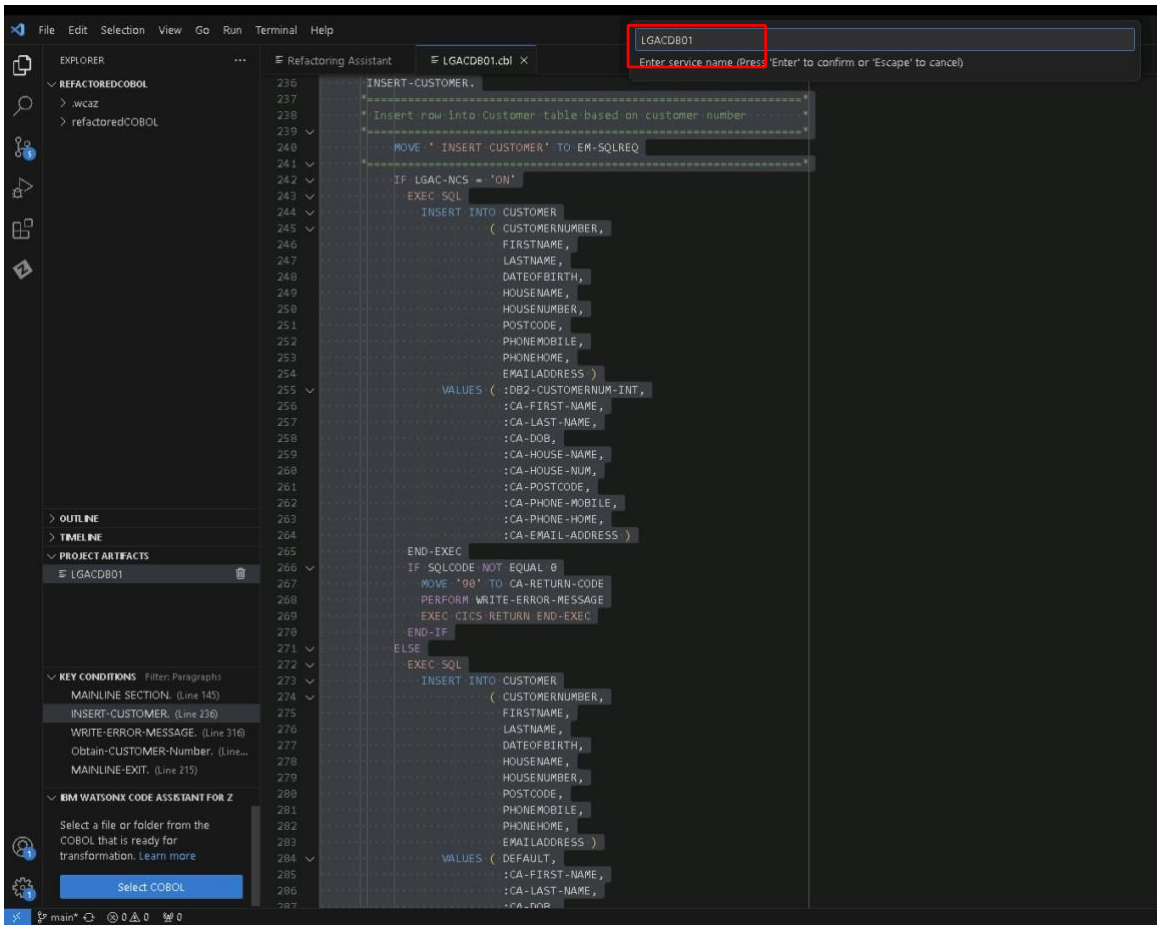
3.4



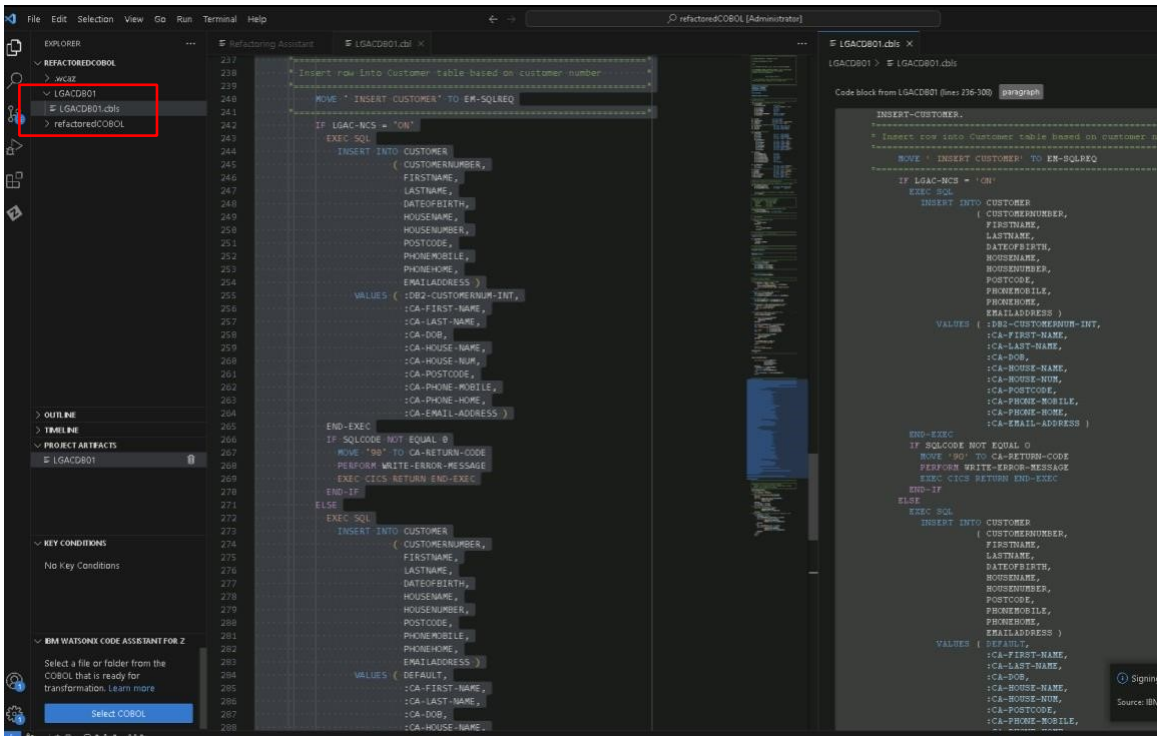
Right-click the paragraph named *INSERT-CUSTOMER* and select menu item "*Slice on condition to new service*".



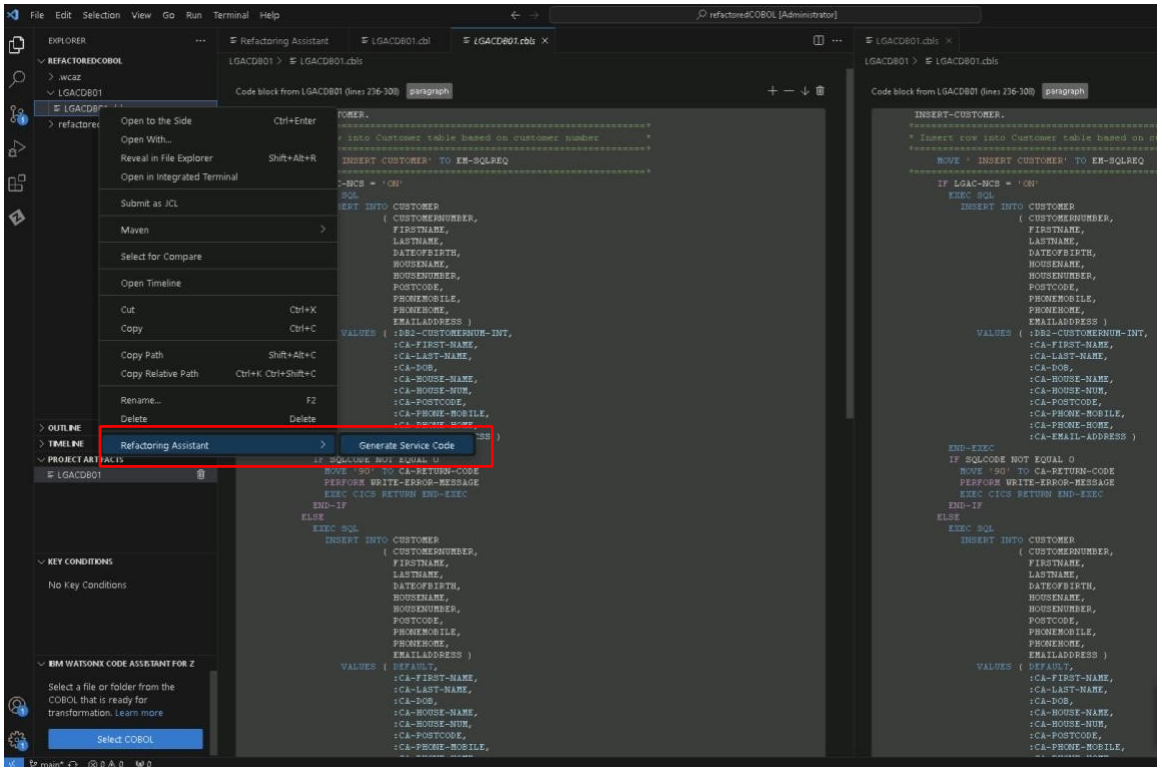
You will be prompted to type the name for a new service. Give it the name *LGACDB01*, and press the *Enter* key, as indicated in the diagram below.



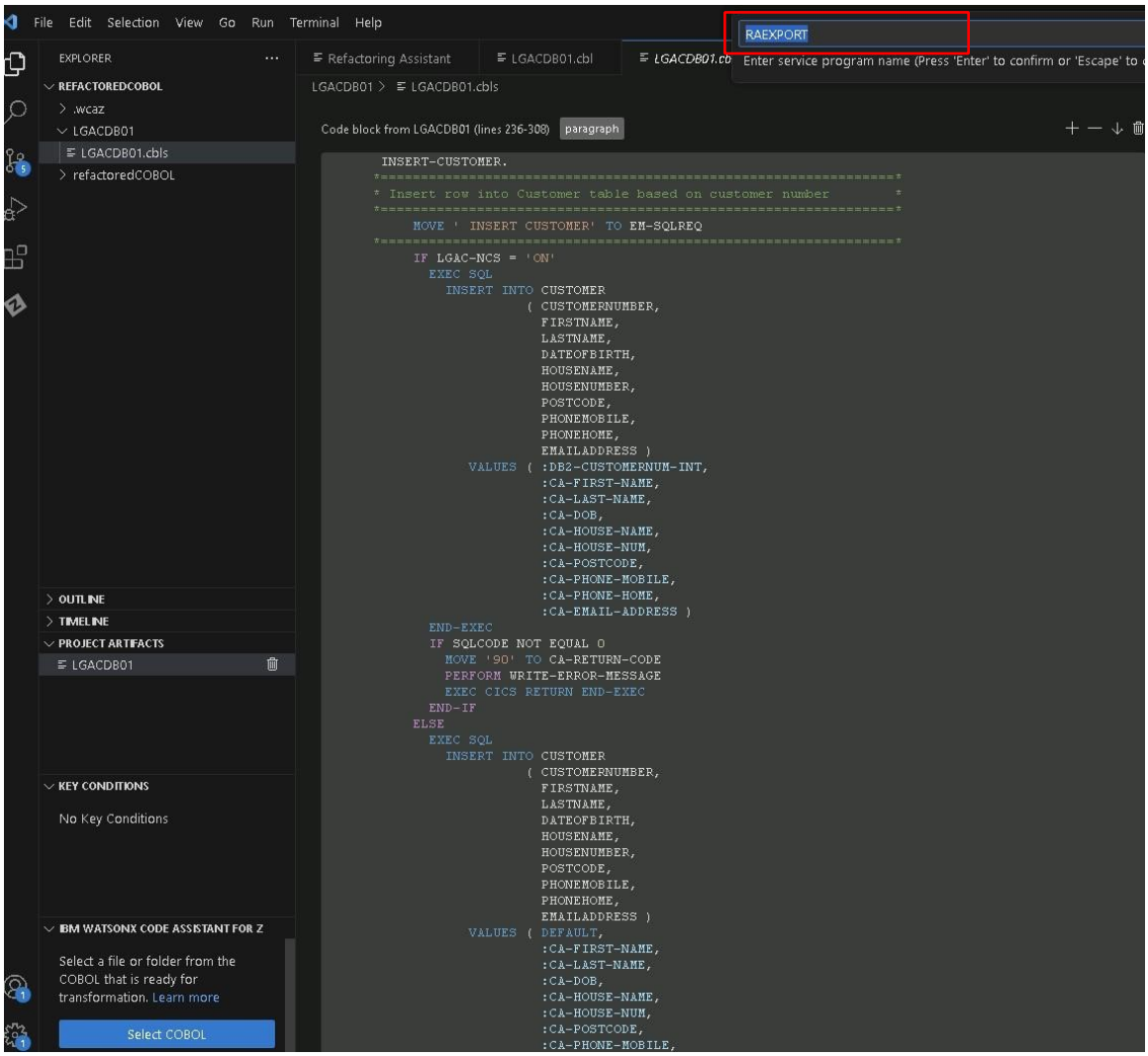
In the file explorer on the left side, you will see a new subfolder for a new COBOL service, *LGACDB01.cbls*. The name matches the service name you gave in the previous step.



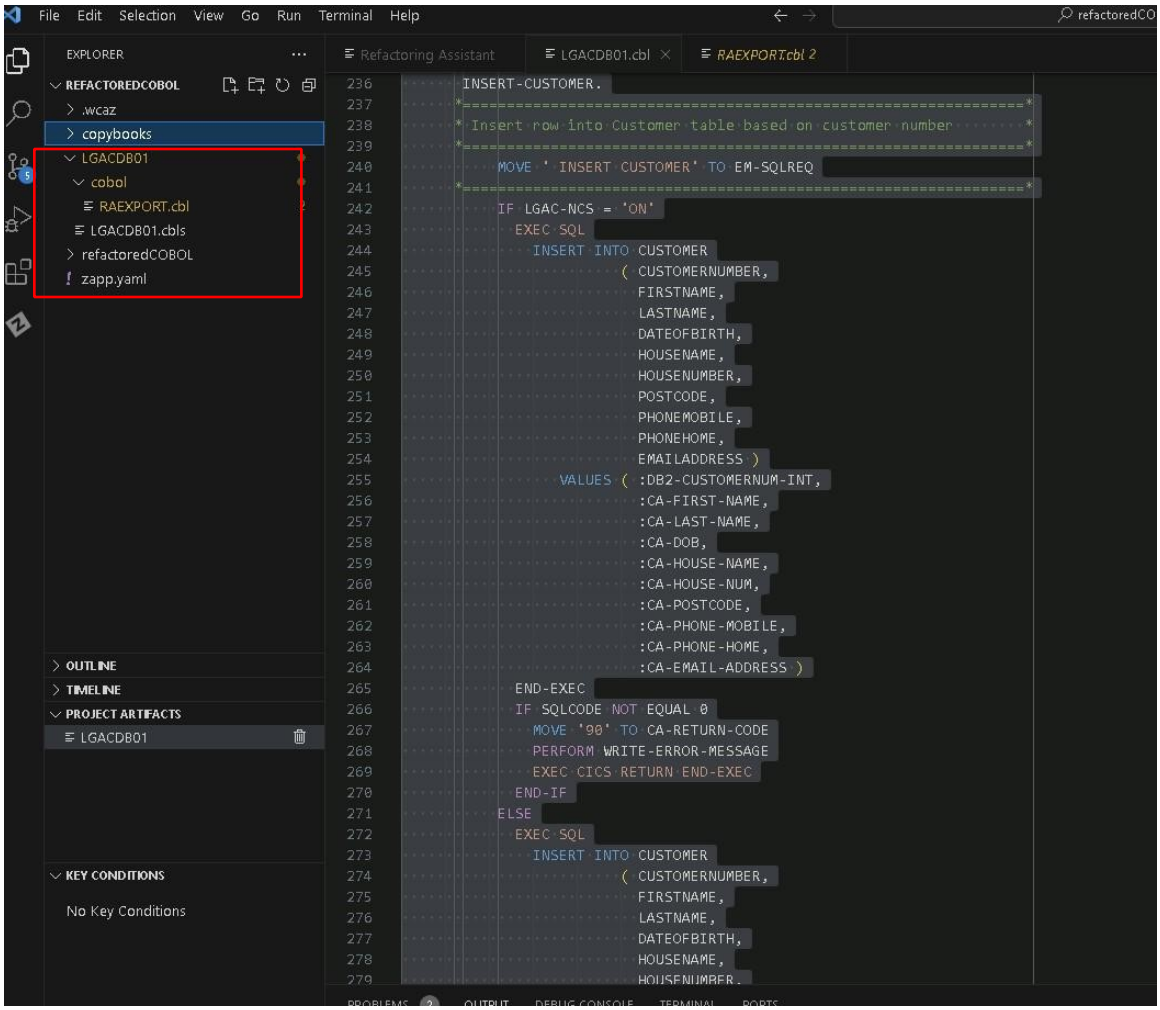
Select the *LGACDB01.cbls* file on the left, right click and select "*Refactoring Assistant > Generate service code"*.



You will be prompted for a service name. Enter *RAEXPORT* and then press the *Enter* key.

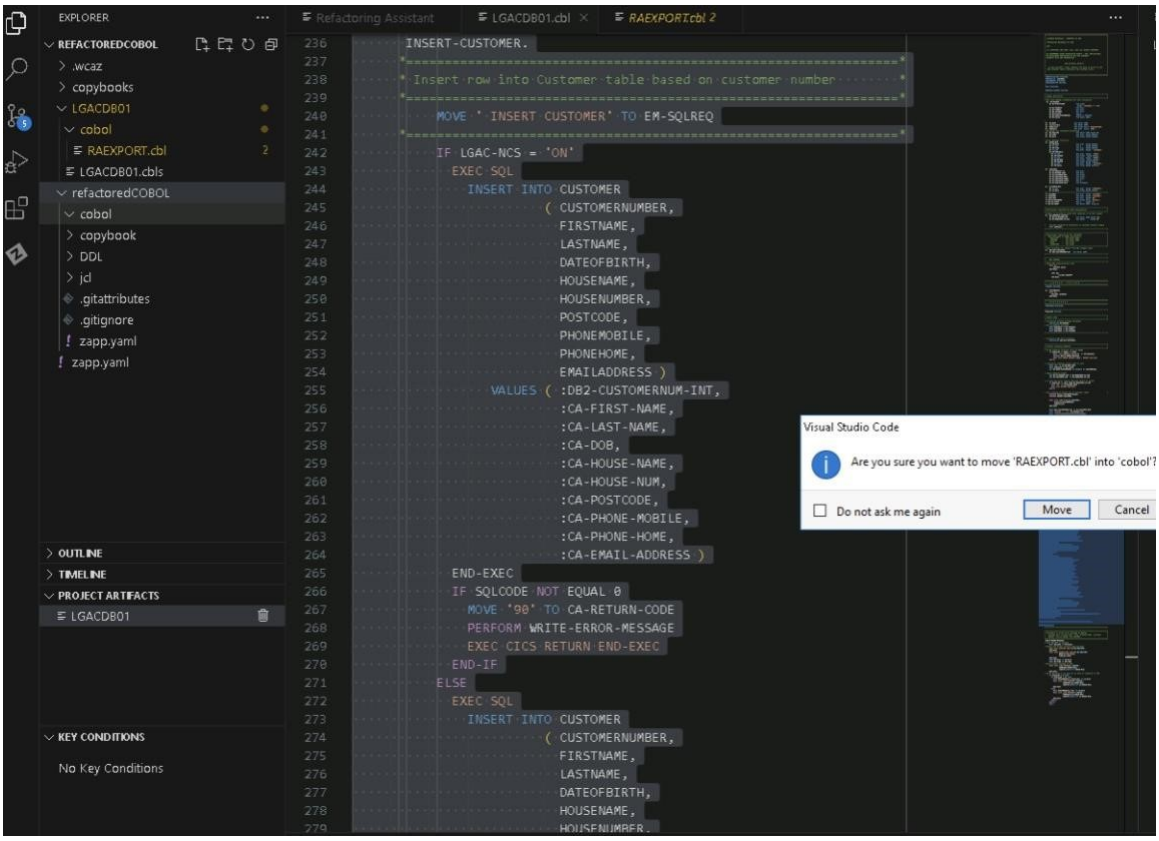


You will see the refactored COBOL code *RAEXPORT.cbl* in the folder *LGACDB01/cobol*, as indicated in the diagram below.

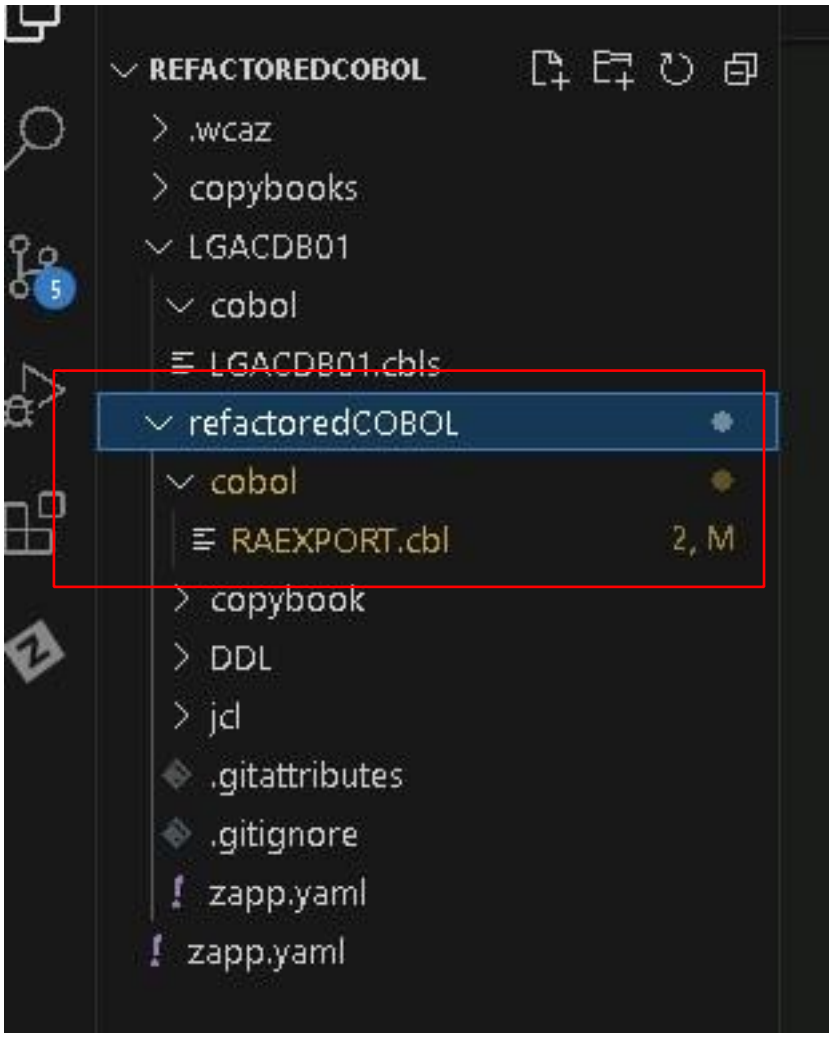


IMPORTANT!!! DO NOT SKIP THIS STEP.

To correctly execute all the next steps and generate Java code, you will need to drag and drop refactored COBOL file *RAEXPORT.cbl* to the folder *refactoredCOBOL/cobol*, all within VS Code file explorer. The refactored COBOL program must reside in the folder *refactoredCOBOL/cobol*.



Move the refactored COBOL program to *refactoredCOBOL/cobol* folder in VS Code. You may see changes that need to be staged and pushed to your Git repo. If so, add a commit message and then *"Commit & Push"* your copybooks to your *refactoredCOBOL* Git repo. You should be all set to generate the Java code.



END OF REFACTOR SECTION

2024 IBM Corporation

4 03 Transform

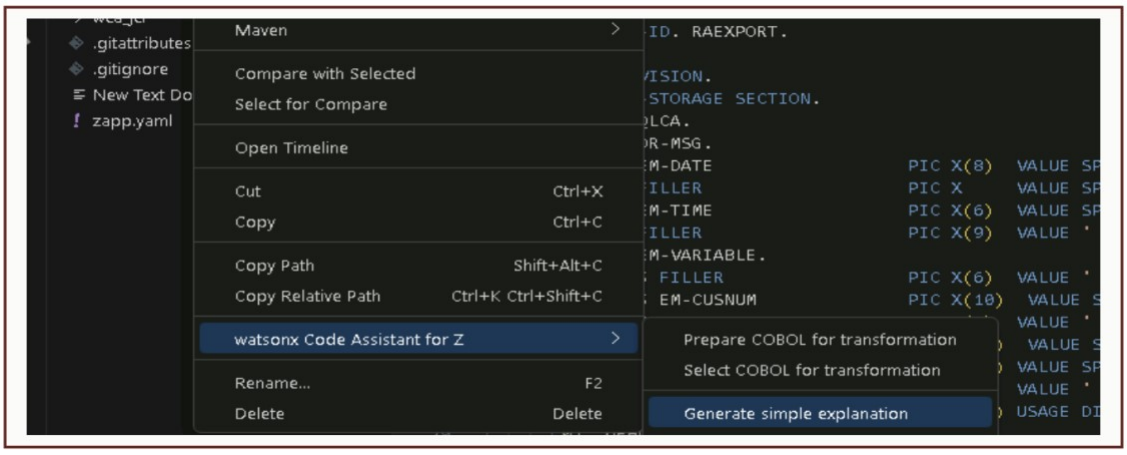
All the steps below are performed within the ADDI Server image.

Before you start to transform the refactored COBOL to Java:

1. Ensure your refactored COBOL code is in your *refactoredCOBOL* repo
2. Git Push your changes to the Git repo

Step 1 - Executing Code Explanation:

In your VSCode, right click on your refactored COBOL program -> watsonx code Assistant for Z -> Generate simple explanation



4.1 Step 2 – Build Meta Data

Run the *automation.bat* located in:

"C:\Program Files\IBM Application Discovery and Delivery Intelligence\WCA4Z Data Generator" via a Windows Command Prompt (cmd) in your ADDI Server image.

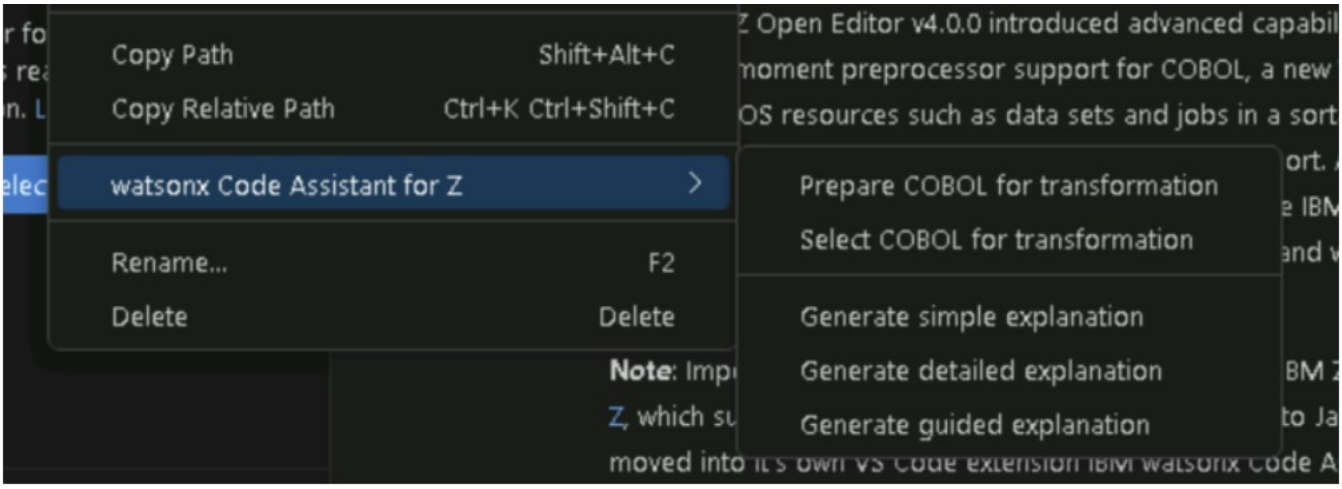
cmd> cd "C:\Program Files\IBM Application Discovery and Delivery Intelligence\WCA4Z Data Generator"

cmd> automation.bat

This step populates the DB2 Meta Data, providing the context the LLM Service uses to generate Java classes and methods.

4.2 Step 3 – Select COBOL Program

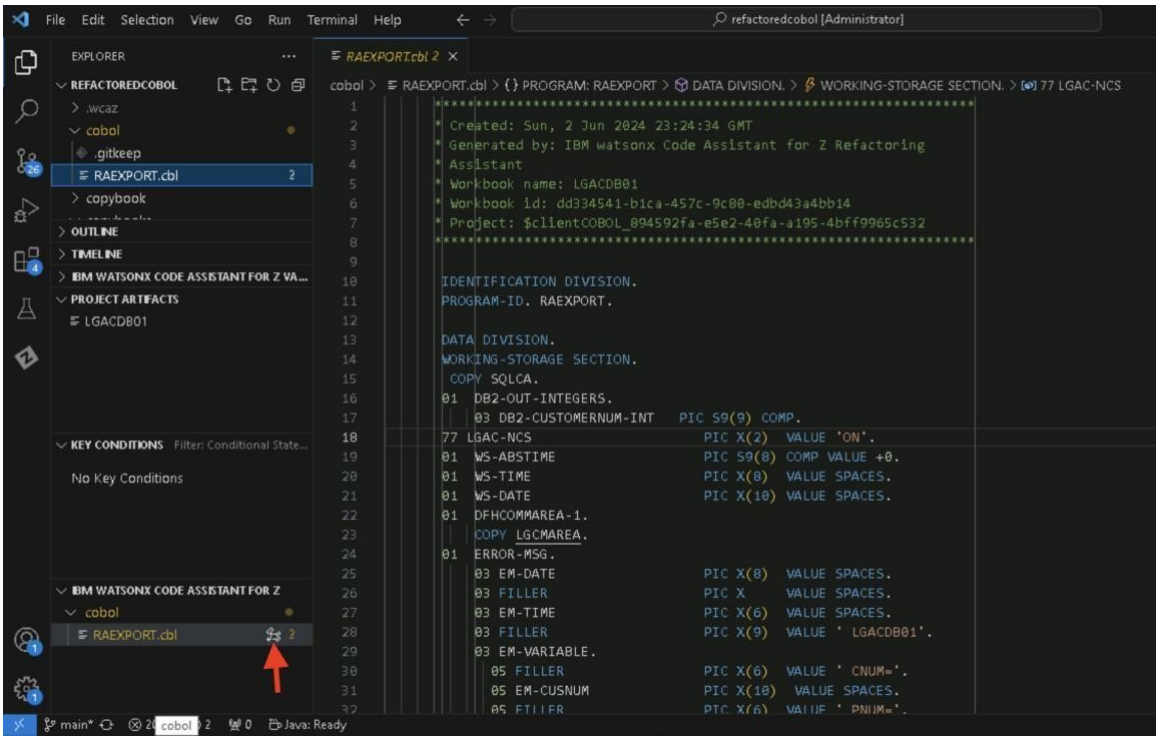
Right-click on the refactored COBOL program *RAEXPORT.cbl* you want to transform to Java, select "*watsonx Code Assistant for Z > Select COBOL for transformation*", as indicated in the diagram below.



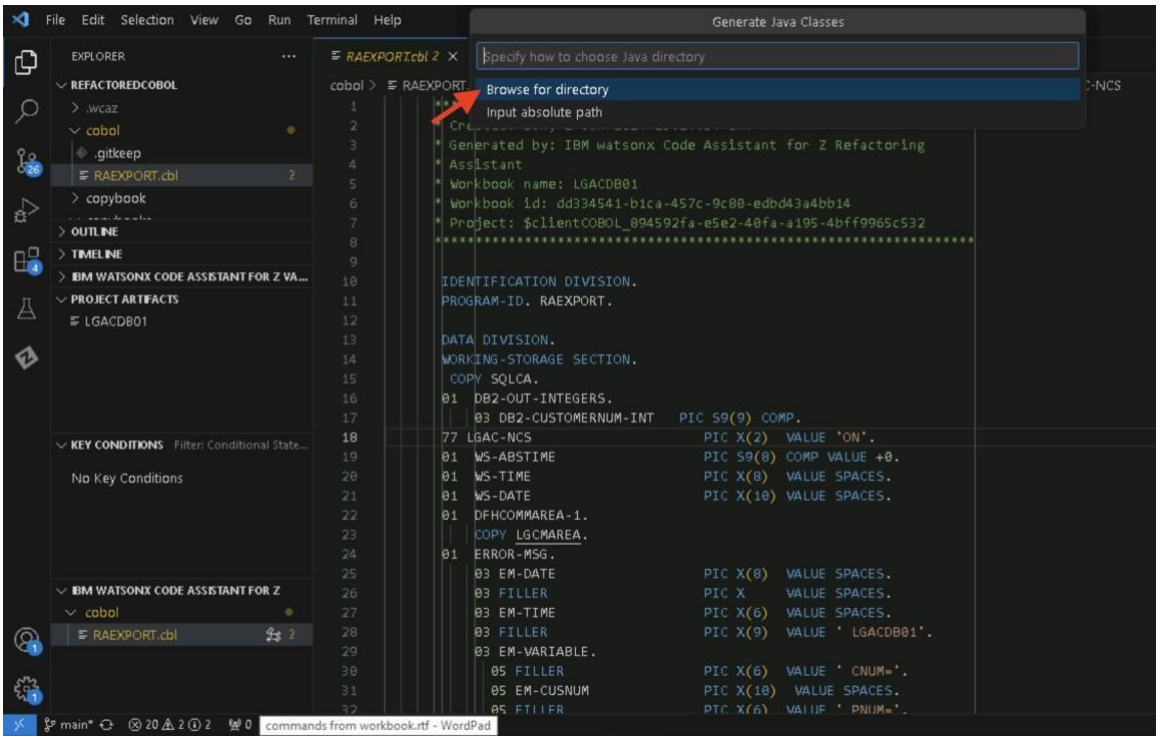
This adds the selected COBOL program *RAEXPORT.cbl* to "*IBM WATSONX CODE ASSISTANT FOR Z > cobol*" folder located on the bottom of the *Explorer* pane on the left side, as indicated in the diagram below.

4.3 Step 4 - Generate Java Classes

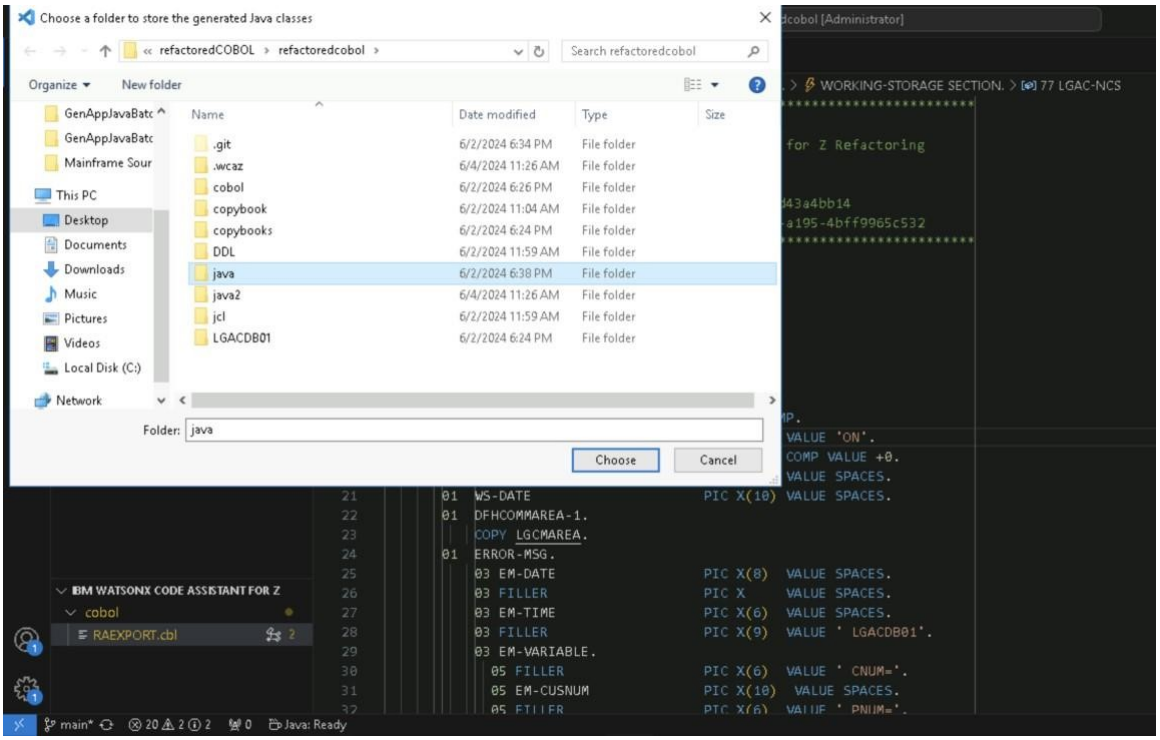
Click the "Generate Java classes" icon (as indicated by the red arrow in the diagram below) to generate Java classes.



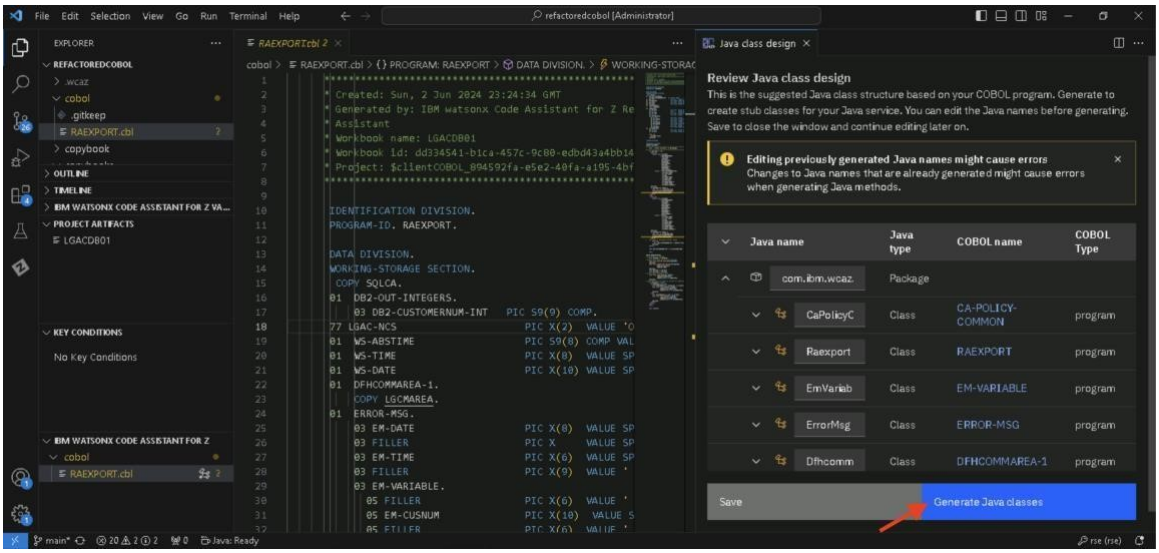
Click on "*Browse for directory"* to select the folder for generated Java classes, as indicated in the diagram below.



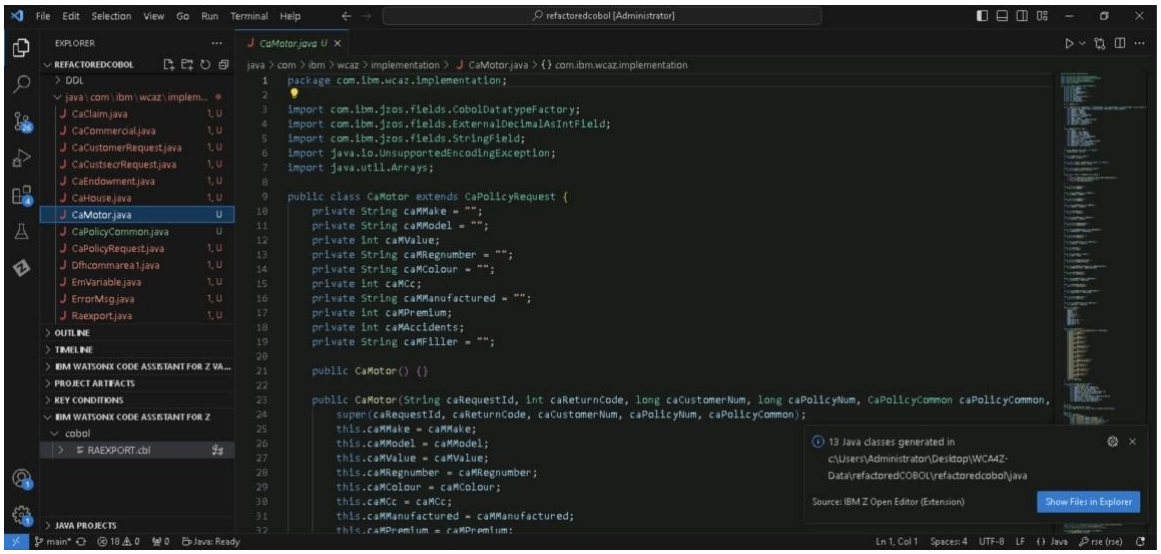
Select the folder *java* from the pop-up window. You can create a new folder and select it, such as *java2*.



The right side of the diagram below shows the *Java class design*. Here, you can review the classes and their attributes and make changes if desired. Notice these classes are derived from the COBOL copybooks and DDLs. Click the *"Generate Java classes"*, as indicated by the red arrow.



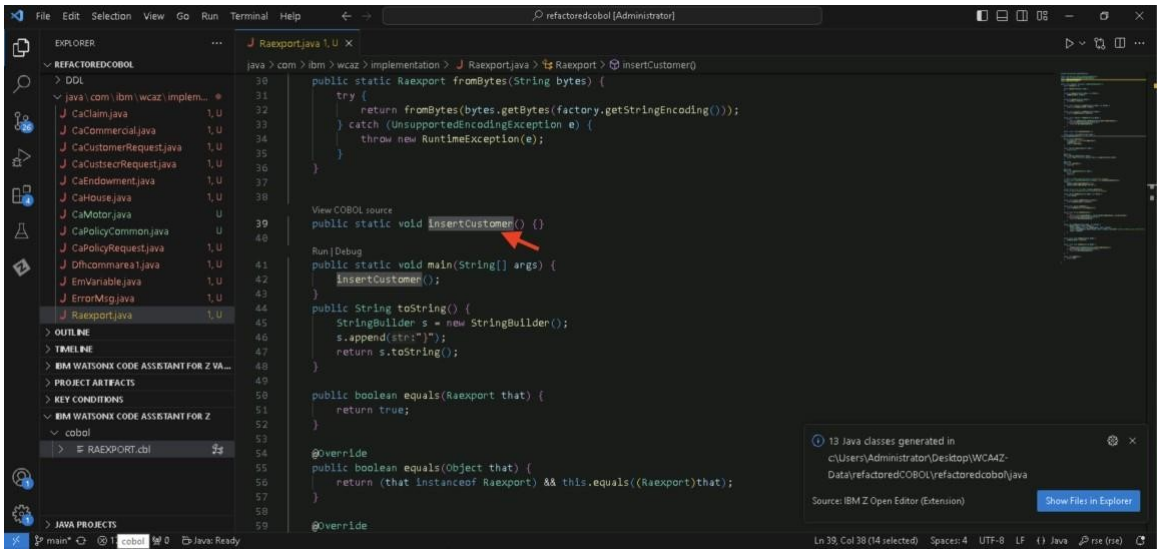
Java classes are generated in the folder *java* (or whatever you selected before). Expand the folder on the *Explorer* pane to see the classes generated. Review the classes and understand the relationships between them.



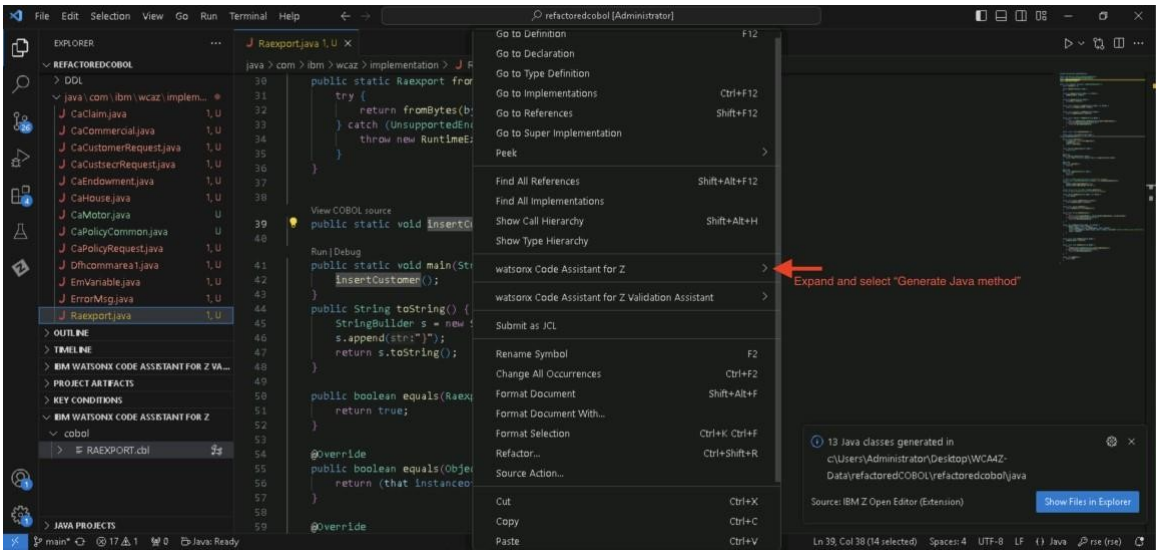
4.4 Step 5 - Generate Java Method

Select and review the Java class that matches the COBOL program name *Raexport.java*.

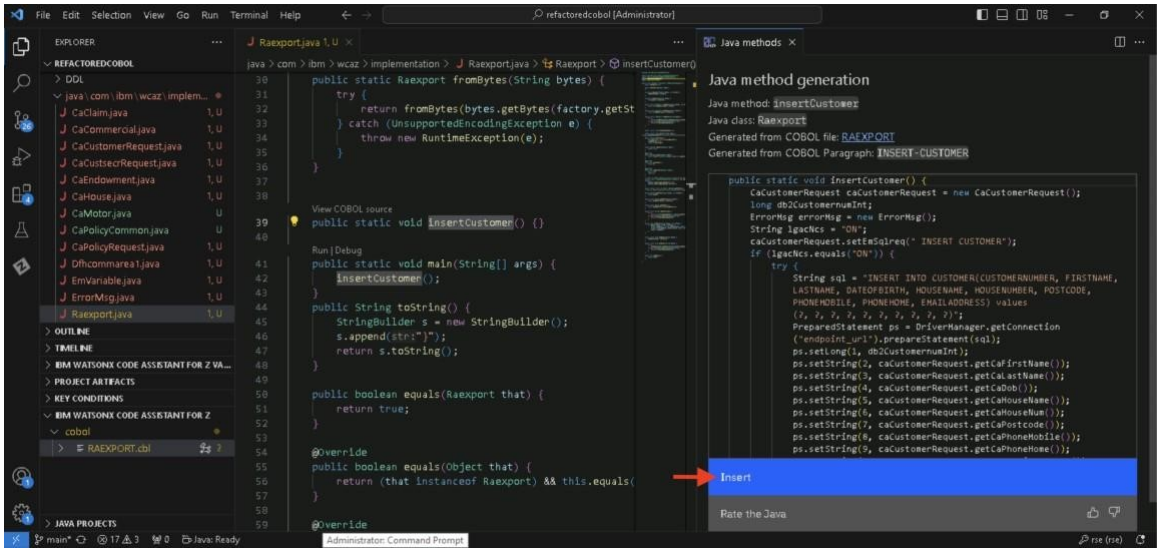
Then, find and select the method *insertCustomer* that you want to generate Java code.

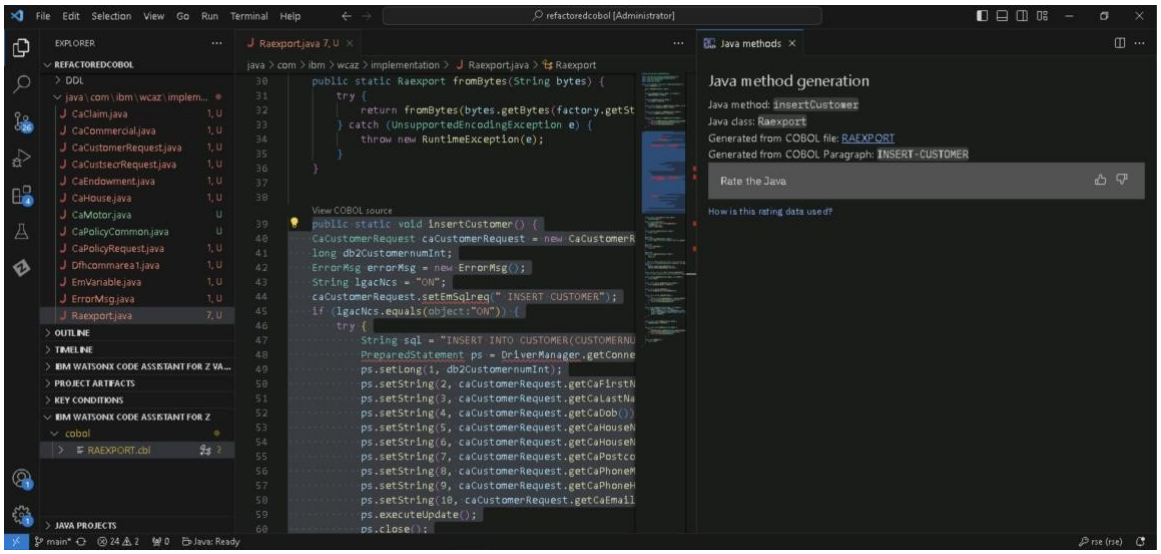


Right-click on the method, select *"watsonx Code Assistant for Z > Generate Java method"*, as indicated in the diagram below.



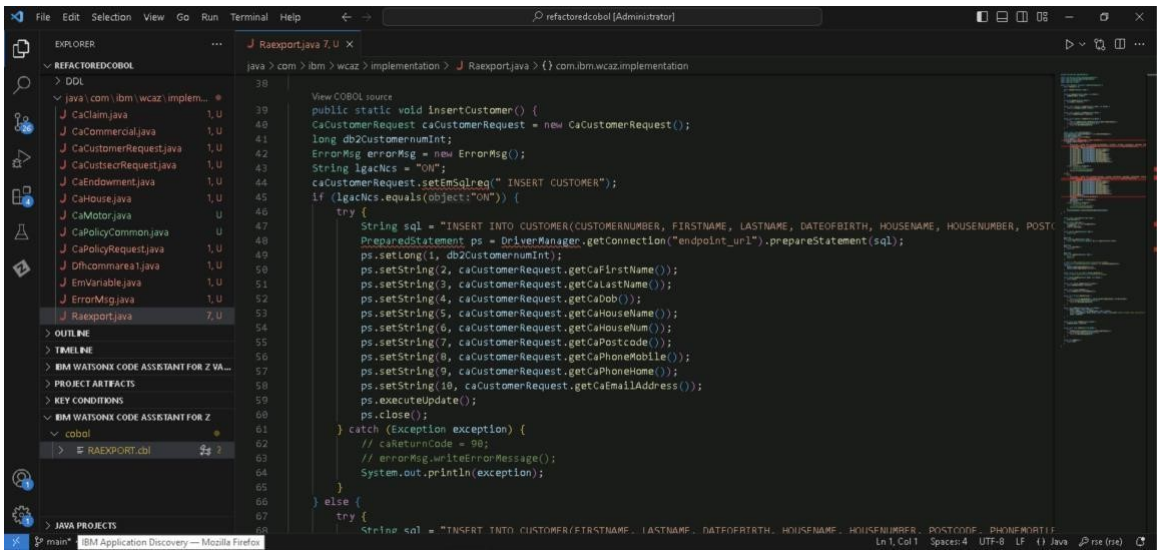
The right-side pane shows the *Java method generation*, as indicated in the diagram below. Here, you can review the code and click *"Insert"* to insert the code into the Java method. This code is generated based on the code in the COBOL program.





4.5 Step 6 – Review and Improve Java Code

Take your time reviewing the code and comparing it with the COBOL code. You may have to fix some of the code to eliminate the errors and refine it to use the specific code library and configurations specific to your application, such as the JDBC driver and URL to the underlying datastore. Notice that the tool is an assistant to the Java developers — it's not meant to replace them completely.



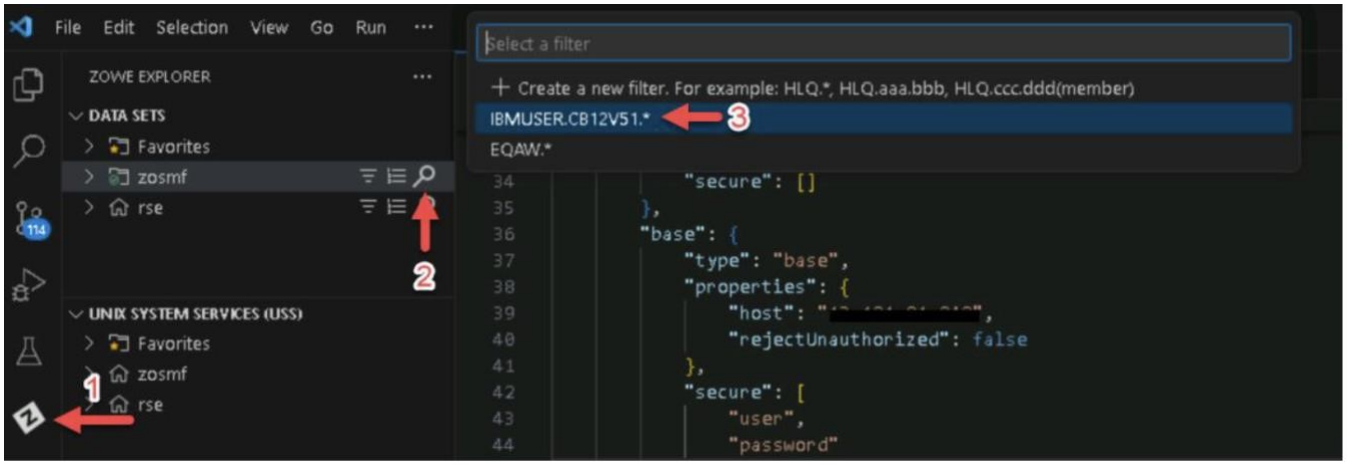
WCA4Z also supports code validation by auto-generating test cases to determine whether the generated Java code is equivalent to the COBOL code.

5 04 Validation

Within VS Code on your ADDI Windows instance, navigate to the ‘ZOWE Explorer’ tab on the left-hand menu

In VS Code, connect to z/OS by clicking on the magnifying glass next to ‘zosmf’ and

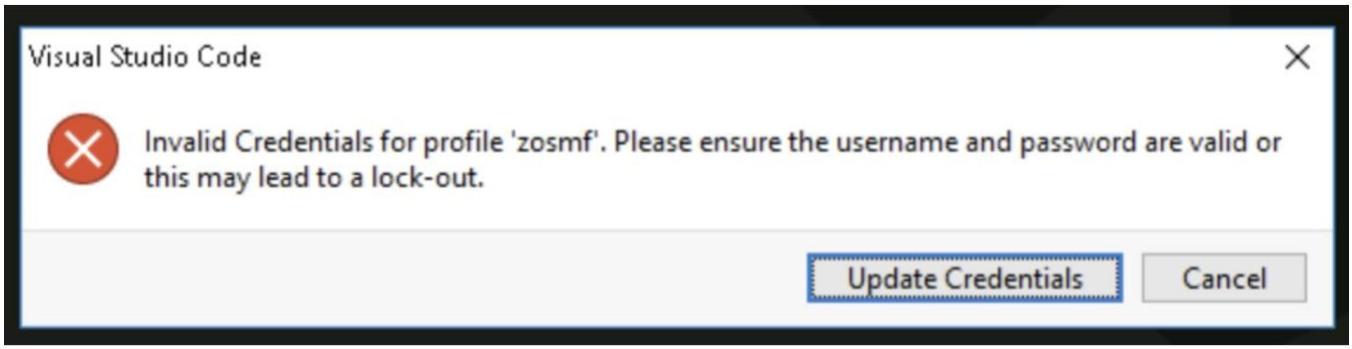
select the filter IBMUSER.CB12V51.* in the Zowe Explorer. If the filter does not exist, then create a new filter.



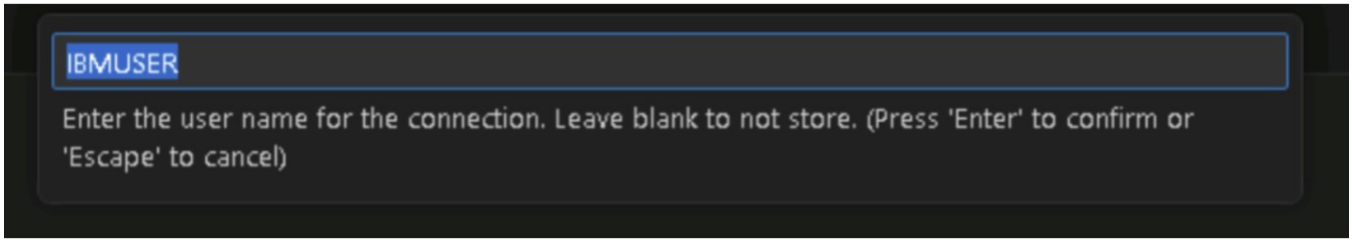
Click enter when you see this at the top of the screen:

A message may then appear related to ‘Invalid Credentials’ for the zosmf profile. Click

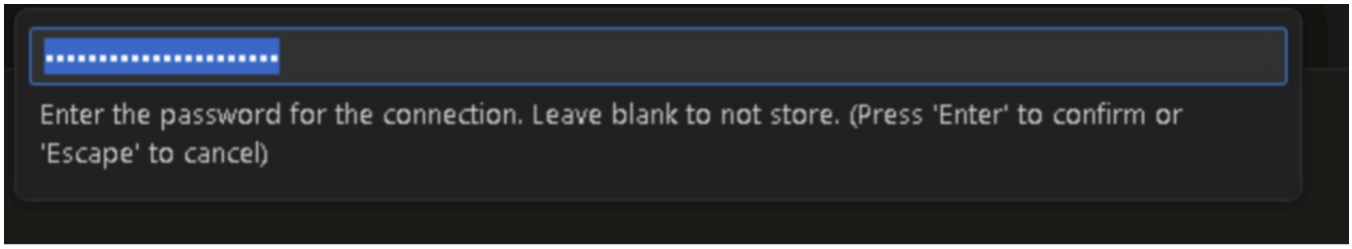
“Update Credentials”:



When prompted for the user name, leave the default “IBMUSER” and click enter:



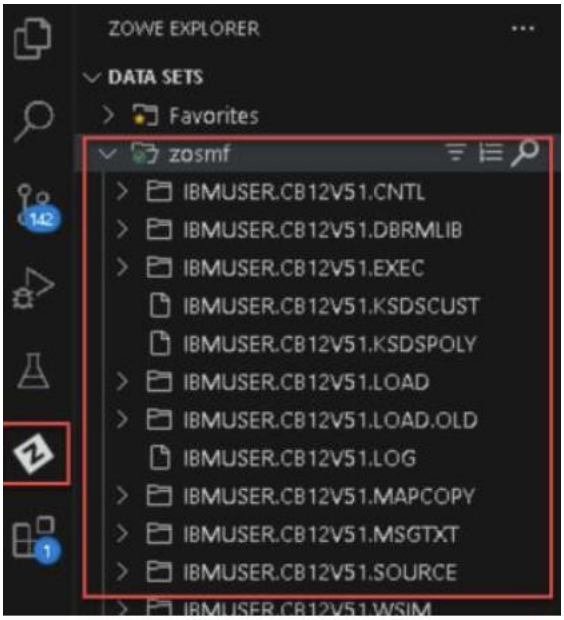
Then enter in the password and click enter-



And lastly, click on the magnifying glass again next to ‘zosmf’ with the filter and hit enter.

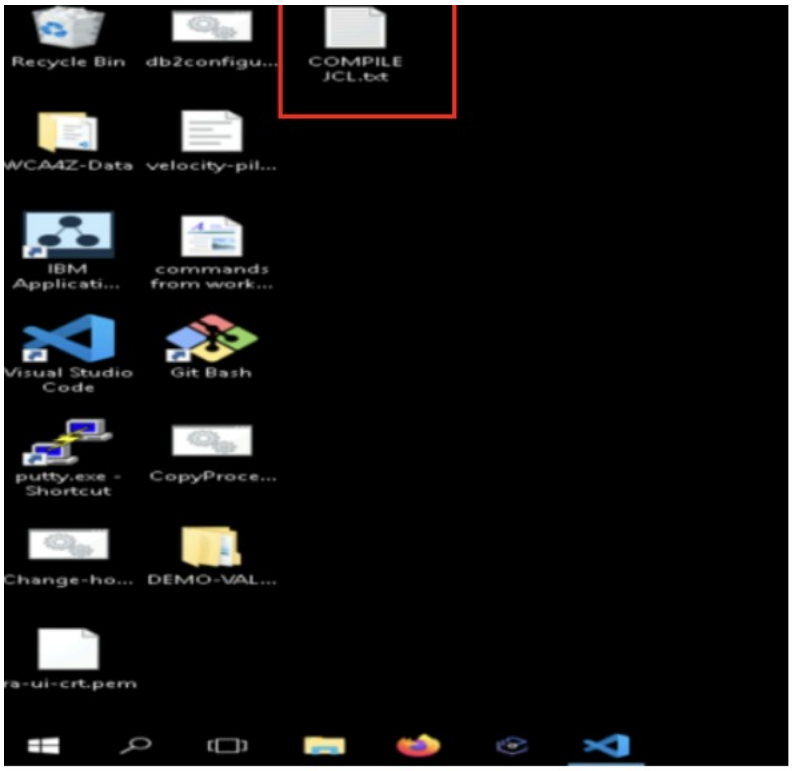
You should see a list of datasets show up here now, if not, please ensure your correctly

entered the passphrase you changed and that you have entered the correct public IP address in the step above OR reselect the filer to IBMUSER.CB12V51.*

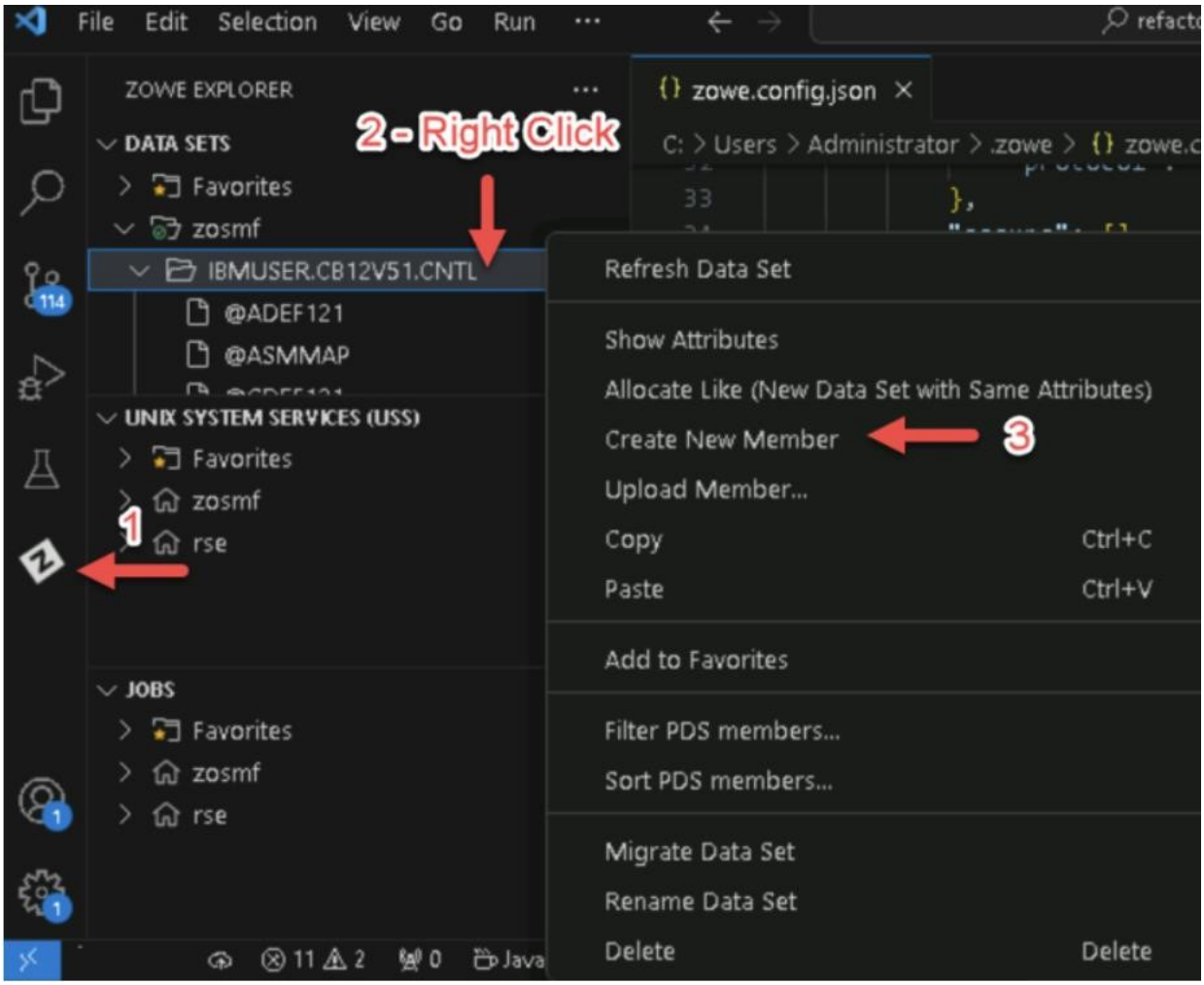


Create a compilation member in IBMUSER.CB12V51.CNTL dataset

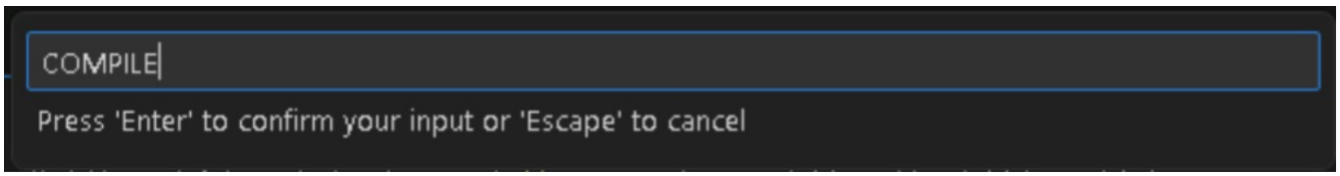
1. Locate the ‘COMPILE JCL.txt’ file on the Desktop in your Windows Image and copy its contents.



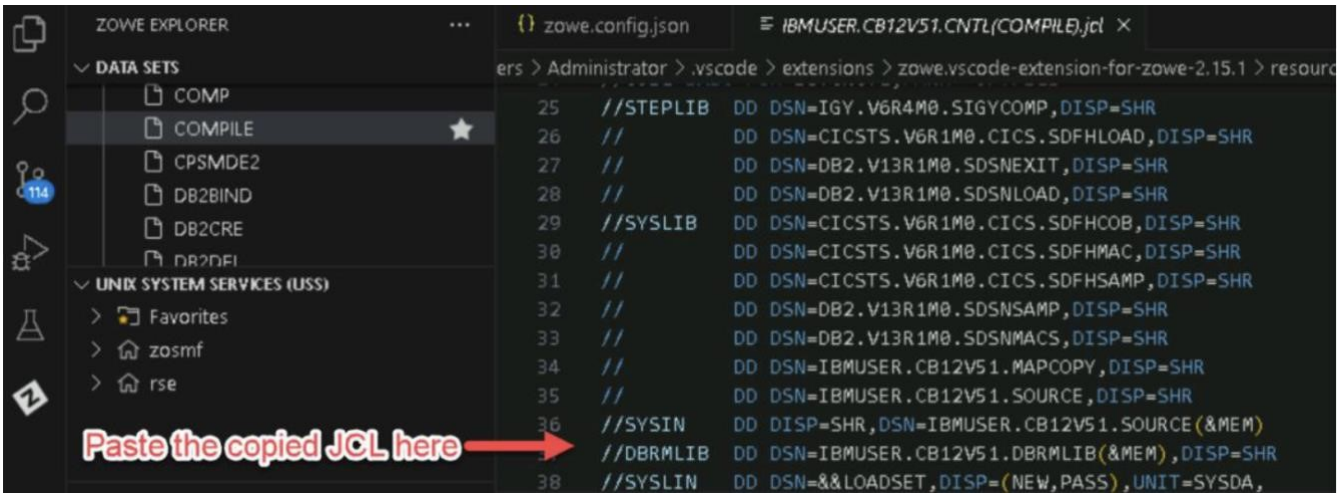
2. Within VS Code, navigate to ZOWE Explorer, right-click on the 'IBMUSER.CB12V51.CNTL' dataset, and then select 'Create New Member':



3.Type ‘COMPILE’ as the name of the new member and hit enter:



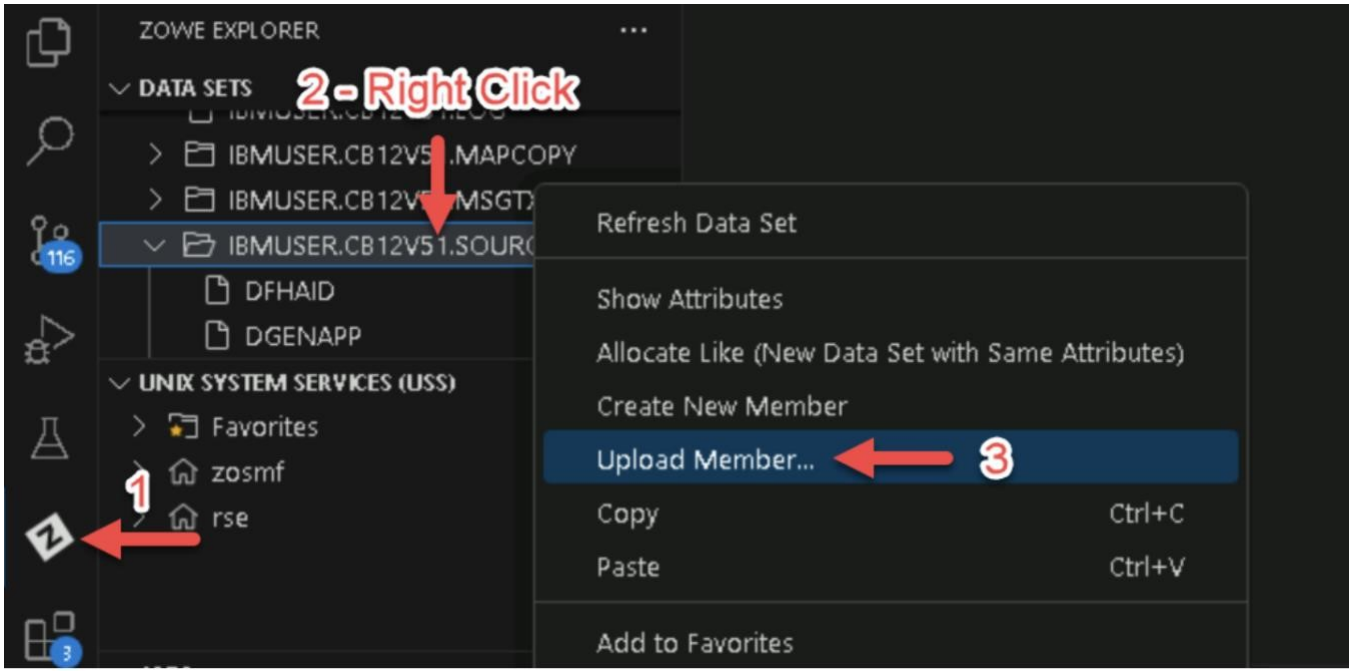
4. Past the copied JCL contents from the desktop into the editor for the COMPILE member and save it:



PLEASE NOTE: This JCL will be used to compile and link-edit your refactored curated COBOL.

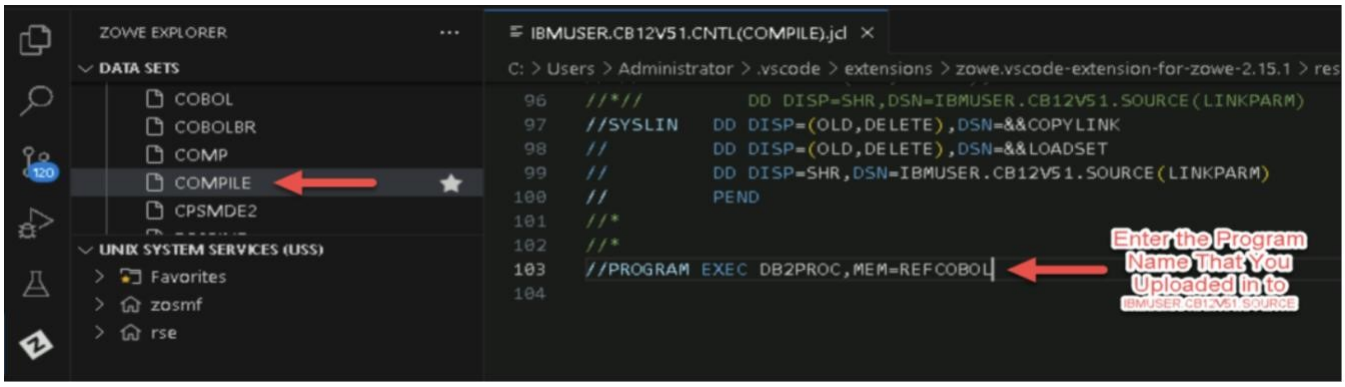
Compile your refactored curated code

1. Locate the IBMUSER.CB12V51.SOURCE dataset and upload your refactored COBOL to it:

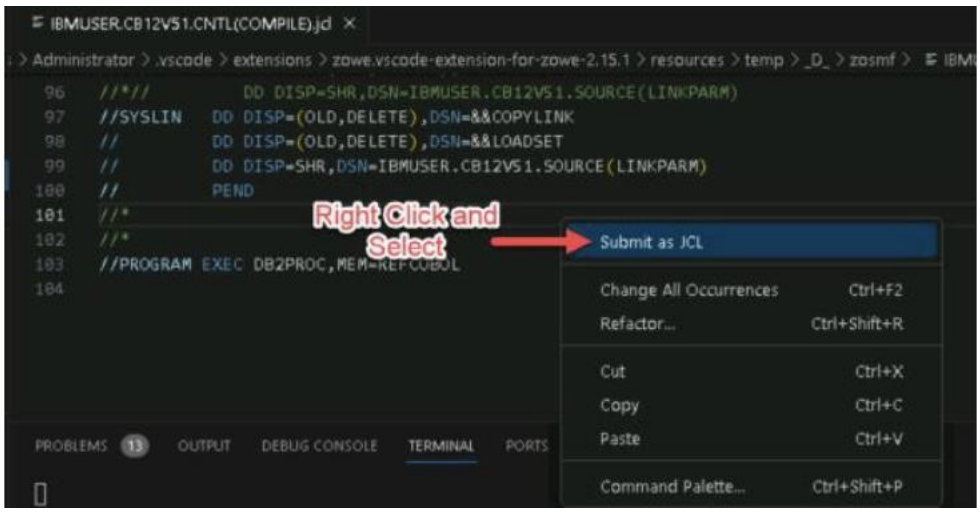


2. Now go to your IBMUSER.CB12V51.CNTL(COMPILE) member and update line 103.
Change MEM= to the refactored program name that you uploaded into

IBMUSER.CB12V51.SOURCE and click save (see below):



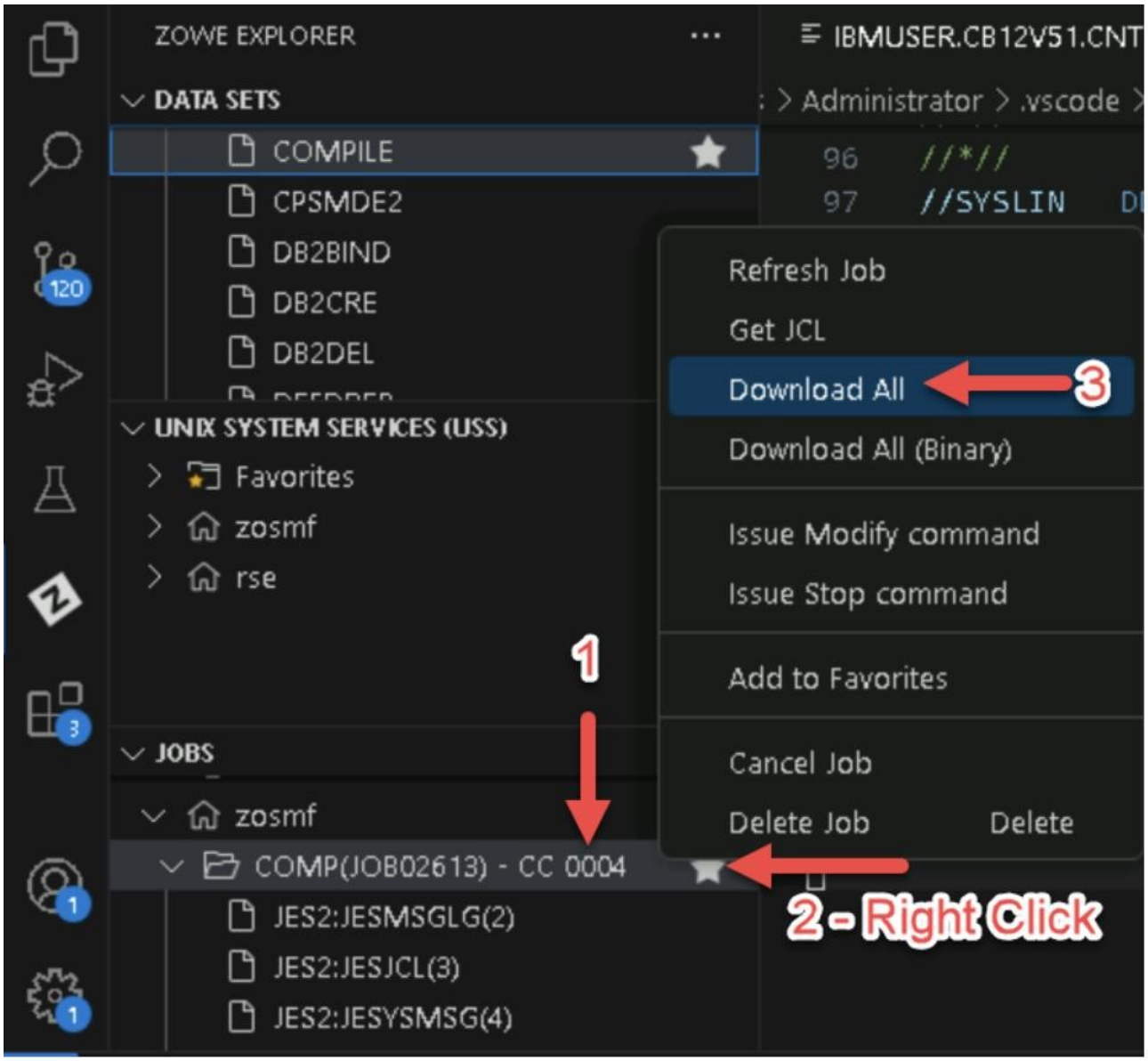
3. After saving, submit the COMPILE JCL member by right-clicking anywhere in the file and select “Submit as JCL”:



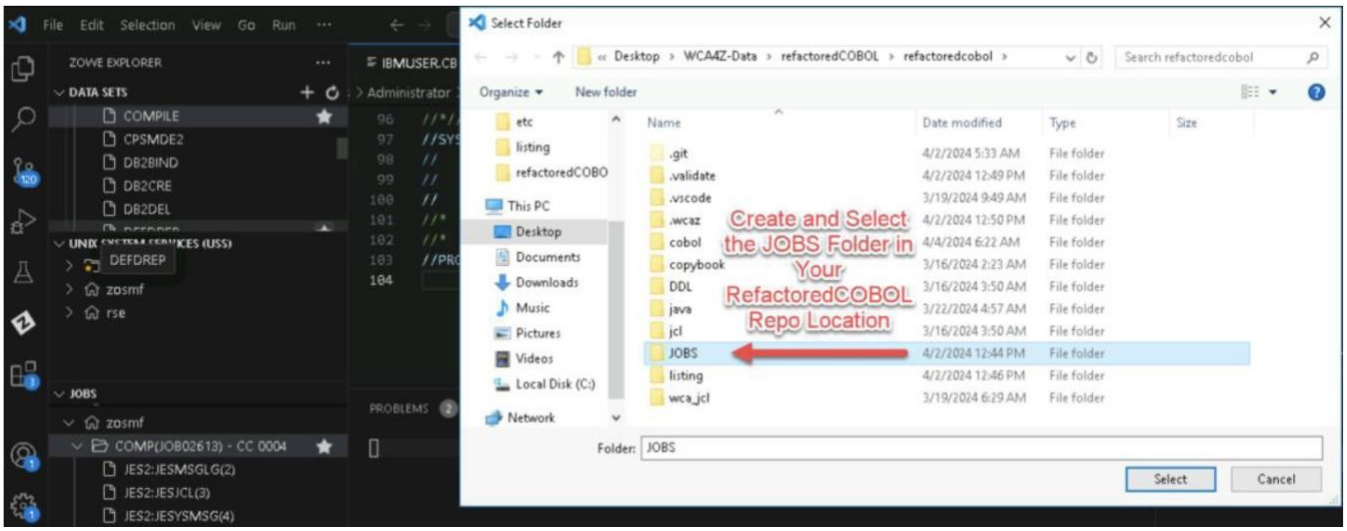
4. After submitting, a message should appear at the bottom right of the screen indicating that the Job was submitted. Click on the Job number:

5. This should take you to the job output where you are expecting to see a return code 0 or 4 (see below as an example). If you don’t see 4 or 0, then you need to review the output and fix it with your clients COBOL developer / resource.

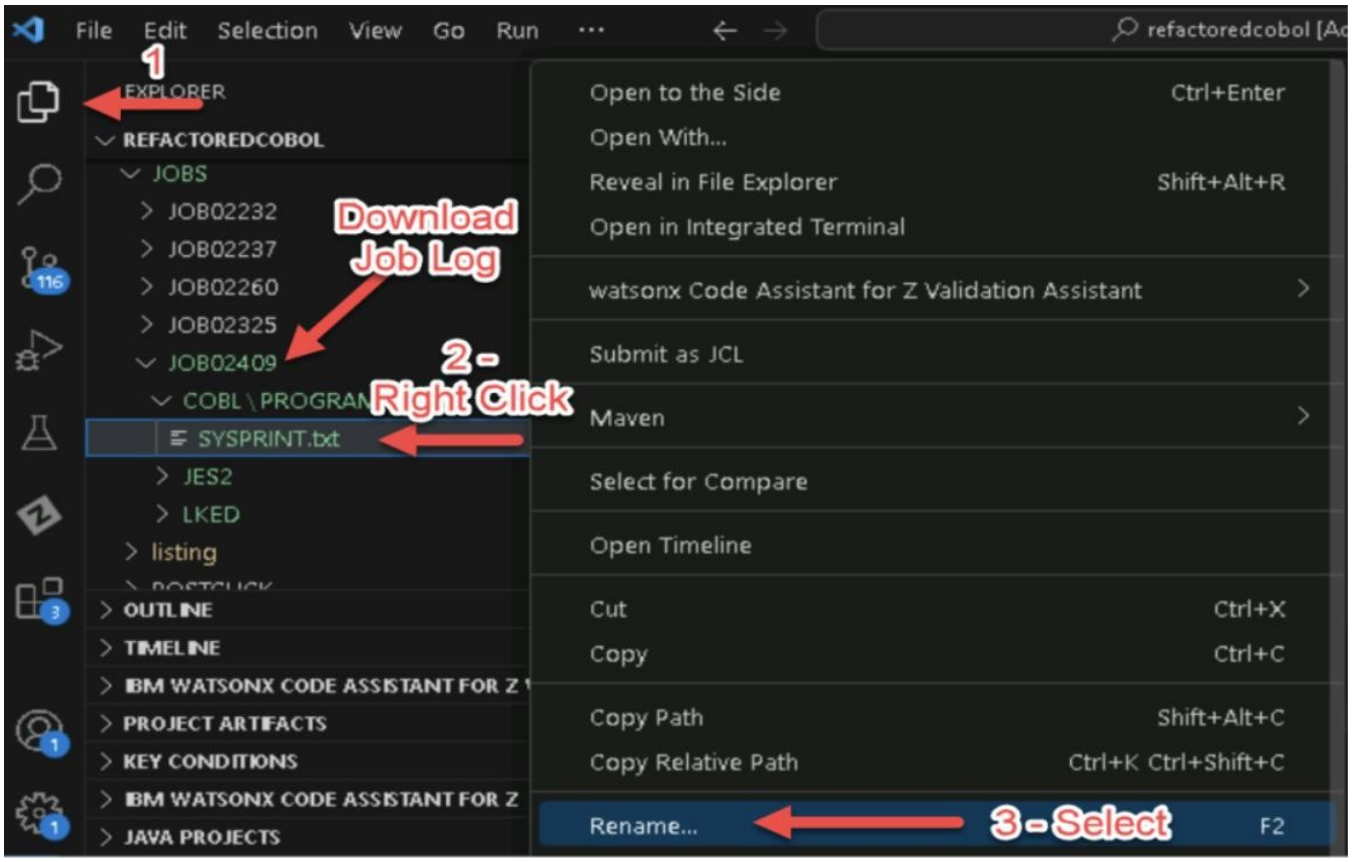
Now right-click on the COMP job output and select “Download All”:



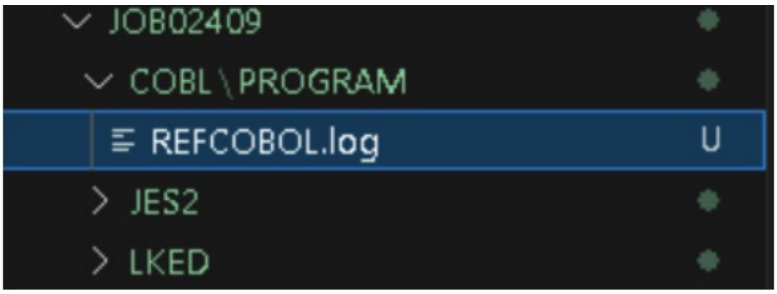
6. Then create and select the 'JOBS' folder in C:\Users\Administrator\Desktop\WCA4Z-Data\refactoredCOBOL\refactoredcobol



7. Within VS Code, click on the Explorer tab to locate the download job log in your JOBS folder in Project Explorer view in VS Code and rename the SYSPRINT.txt file to <your refactored program name>.log

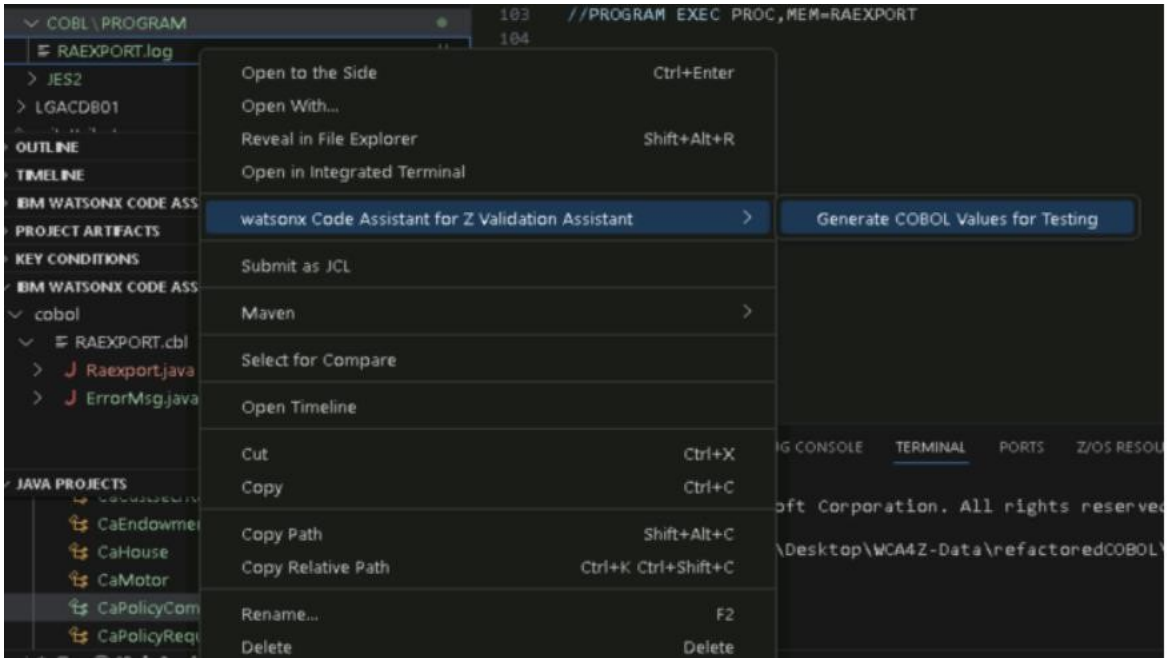


This is an example, you should rename it to your <your refactored program name>.log

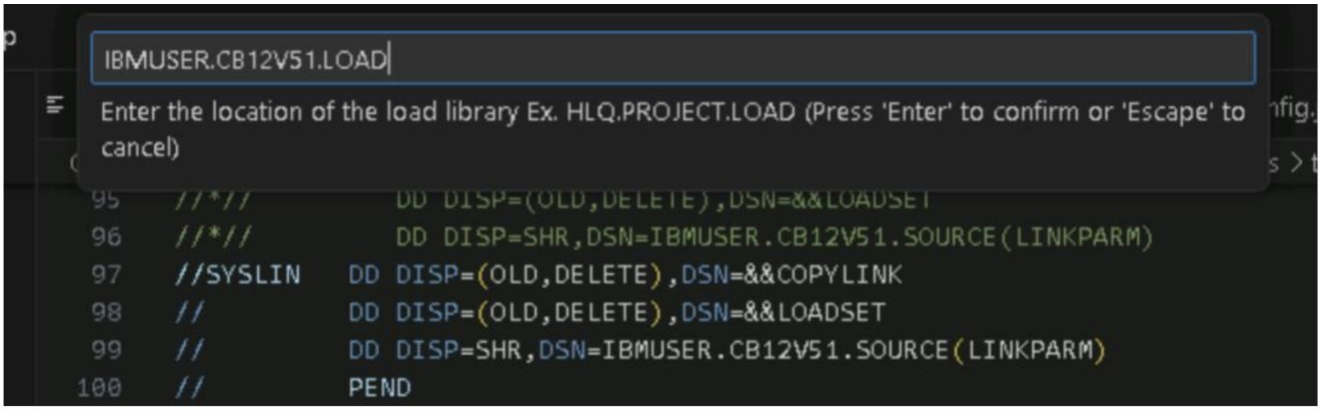


Now right click on the <your refactored program name>.log and select ‘Generate

COBOL Values for Testing’:



You will then be prompted for the location of the load library at the top of the VS Code window. Enter “IBMUSER.CB12V51.LOAD” and hit ENTER:



If you see this error below in the output:Failed to Generate Cobol Outputs: Error:

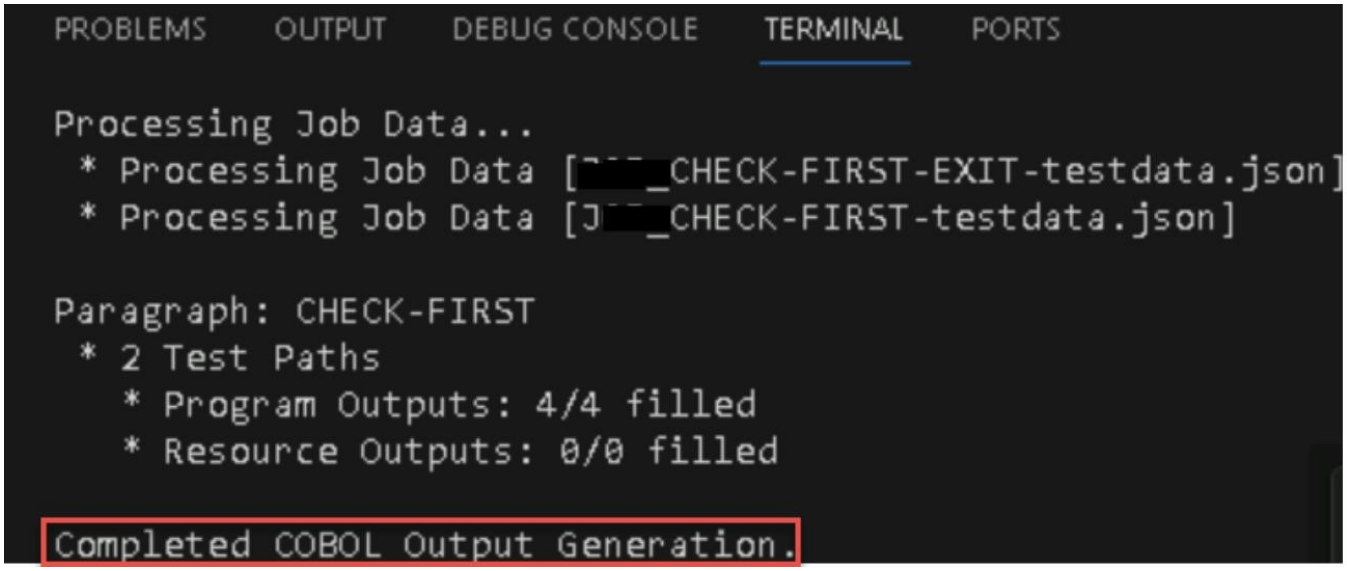
Failed Running Generating COBOL Outputs:

CodeExpectedError: cannot open file:///c%3A/Users/Administrator/Desktop/WCA4Z-Data/refactoredCOBOL/refactoredcobol/.validate/java/debugger/output/jcls/<progra m name>_<paragraph name>-testdata_test_1_out.txt

Then please close VS Code and delete this folder before reopening VS Code and retrying 'Generate COBOL Outputs' above: C:\Users\Administrator\Desktop\WCA4Z-

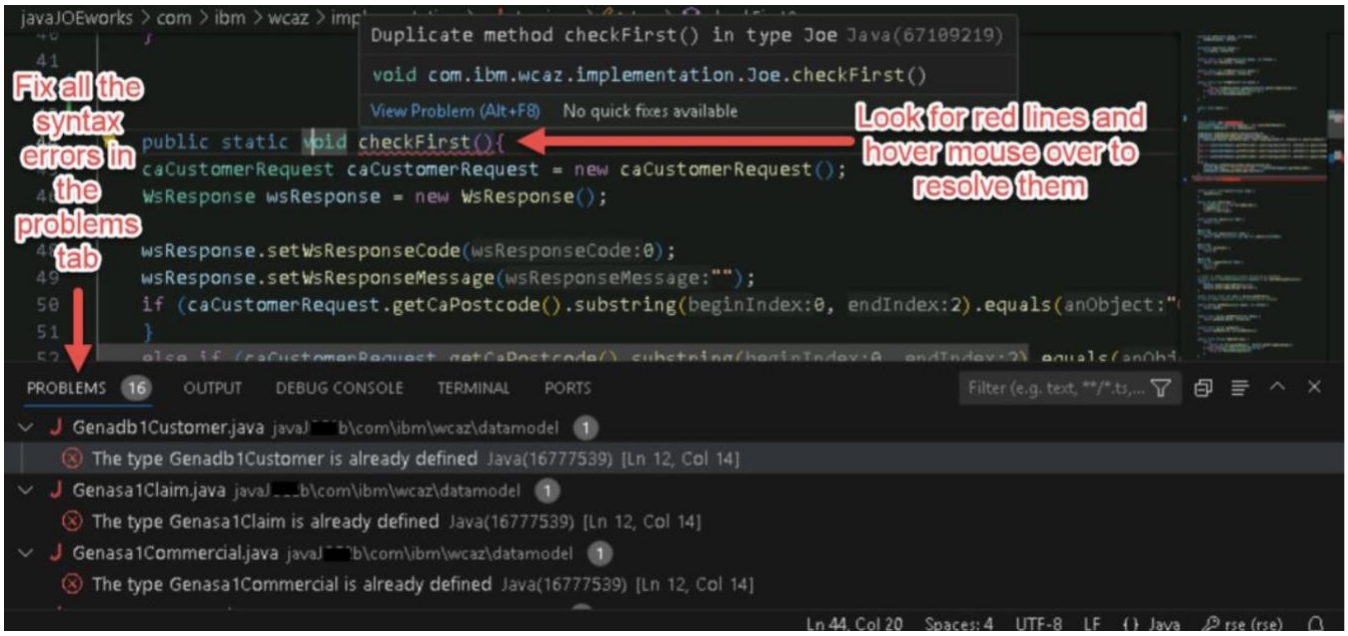
Data\refactoredCOBOL\refactoredcobol\.validate

Ensure you see this output, if you don't, please intently look at the OUTPUT log

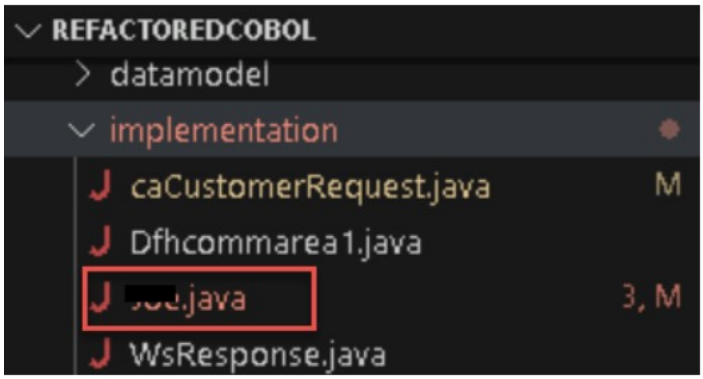


Executing the Equivalence Test

Make sure the transformed Java is compile-able using VS Code

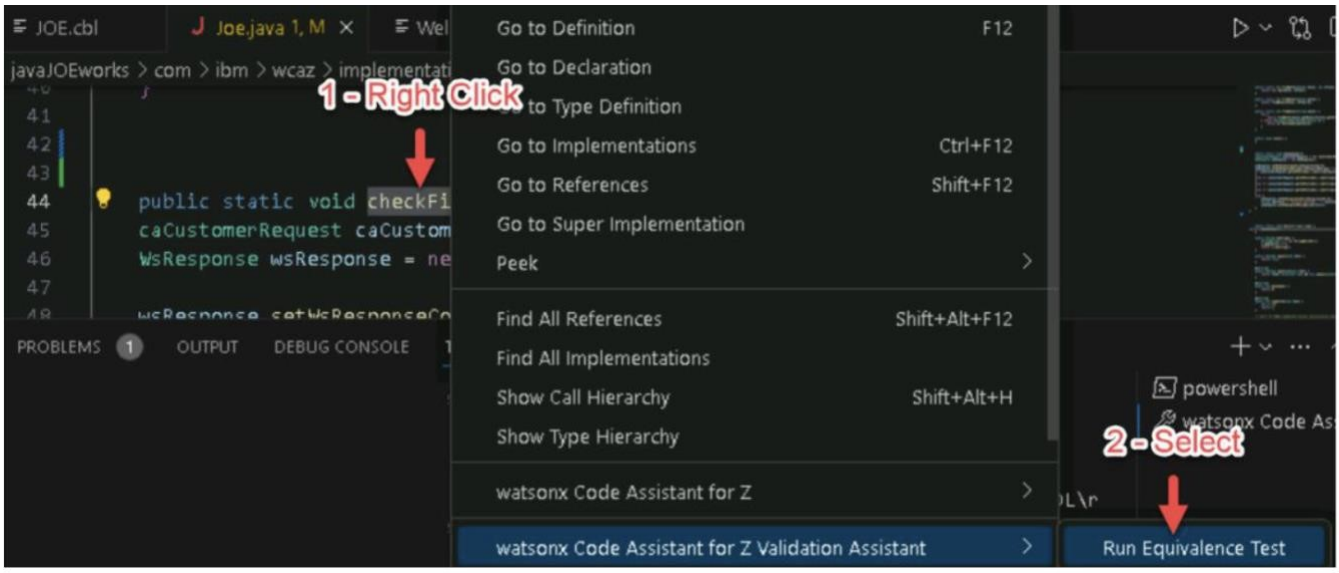


When your Java is highlighted in ‘red’ it likely means your Java has syntax issues and will likely not compile during the Validation phase.

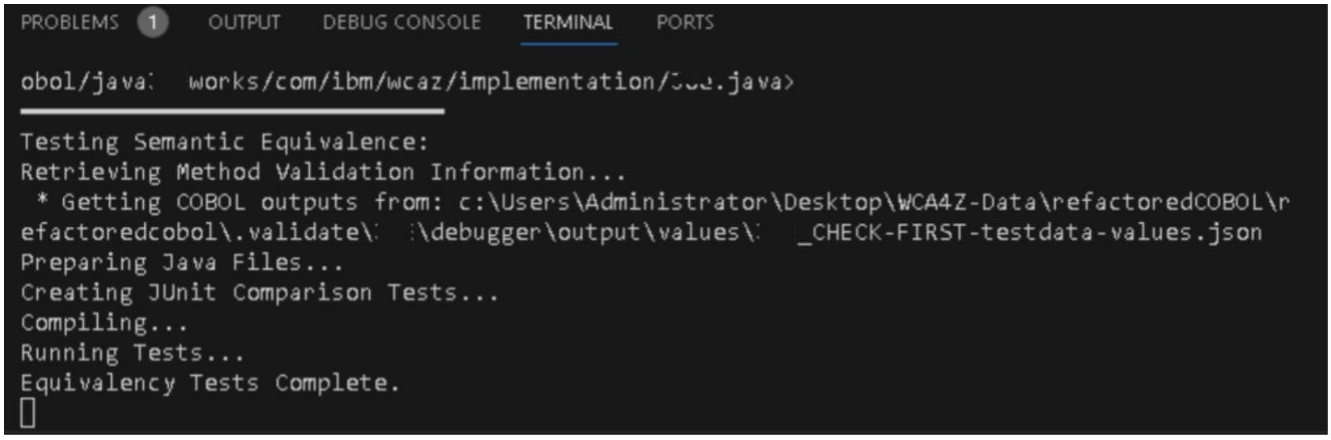


Once your Java has no syntax issues, highlight the relevant Java method which has the

generated Java code, and then run the ‘equivalence test’



Wait for either a successful message like below:



Or if there is a failed execution of the generated J Unit tests, then look at the Terminal output log and attempt to address the issues. ITS VITAL TO ENSURE YOU HAVE A JAVA DEVELOPER WITH YOU FROM YOUR CLIENT TO HELP DIAGNOSE AND VALIITDATE THE ISSUE.

Locate the J Unit test here to see what was generated and then executed against your

Java here: C:\Users\Administrator\Desktop\WCA4Z- Data\refactoredCOBOL\refactoredcobol\.validate\<project name>\generated_code\test\com\ibm\wcaz\implementation