
Lab Exercise: **Using IBM zUnit to Unit Test a COBOL/DB2 batch program**

Session ID: z203780

Speaker Name: Reginaldo Barosa

Room: ?????…

FOR LAB ROOM ONLY -- DO NOT REMOVE, DEFACE or WRITE ON THIS LAB BOOK

# Introduction

This lab will take you through the steps of using the automated unit testing (zUnit) capabilities of IBM Developer for z (IDz) to create a unit test case for a **COBOL/DB2 batch program**. This enables the testing of just a single program using a JCL being executed. This is done by stubbing out DB2 calls, enabling the program to be tested without a DB2 environment being active. This enables early tests without using DB2 and necessary bindings. In this lab you will record interaction with a program via JCL execution and import the recorded data into the unit test case generation process. You will build and run the unit test, make a change to the COBOL/DB2 program, and rerun the test.

## Instructions to use the Windows + Linux + Z/OS on Cloud

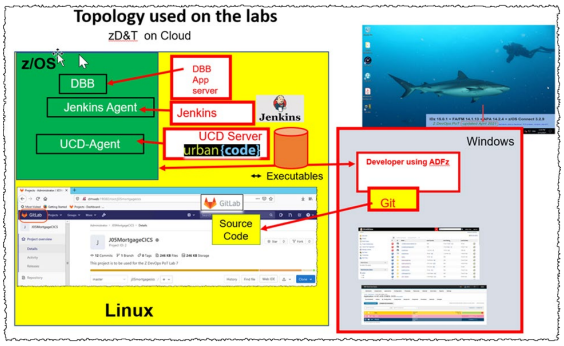
There is 2 ways to access the Windows + z/OS + Linux cloud instances..
1. Via **Windows Remote Desktop** (Better performance, but firewalls might block.
2. Via **Web Browser** (suggested Firefox or Chrome)

In any case be sure that you have the **provided link to access the cloud instance**, the userid and **password** to access the Windows client remotely.
Please always Use COPY/PASTE for password.
Some letters may be identical, example: **I (**uppercase i) and **l** (lowercase L).
Example of provided links

wIBM Z System DevOps Workshop

| Serial No | URL | User ID for Web browser | IP Address for Remote | Password | User ID for Remote Desktop | Domain | machine # |
|---|---|---|---|---|---|---|---|
| 1 | https://169-63-141-246-T-5282.ibmztrialmachines.com/ | Administrator | 169.63.141.246 | KS8E6Pep | Wwin-5282\Administrator | Wwin-528 | 5282 |
| 2 | https://169-60-90-69-T-5283.ibmztrialmachines.com/ | Administrator | 169.60.90.69 | FRthmGL5 | Wwin-5283\Administrator | Wwin-528 | 5283 |

Below is what you will access using the cloud environment.
Be sure that you have an IP address, the userid ( Userid will be different if using **Remote Desktop** or **Browser** ) and password to access the Windows client remotely.

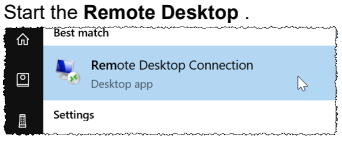


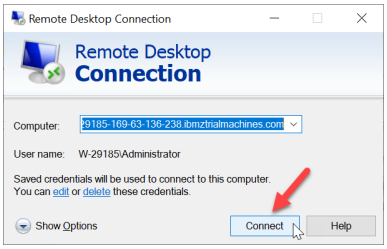
# Using Windows Remote Desktop (preferable way)

This is the preferable way to run the labs, but in some customer location this capability is blocked via Firewalls. If this is your case, use the Web Browser. Instructions are listed here as well..
If you are a Mac user may consider downloading the Remote desktop from
https://itunes.apple.com/us/app/microsoft-remote-desktop-10/id1295203466?mt=12

1. Here one example using Windows 10.
Start the **Remote Desktop** .

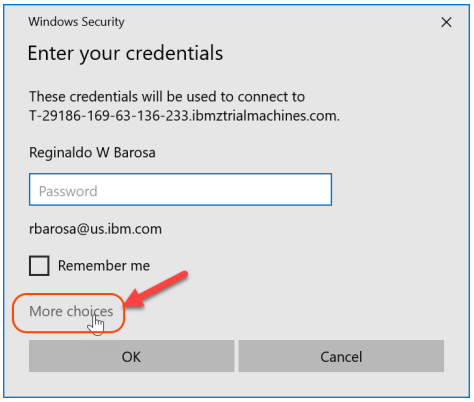



2. Click **Connect**



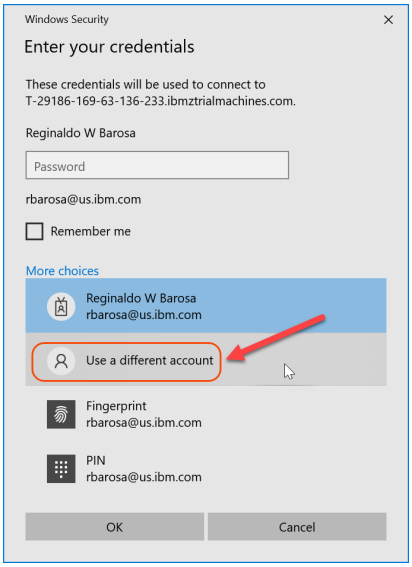

3. Click **Connect** if a dialog as to *trust the remote connection*

4. Click **More Choices**

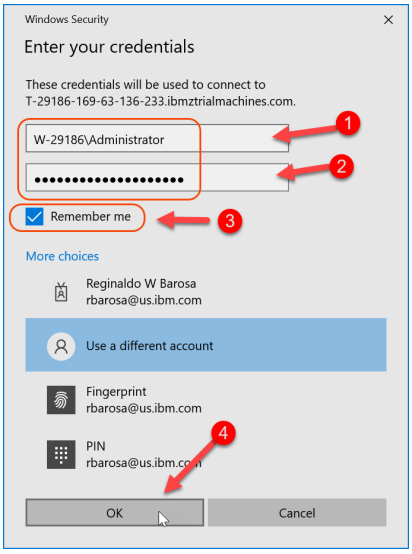


5. Click **Use a different account**

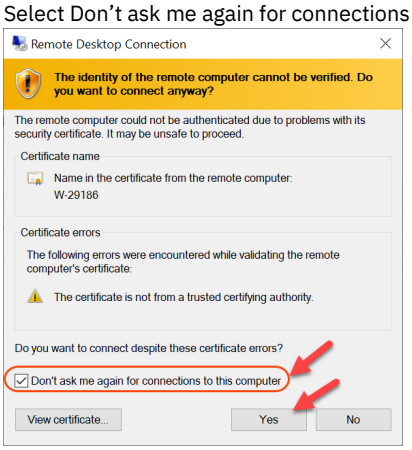



6. Copy and paste the provided "***User ID for Remote Desktop***" and ***Password***, select **Remember me** and click **OK**.

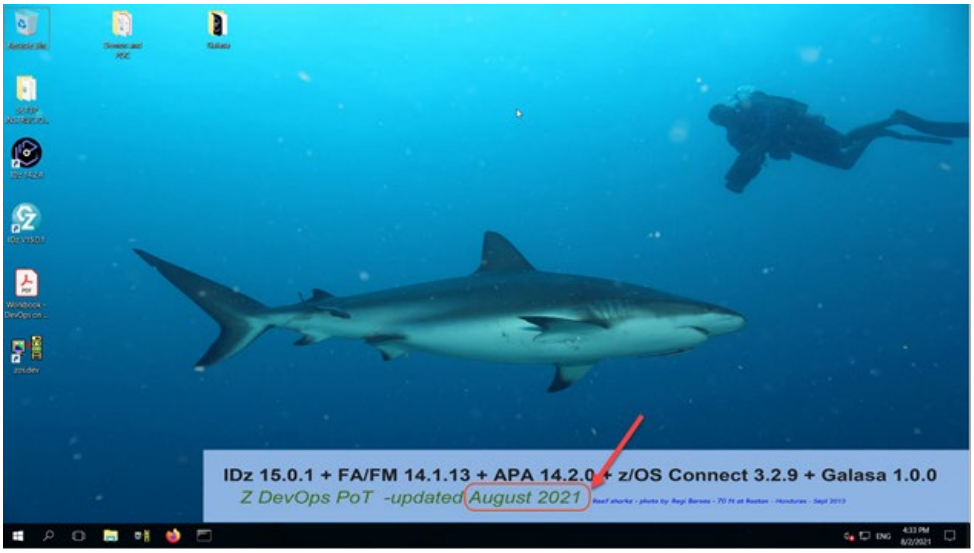


7. You should get a dialog as below..
Select Don't ask me again for connections to this computer and click Yes




8. You should get a screen with a shark. That indicate that you have access to the Windows client on cloud.




On the windows desktop there is a PDF icon with the Labs workbook.. you will find this lab also there.
It may help for copy/paste, etc...
But to follow the labs I suggest printing or use another display or iPhone/iPad to better follow the instructions or use another Monitor to follow the lab instructions.

# Using **Web Browser** (if using VPN or Firewalls that block Remote Desktop)

**Commented [RWB1]:**

Be sure that you have the **provided link to access the cloud instance**, the userid (**Administrator**) and **password** to access the Windows client remotely. Always use COPY/PASTE for password.

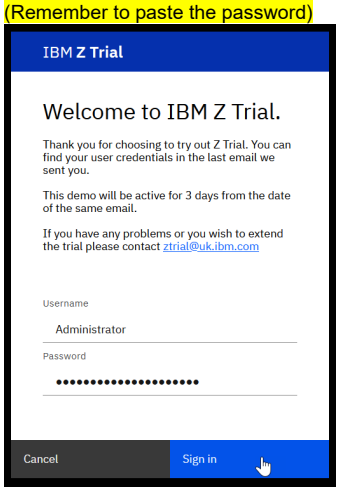
Some letters may be identical, example: **I (**uppercase i) and **l** (lowercase L).

Example

wIBM Z System DevOps Workshop

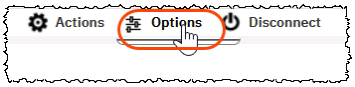
| Serial No | URL | User ID for Web browser | IP Address for Remote | Password | User ID for Remote Desktop | Domain | machine # |
|---|---|---|---|---|---|---|---|
| 1 | https://169-63-141-246-T-5282.ibmztrialmachines.com/ | Administrator | 169.63.141.246 | KS8E6Pep | Wwin-5282\Administrator | Wwin-528 | 5282 |
| 2 | https://169-60-90-69-T-5283.ibmztrialmachines.com/ | Administrator | 169.60.90.69 | FRthmGL5 | Wwin-5283\Administrator | Wwin-528 | 5283 |

1. Use the link provided by the instructor and type **Administrator/password** provided and click **Sign in** (Remember to paste the password)




**2.** Click on the **small mark** as seen below


3. Click on **Options**



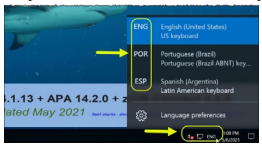

And select **Full Screen**

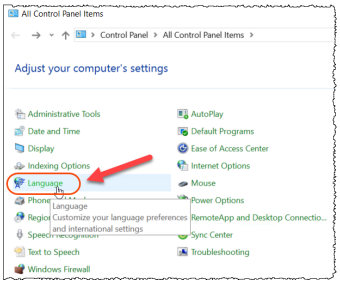
# Adjusting the keyboard for other languages than English

In some countries the keyboard must be mapped due language differences.
If you are in Brasil or any other Latin America country can select the language as below

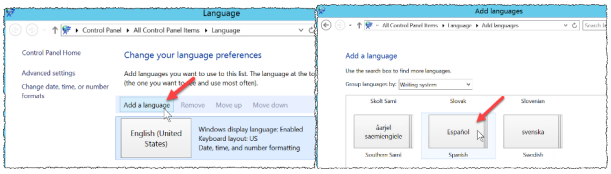



If you use another keyboard other than ENG, POR or ESP, you need to make updates..

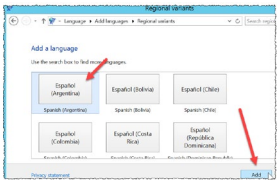
1. Go to windows **Control Panel** and select **Language**



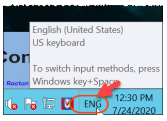

2. Click **Add a Language**, and follow the dialogs. See one example on the screen captures below




Click **Add**




To switch the keyboard, just select the desired language on the right corner of the screen




and select the language

# Exercise instructions

## Overview of development tasks

To complete this tutorial, you will perform the following tasks:

**PART #1 – Unit test on COBOL/DB2 program DB2BATCH and introduce a bug**

1. **Run the COBOL/DB2 batch program using JCL**
   → You will submit a JCL to execute a COBOL/DB2 program that calls a COBOL subprogram to print a small report.
2. **Use zUnit to record the batch execution.**
   → You will record the batch execution using the COBOL/DB2 program and subprograms.
3. **Generate, build, and run the unit test generated program**
   → You will compile and link-edit the generated unit test program, followed by running the unit test.
4. **Modify the COBOL/DB2 program (introduce a bug) and rerun the unit test**
   → You will modify the COBOL/DB2 program introduce a bug, rerun the unit test, and observe the failure of the test case.

**PART #2 – Fix COBOL/DB2 program DB2BATCH and re-run the Unit test**

5. **Run the batch program using the provided JCL and verify the bug .**
   → You will run the Batch JCL and observe the bug on the printed report..
6. **Use IDz to fix the bug, recompile the COBOL/DB2 program**
   → The bug is fixed using IDz, program fixed is rebuilt
7. **Rerun the zUnit and verify that the bug is eliminated**
   → When the zUnit test case is executed you can verify that the program is fixed.

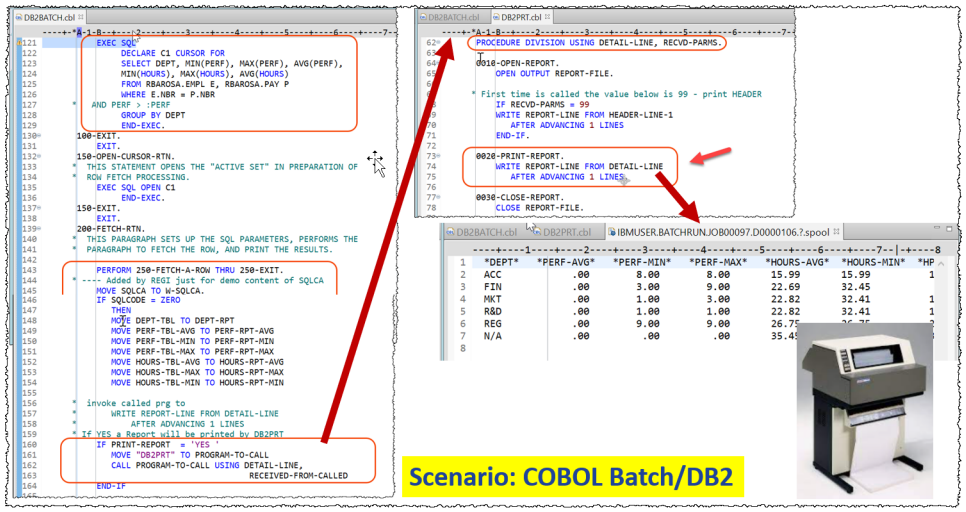
 

# PART #1 – Unit test on program DB2BATCH and introduce a bug

## Section 1. Run the COBOL/DB2 batch program using JCL

You will submit a provided JCL to execute on z/OS and verify the report produced by the second program that is invoked by a dynamic call.

Below it is showing the COBOL program DB2BATCH invoking the program DB2PRT that prints the report




### ___ 1.0 Connect to z/OS using IDz

1.0.1 Start *IBM Developer for z Systems version 15* if it is not already started otherwise go to step 1.1.1

⏭ Using the desktop double click on **IDz V15** icon.

⏭ Verify that the message indicates that it is Version **15.0.1**

**IMPORTANT** -> This icon will start an eclipse workspace that has already some definitions required for this lab. PLEASE DO NOT start IDz using other way than this icon.

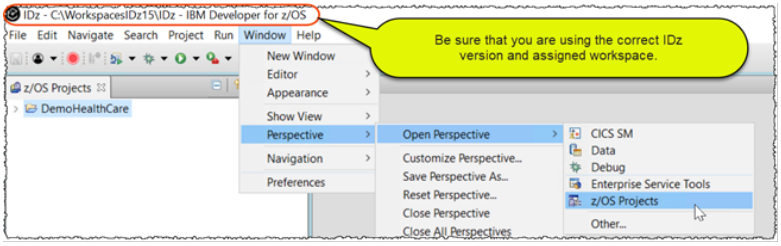


## ___ 1.1 Submit a provided JCL for execution

You will use IDz to submit a provided JCL .

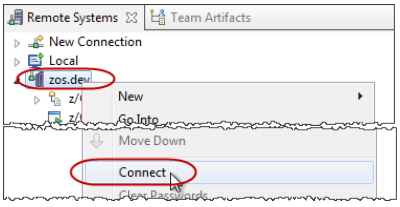
1.1.1 ⏭ Open the **z/OS Projects** perspective by selecting
**Window > Perspective > Open Perspective > z/OS Projects**



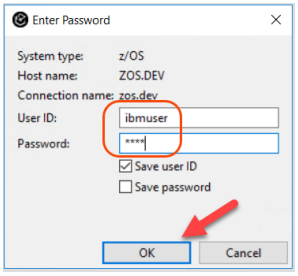

Nothing will happen you are already at this [perspective.

1.1.2 On this lab you will use userid **ibmuser** . and password **sys1**.
If you are connected as *ibmuser*, jump to step 1.1.4 otherwise.

⏭ Using *Remote Systems* view, right click on **zos.dev** and select **Disconnect**
and then right click on **zos.dev** again and select **Connect**

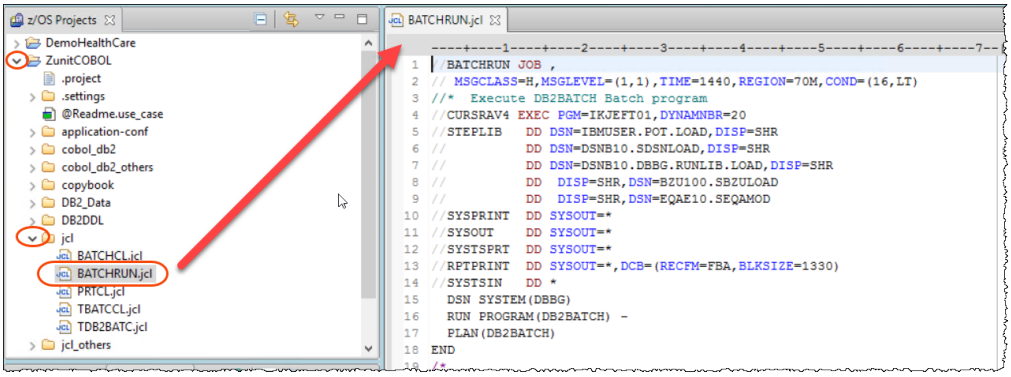



1.1.3 ⏭ Type **ibmuser** as userid and **sys1** as password.
The userid and password can be any case; don't worry about having it in UPPER case.
Click **OK** to connect to z/OS.

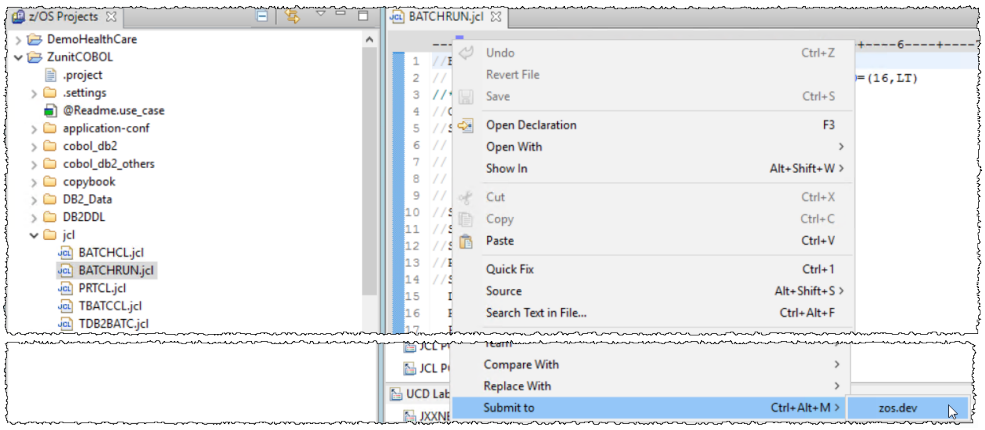


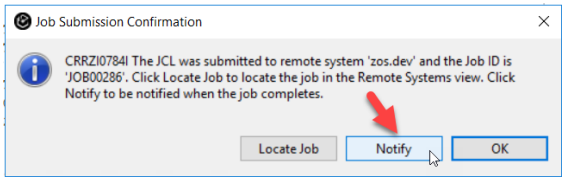
1.1.4 ⏭ Under **ZunitCOBOL** expand **jcl** and double click on **BATCHRUN.jcl** to edit the JCL that will be submitted for execution.




1.1.5 ⏭ Right click on the JCL edited and select **Submit to > zos.dev**

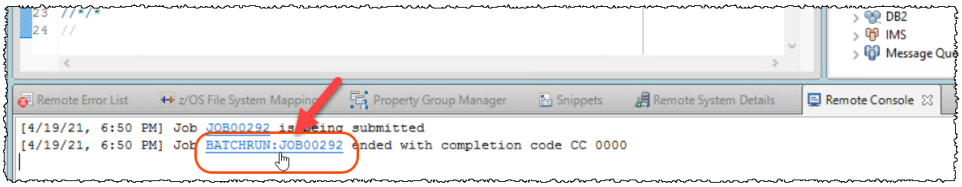



1.1.6 ⏭ Click **Notify** to be notified when the execution is complete.



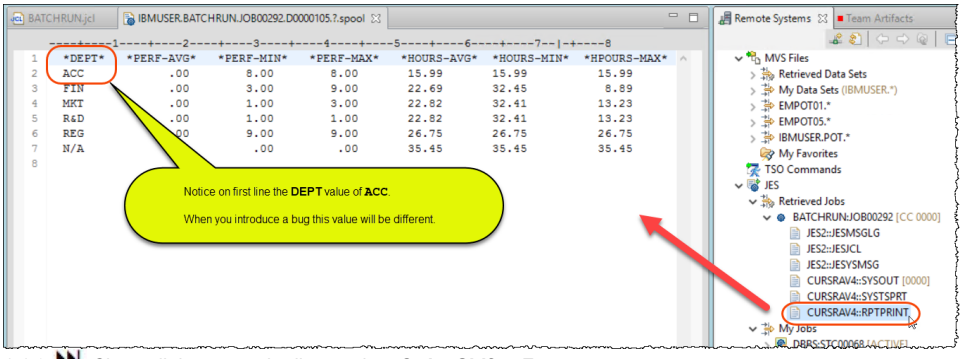

 

1.1.7 Under *Remote Console*, you will be notified when execution is completed.
⏭ Once the execution ends, **click** on the link **BATCHRUN:JOB00xxx**
(where 00x**xx** is a number) to point to the Job output. This number vary depending on z/OS.

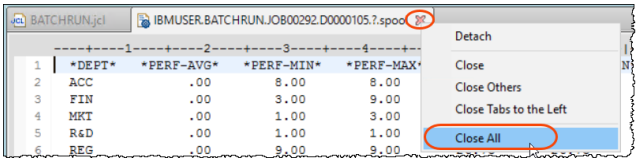



1.1.8 ⏭ Under *Remote Systems* view **scroll down**, expand **JES > Retrieved Jobs > BATCHRUN:JOB00xxx** and **double click CURSRAV4::RPTPRINT** step
and you will see the report produced by the ***DB2PRT*** called COBOL subprogram.

Tip: If there is no jobs under "*Retrieved Jobs*", is because you did not click on link as stated at 1.1.7.
You can also see this output once you right click on "*My Jobs*" under *JES* and select ***Refresh***.




1.1.9 ⏭ Close all the opened editors using **Ctrl + Shift + F4,**
Or right click on the ⊠. and choose **Close all.**




**What have you done so far?**

You submitted a JOB to be executed under batch. This job uses two subprograms being invoked.
One of the programs invoked dynamically prints a small report from data retrieved from a DB2 table.

## Section 2 – Use zUnit to record the batch execution.

Using IDz you will record the COBOL/DB2 batch execution via JCL.

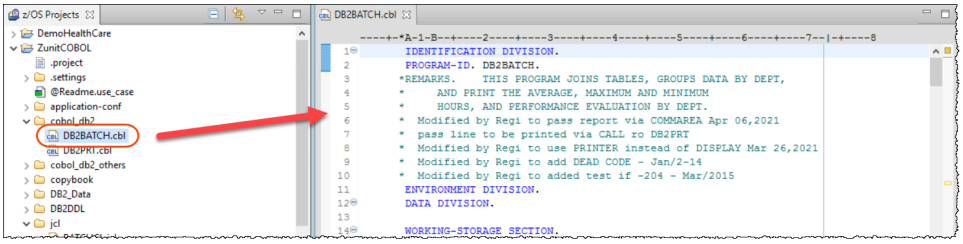
The main COBOL program (**DB2BATCH**) reads from a DB2 table and pass some data to be printed by another dynamically called COBOL subprogram (**DB2PRT**).

Notice that the main COBOL program also invokes a third COBOL program (**REGI0C**) using a static call.

## ___ 2.1 Understanding the main COBOL program that reads from DB2 table

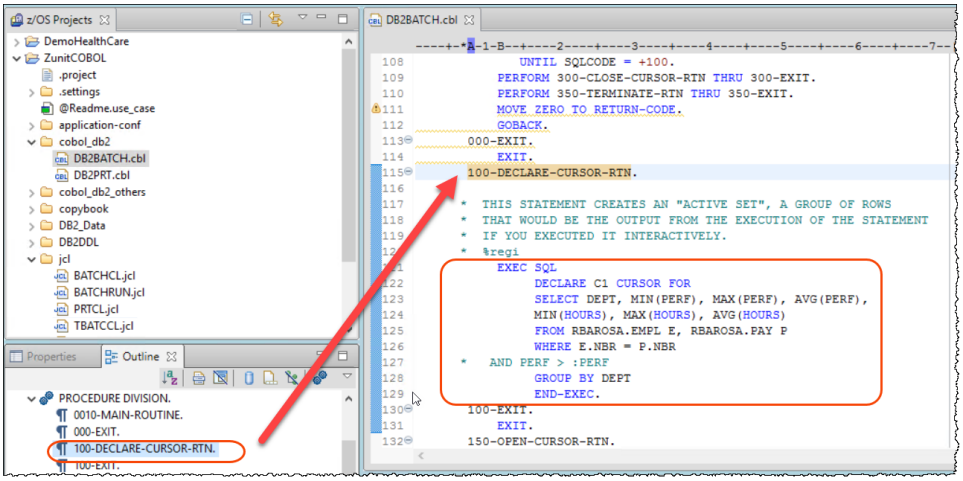
The main COBOL code that reads the DB2 tables is the program ***DB2BATCH***

2.1.1 ⏭ Using *z/OS Projects* view
double click on **DB2BATCH.cbl** under **ZunitCOBOL/cobol_db2.**
This is the program that you will update later on..




2.1.2 ⏭ Using the *Outline view* on left expand **PROCEDURE DIVISION** and
click on **100-DECLARE-CURSOR-RTN.**

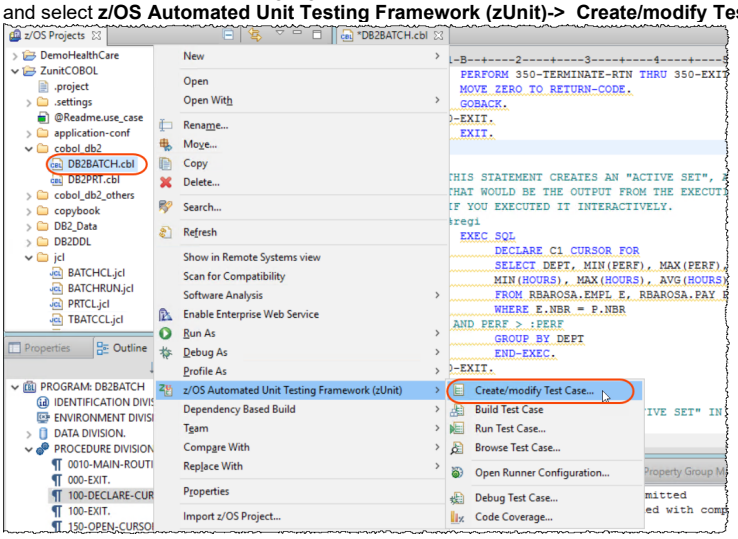
This is where the DB2 select statement is defined. Notice that the program will execute a DB2 table join and scan the rows resulting from that join. Later on you will introduce a bug in this program.



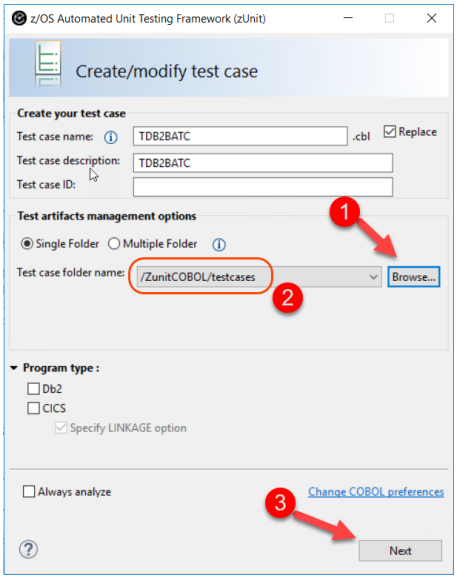

 

## ___ 2.2 Recording the batch JCL execution

2.2.1 ⏭ To start the recording, right click on **DB2BATCH.cbl**
and select **z/OS Automated Unit Testing Framework (zUnit)-> Create/modify Test Case.**.

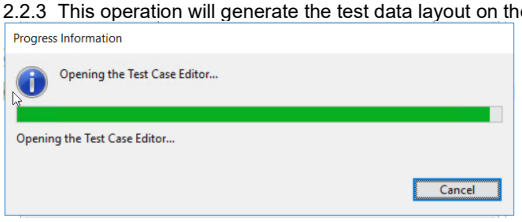



2.2.2 This opens a dialog where you can name your test case.
⏭ Using the Browse button select **/ZunitCOBOL/testcases** as folder name and click **Next**.

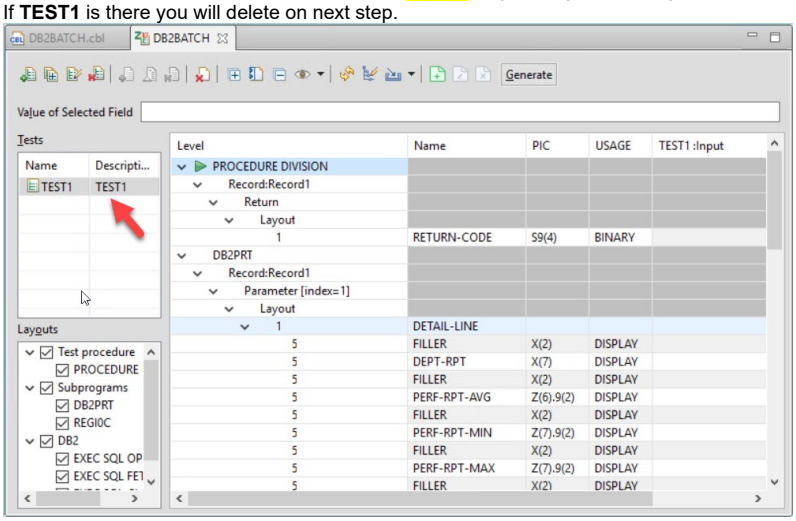


2.2.3 This operation will generate the test data layout on the local workspace and open the *Test Case Editor.*

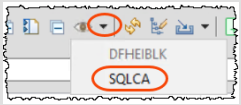



2.2.4 The *Test Case Editor*, as shown below. TEST1 may or may not be on your screen.
If **TEST1** is there you will delete on next step.




### Understanding the *test case editor*

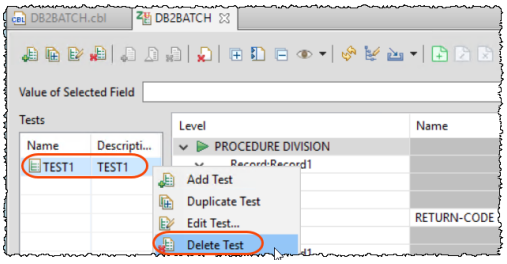
8. The bottom left box summarizes all the input output variable structures – In our exercise, the main COBOL program(*DB2BATCH*) has 2 Subprograms (*DB2PRT* and *REGI0C* ) and the LINKAGE Section of those programs are displayed. Also the areas used by DB2 statements are listed.
9. The DB2 SQLCA area can be displayed once is selected as below.



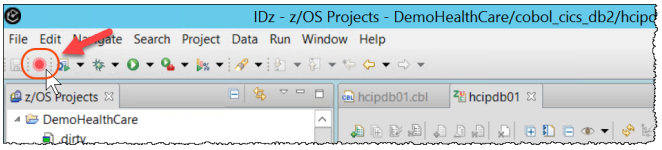

The test entry Input and Expected output columns represent the flow of data into and out of the main program, that is, the program being tested by zUnit. When data is added to these columns in a subroutine, the flow of data is:
***Input:*** data passed into the main program: that is, what is passed to the main program when the subroutine completes execution.
***Expected output:*** data passed out of the main program: that is, what is passed to the subroutine when it is called by the main program.

2.2.5 ⏭ Since we will be recording the program execution, delete the **TEST1** or any other entry (if it exists) by right clicking and selecting **Delete Test**

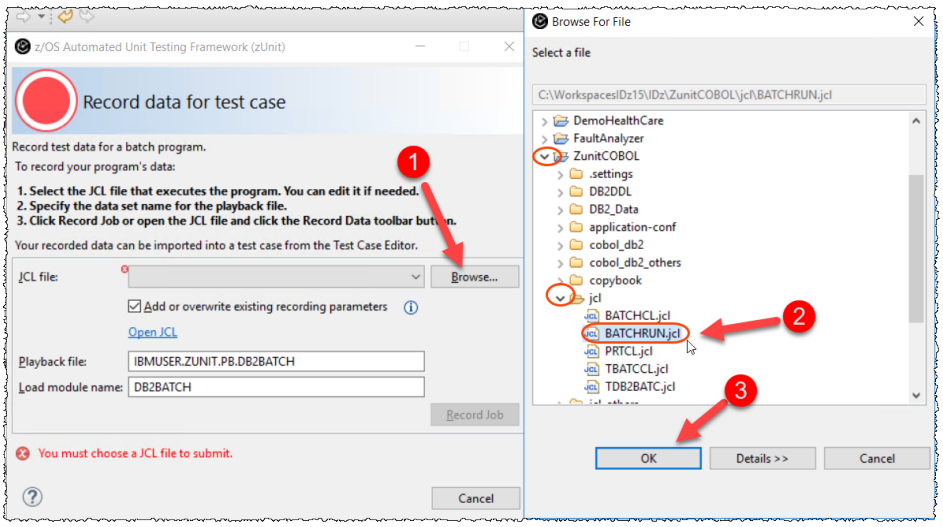
 




2.2.6 To record from a JCL batch execution into the test case, click the **Record** button on the IDz toolbar.

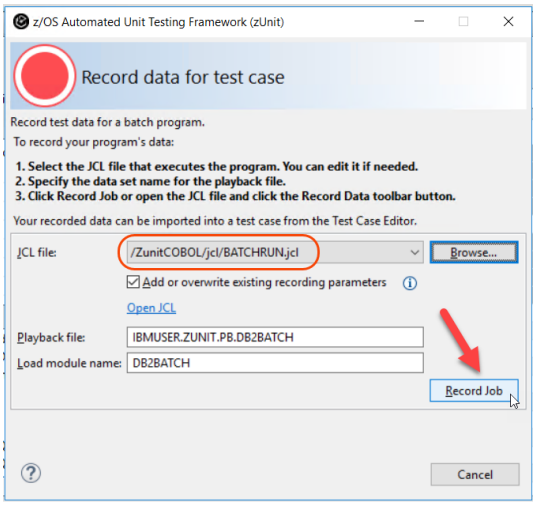



2.2.7 In the dialog that comes up, use the **Browse** button to select **BATCHRUN.jcl** under **ZunitCOBOL/jcl** and click **OK**.

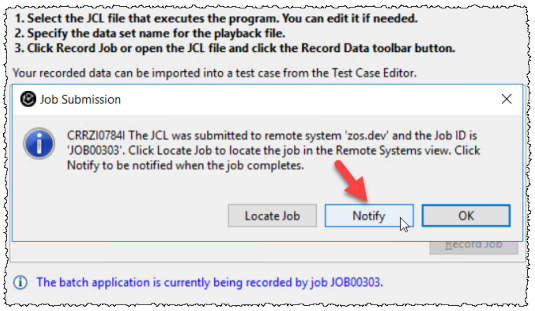


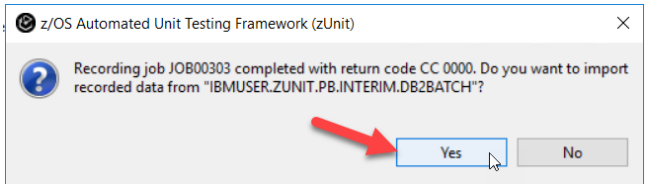
2.2.8 ⏭ Click on **Record Job** to submit the JCL for execution.




2.2.9 ⏭ Click **Notify** for the dialog below. The JCL will be submitted to run on z/OS.

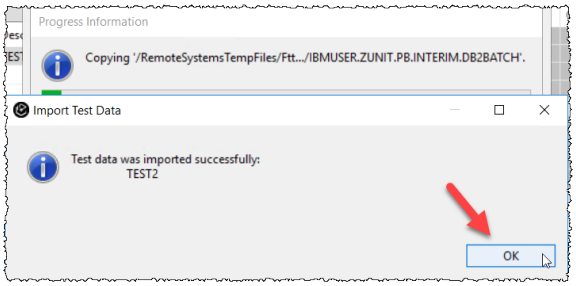



2.2.10 When the job is completed the dialog below is displayed.
⏭ Click **Yes** to create the playbackfile and import the test data.

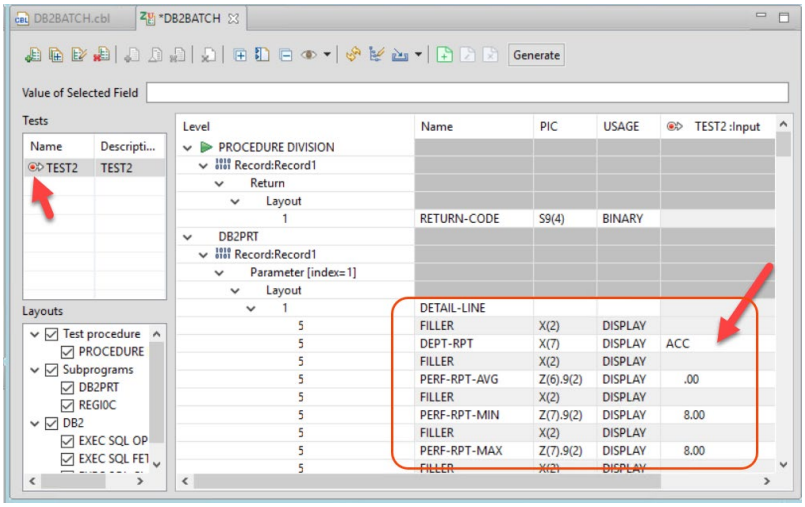



2.2.11 ⏭ Click **OK** after the data is successfully imported to the test data and playback file.

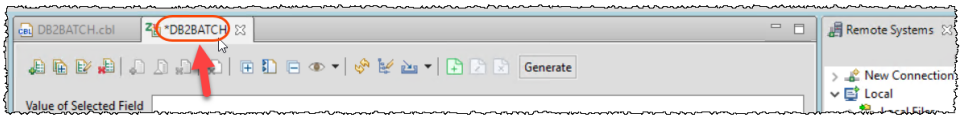
 




2.2.12 The test case editor is populated with the data recorded

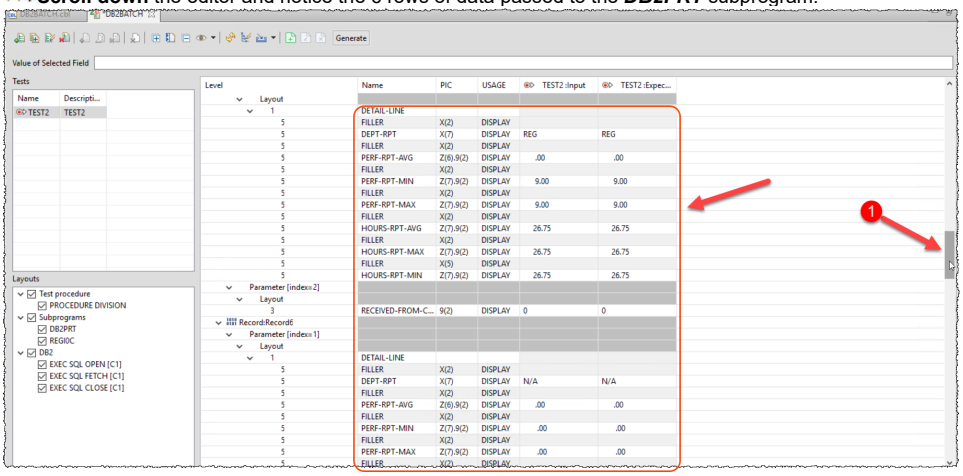



2.2.13 ⏭ Double click on the **DB2BATCH** title to enlarge the view.

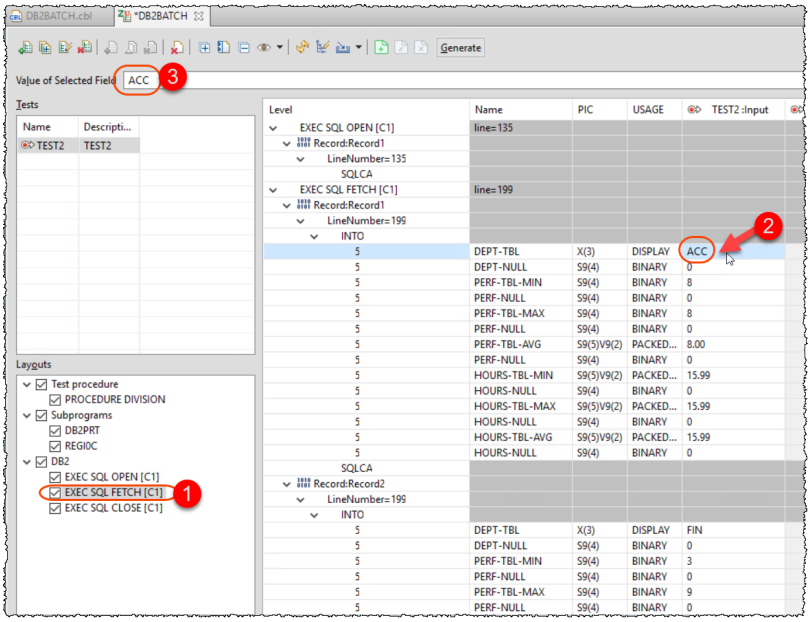


2.2.14 You now see a new test created in the editor and populated with the values from the live run.
⏭ **Scroll down** the editor and notice the 6 rows of data passed to the ***DB2PRT*** subprogram.



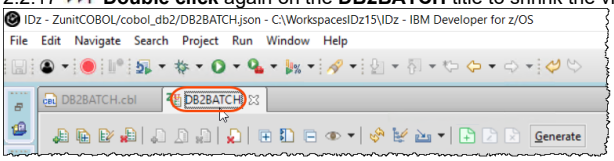
2.2.15 ⏭ 1 Select **EXEC SQL FETCH (C1)** to position the data layout for this statement.
2 Click on **ACC** . and notice that value is displayed on the 3 *Value of Selected Field*



2.2.16 ⏭ Press **Ctrl+S** to save any changes.

2.2.17 ⏭ **Double click** again on the **DB2BATCH** title to shrink the view.

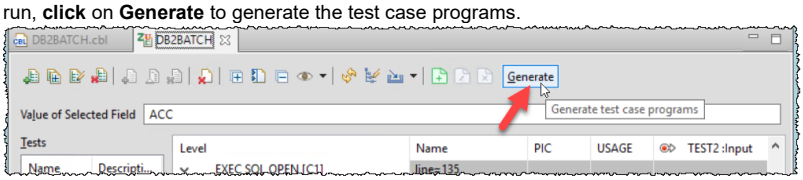



## Section 3. Generate, build, and run the unit test generated program.

You will generate, build, and run the unit test for the test case created.

### ___ 3.1 Generating the COBOL test case programs.

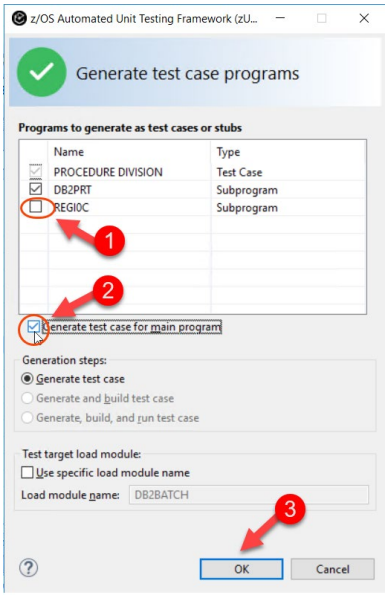
3.1.1 ⏭ Now that the expected input and output values have been set in the test case editor from the recorded run, **click** on **Generate** to generate the test case programs.




On the ***Generate test case programs*** dialog,

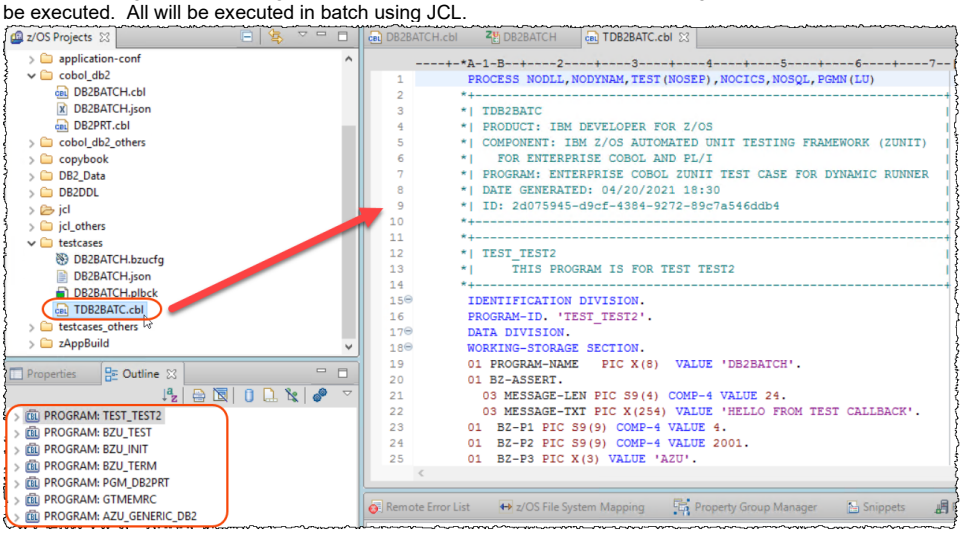
3.1.2 ⏭ **Un-select REGI0C** (you don't need stubs for this subprogram since is ready)

⏭ Select **Generate test case for main program** and then click **OK**.



3.1.3 The COBOL programs that will run the test case are generated under the folder *testcases*.
⏭ Expand **testcases** and double click on **TDB2BATC.cbl** to verify the generated programs.
Notice that in fact 7 COBOL programs were generated from this test case . This can be seen on the ***Outline*** view
You could navigate to each program if time allows, note that NONE of these programs will require CICS or DB2 to be executed. All will be executed in batch using JCL.

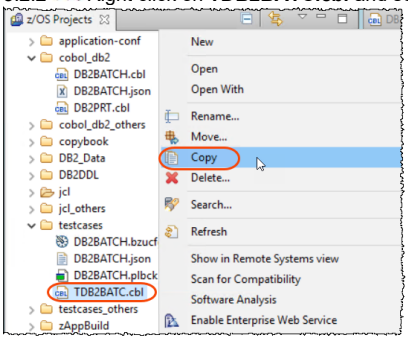



## ___ 3.2 Generate, build, and run the unit test generated program.

The 7 generated COBOL programs need to be compiled/link edited to be executed. This must be done on z/OS using the COBOL compiler.
To make this easier you could use the IBM DBB (Dependency Based Build) that is part of IDz.
But you also could use the traditional JCL once you moved the generated programs to the z/OS (PDS) to be complied and linked. Here we will show the traditional JCL way.

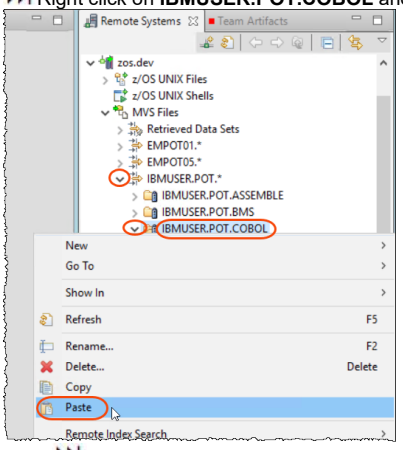
3.2.1 ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

3.2.2 ⏭ Right click on **TDB2BATC.cbl** and select **Copy**

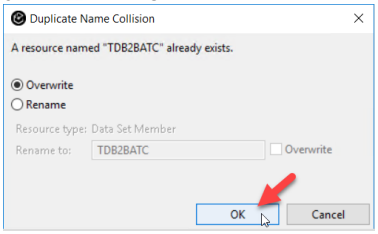


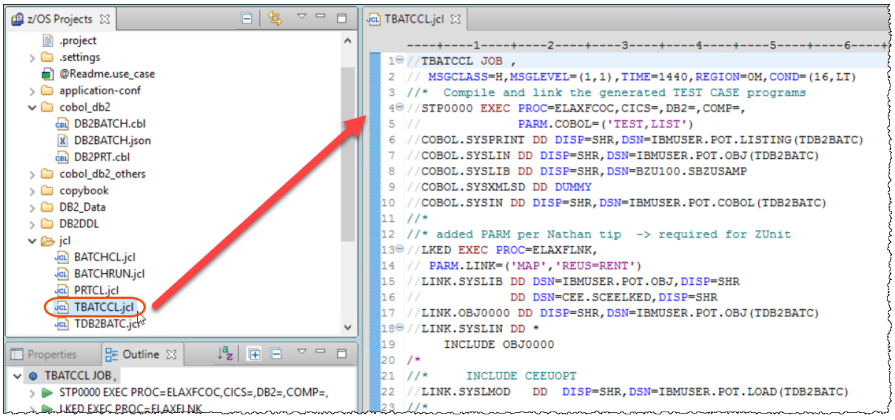
3.2.3 ⏭ Using the Remote systems (on right), expand **IBMUSER.POT.*** and **IBMUSER.POT.COBOL**
⏭ Right click on **IBMUSER.POT.COBOL** and select **Paste**




3.2.4 ⏭ Select **OK** to overwrite this member since it already existed on z/OS

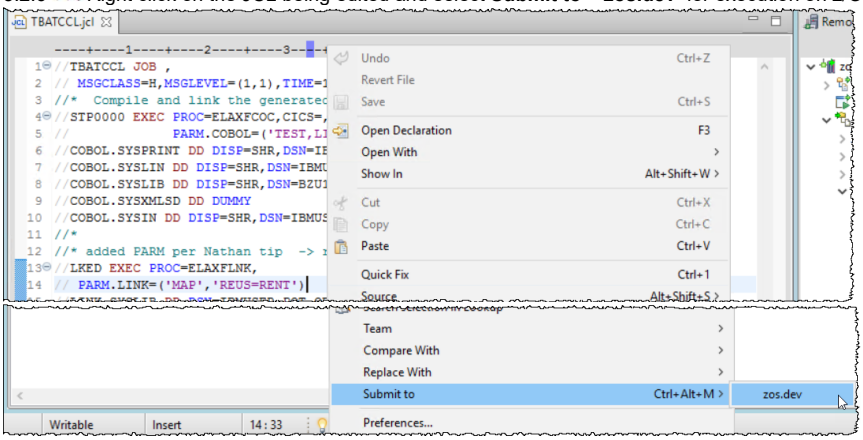



3.2.5 ⏭ Under z/OS Projects expand **ZunitCOBOL/jcl**, and double click on **TBATCCL.jcl** .
This JCL will compile and link the 7 programs that will create the test case load module.

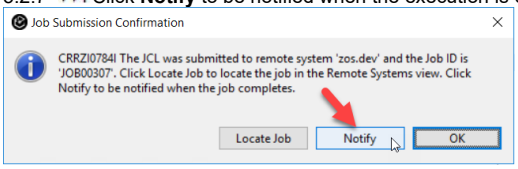


3.2.6 ⏭ Right click on the JCL being edited and select **Submit to > zos.dev** for execution on z/OS

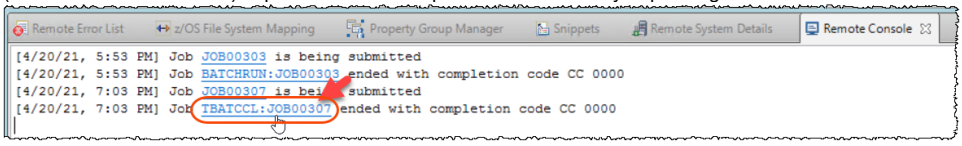



3.2.7 ⏭ Click **Notify** to be notified when the execution is complete.




3.2.8 Under *Remote Console*, you will be notified when execution is completed.

⏭ Once the execution ends, **click** on the link **TBATCCL:JOB00xxx**
(where 00**xx** is a number) to point to the Job output. This number vary depending on z/OS.




3.2.9 ⏭ Under *Remote Systems* view expand **TBATCCL:JOB00xxx** and **double click LKED:LINK:SYSPRINT** step.

⏭ Scroll down the report displayed and you will see the test case load module **TDB2BATC** created at **EMPOT.POT.LOAD** .

3.2.10 ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

**What have you done so far?**

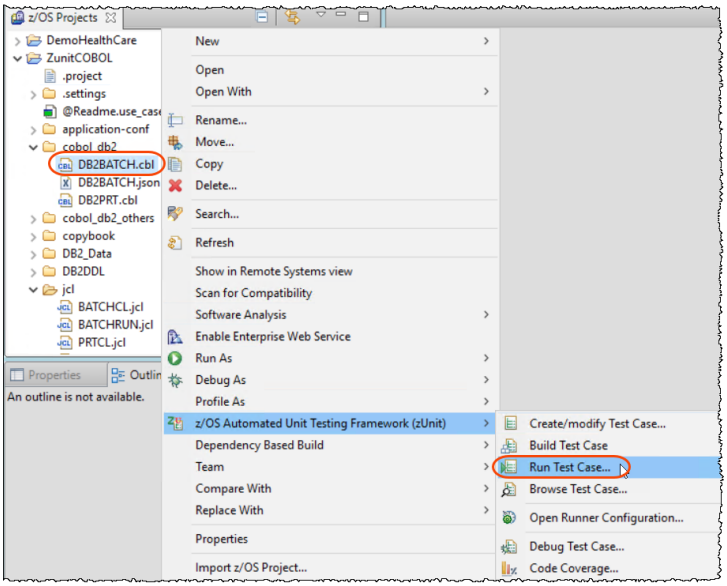
You generated a test case load module (**TDB2BATC**). This load module when executed can verify the correct input and output values handled by the program being verified. The test case can run each time that the program is modified, and the test case must pass it. You will see that

Notice that even that the tested program accessed DB2, this data is stubbed, and the test case will run in Batch via JCL without need to have DB2 active.

## ___ 3.3 Running the test case,

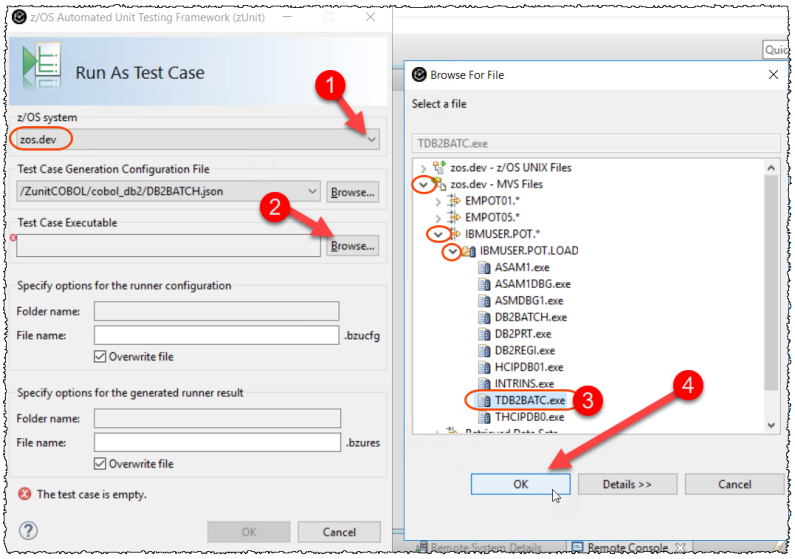
Once you have the test cases load module created you can run the test case against the *DB2BATCH* COBOL program. Since you did not make changes to the program the return code must be zero and all tests should pass.

3.3.1 ⏭ Using *z/OS Projects* view, right click on **DB2BATCH.cbl** and select **z/OS Automated Unit Testing Framework (zUnit)->Run Test Case…**

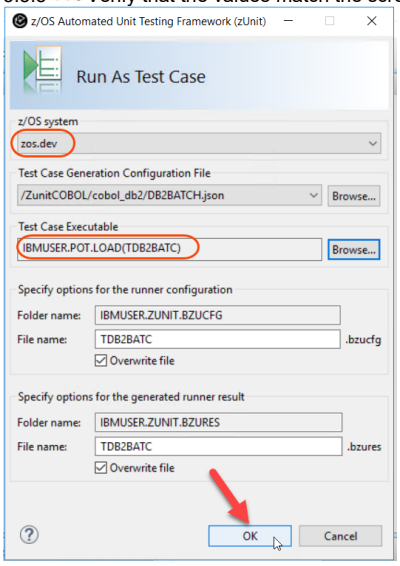


3.3.2 ⏭ Select **zos.dev** for *z/OS system*,
Use the second **Browse** button to select **`IBMUSER.POT.LOAD(TDB2BATC)`**
under **MVS Files** and **IBMUSER.POT.*.**
⏭ Click **OK**.

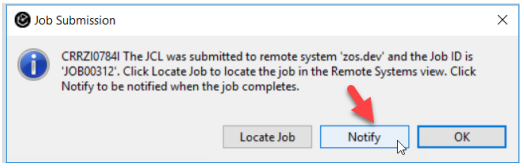



3.3.3 ⏭ Verify that the values match the screen below and click **OK** to generate and submit a JCL.

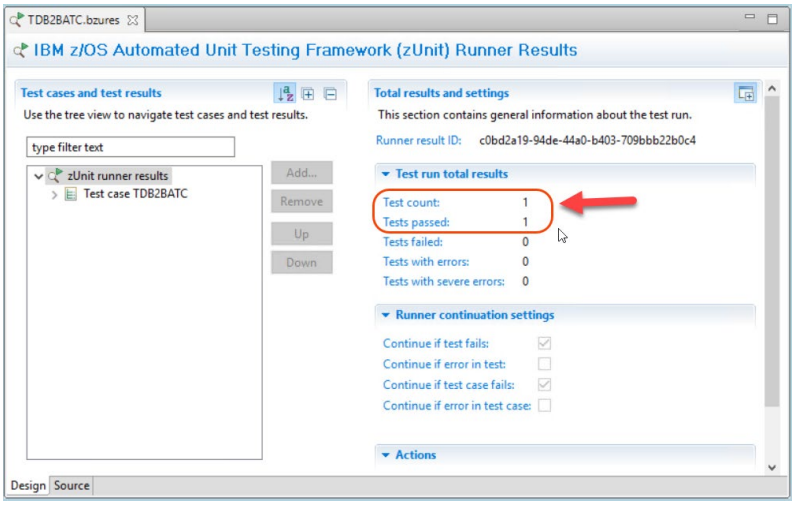


3.3.4 ⏭ A **Job Submission** dialog opens, click **Notify** to be notified of job completion.




3.3.5 Once the unit test run has completed, the results screen is displayed showing test cases ran and passed.



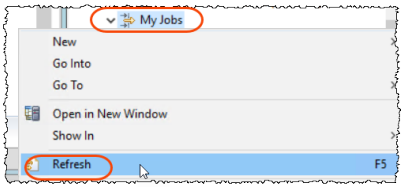

You have now created a test case with data imported from a recorded run and have been successful in testing a COBOL/DB2 batch program without the need of the DB2 environment.

3.3.6 ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

## ___ 3.4 Verify the JCL submitted that runs the test case

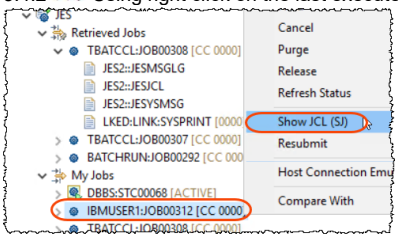
To see the JCL submitted on the previous execution

3.4.1 ⏭ Using *Remote Systems* view on right, under **JES** right click on **My Jobs**, and select **Refresh**.



3.4.2 ⏭ Using right click on the last executed and select **Show JCL (SJ)**




3.4.3 ⏭ The job submitted will be displayed. You could save it in a PDS member and use it when want to run the test cases for this program.
As you see there is no DB2 environment in that batch execution.


## Section 4. Modify the COBOL/DB2 program (introduce a bug) and rerun the unit test

Using IDz you will modify the program and introduce a bug. Then you will rerun the unit test created in the previous section.

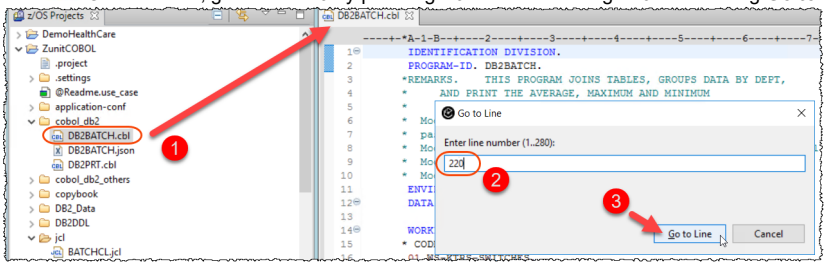
### ___ 4.1 Modifying the program and introduce a bug

.

4.1.1 ⏭ On IDz, close all active windows by pressing **Ctrl+Shift+F4**

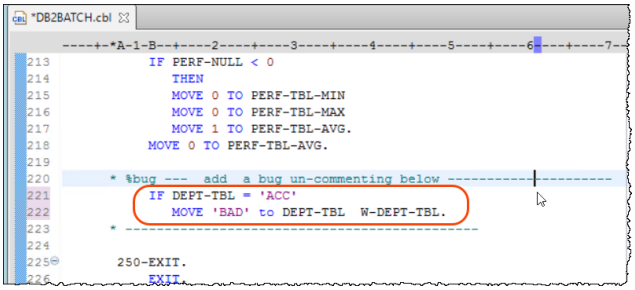
⏭ Open **DB2BATCH.cbl** under **cobol_db2** by double clicking on it in the *z/OS Projects* view.

4.1.2 ⏭ On the editor, go to line 220 by pressing **Ctrl+L** and entering **220** and clicking **Go to Line**.



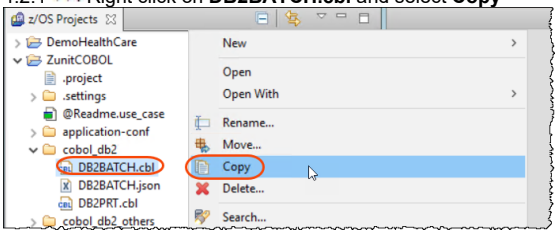
4.1.3 ⏭ Change the lines 221 and 222 removing the * from the statement that moves "**BAD**",
Tip -> Could use **Source > Toggle Comment**
⏭ Press **Ctrl+S** to save the changes. and **Ctrl + Shift + F4** to close all editors.



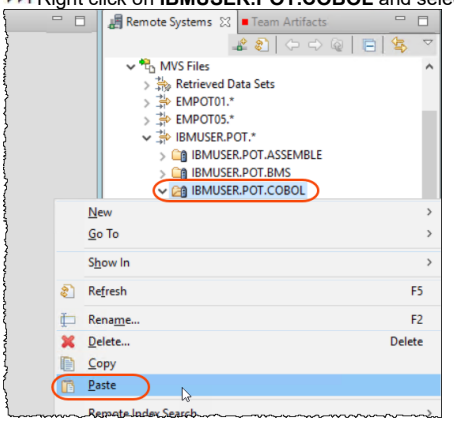

## ___ 4.2 Re-compile and link the changed program

You could use the IBM DBB (part of IDz) to do the compile and link as we already mentioned before, but on this exercise we decided to do it using the old way, via JCL. Notice that the change was made at the local IDz project and the code must be moved to z/OS to compile it there. We had done similar task when we compiled/link the generated test case.

4.2.1 ⏭ Right click on **DB2BATCH.cbl** and select **Copy**

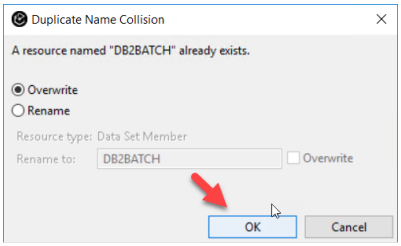



4.2.2 ⏭ Using the Remote systems (on right),
⏭ Right click on **IBMUSER.POT.COBOL** and select **Paste**

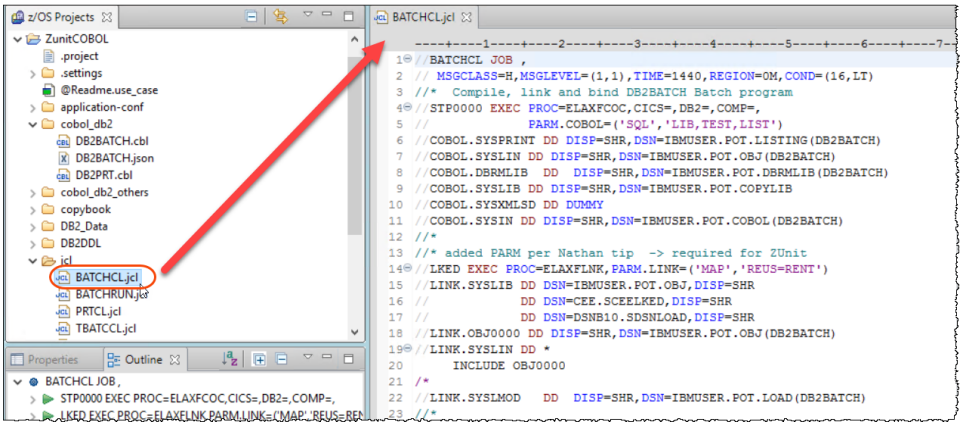


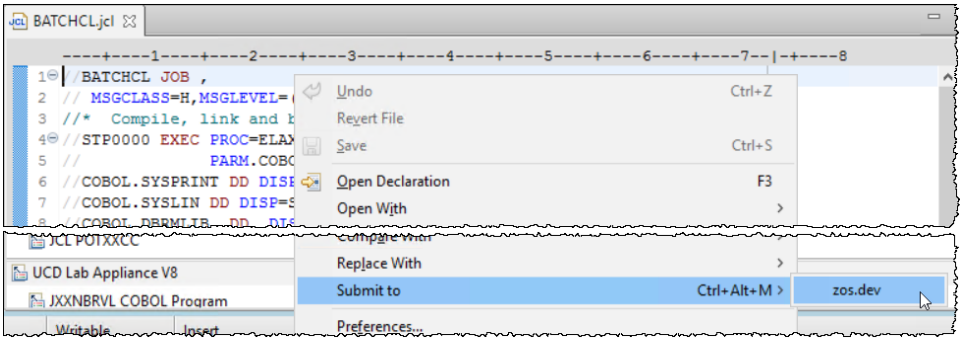
4.2.3 ⏭ Select **OK** to overwrite this member since it already existed on z/OS




4.2.4 ⏭ Under z/OS Projects expand **ZunitCOBOL/jcl**, and double click on **BATCHCL.jcl** .
This JCL will compile, link and bind the program being tested.

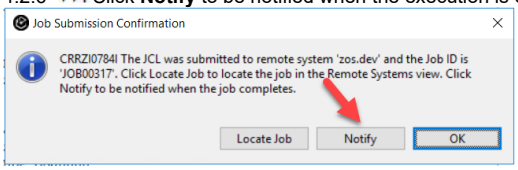



4.2.5 ⏭ Right click on the JCL being edited and select **Submit to > zos.dev** for execution on z/OS



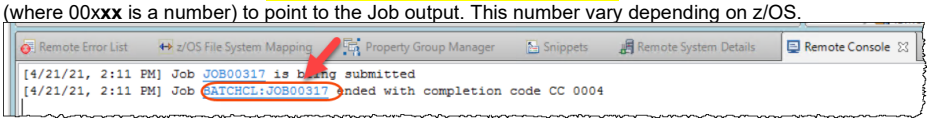

 

4.2.6 ⏭ Click **Notify** to be notified when the execution is complete.



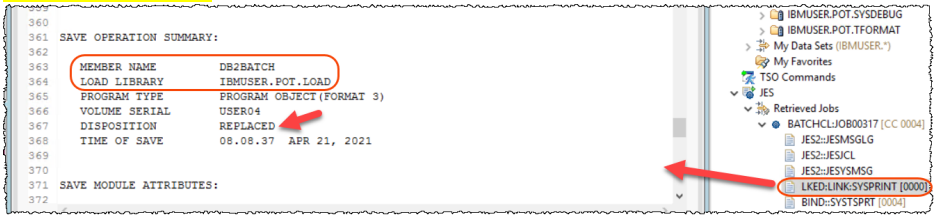

4.2.7 Under *Remote Console*, you will be notified when execution is completed.
The completion code must be 4.

⏭ Once the execution ends, **click** on the link **BATCHJCL:JOB00xxx**
(where 00**xx** is a number) to point to the Job output. This number vary depending on z/OS.




4.2.8 ⏭ Under *Remote Systems* view expand **BATCHJCL:JOB00xxx** and **double click LKED:LINK:SYSPRINT** step.

⏭ Scroll down the report displayed and you will see the load module **DB2BATCH** is replaced at **IBMUSER.POT.LOAD** .

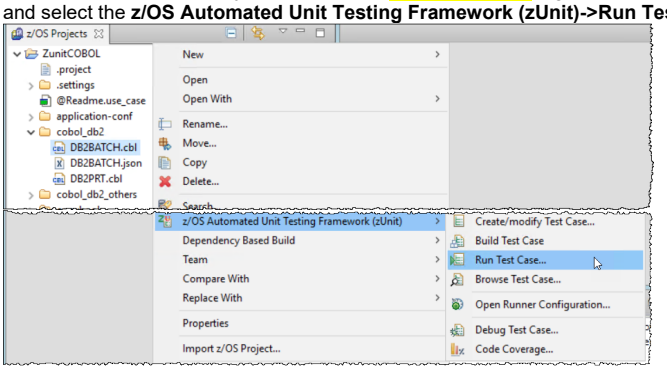



4.2.9 ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

## ___ 4.3 Running the test case again

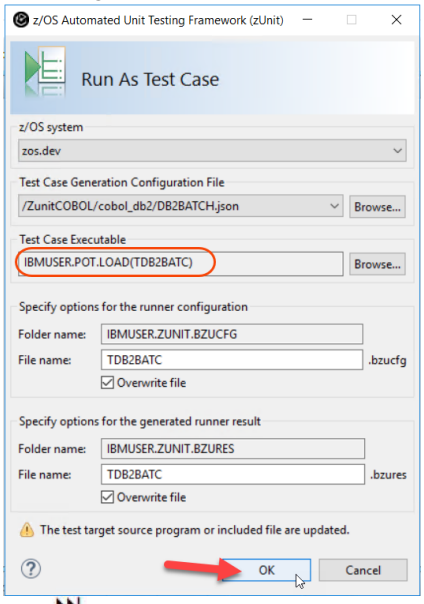
Since we introduced a bug, now the test case must fail.

4.3.1 ⏭ On the z/OS Projects view, select **DB2BATCH.cbl**, right mouse click, and select the **z/OS Automated Unit Testing Framework (zUnit)->Run Test Case..**

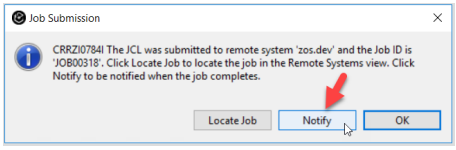


4.3.2 ⏭ Verify that the test case load module **IBMUSER.POT.LOAD(TDB2BATC)** is already selected and click **OK**

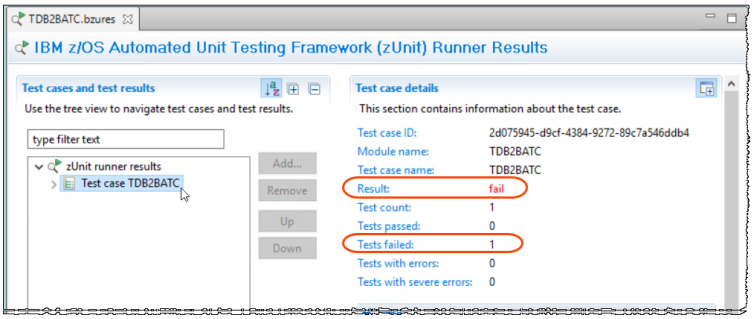



4.3.3 ⏭ Click **Notify** on the Job Submission Confirmation dialog.



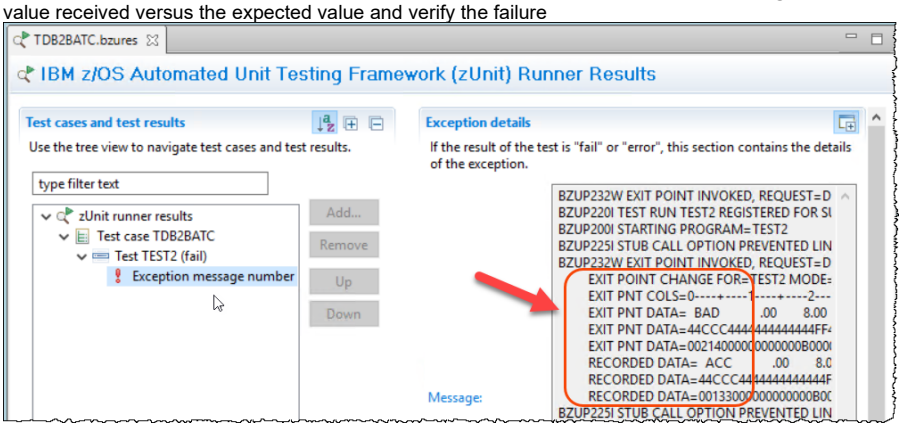

4.3.4 When the test run completes, the test results will be displayed, and it showed that one test ran and it failed. Also notice that the completion code for this JCL execution is 0004 instead of 0000

⏭ Click on **Test case TDB2BATC.** The failure is expected because the expected value of the DEPT is different from the actual value.



4.3.5 ⏭ Expand **Test case TDB2BATC, Test TEST2 (fail)** and click **Exception message number** to verify the value received versus the expected value and verify the failure




As you see the Developer introduced a BUG and the test using ZUnit fails.

## PART #2 – Fix program DB2BATCH and re-run the Unit test

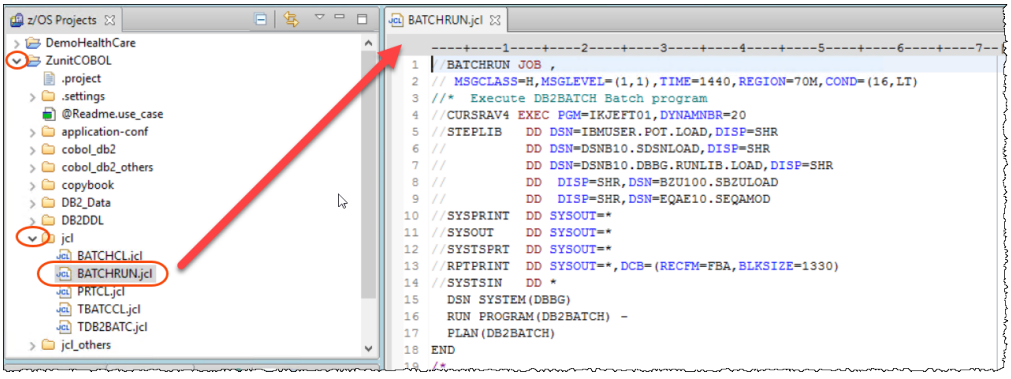
Another developer will see the bug and fix it.
On previous steps you saw on the field *DEPT* the value **BAD** instead of **ACC.**
If you submit the JCL and run this program you will see that bug in the reported printed.

### Section 5. Run the batch program and verify the bug .

You will run the Batch JCL and observe the bug on the printed report

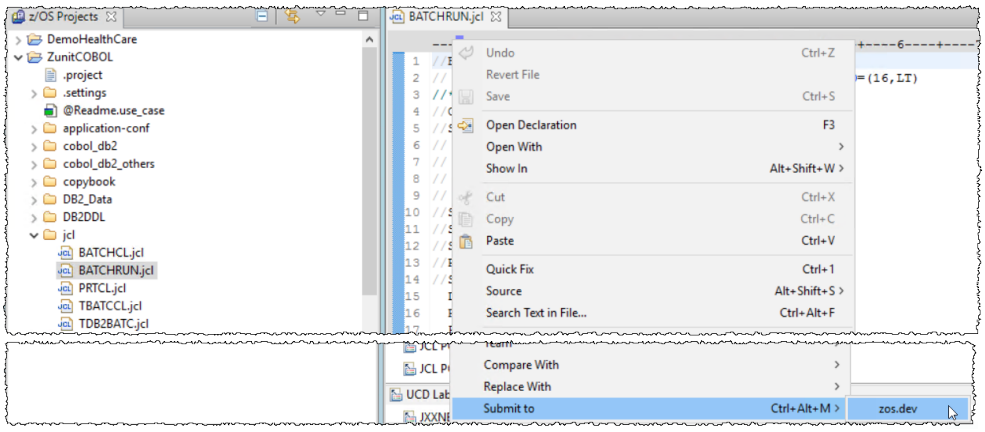
#### ___ 5.1 Verify the BUG on the printed report

5.1.1 ⏭ Under **ZunitCOBOL** expand **jcl** and double click on **BATCHRUN.jcl** to edit the JCL that will be submitted for execution.

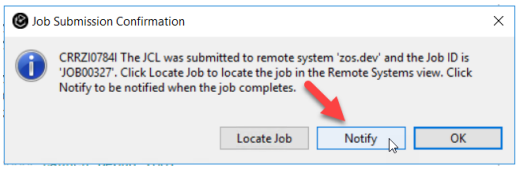


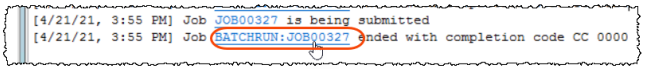
5.1.2 ⏭ Right click on the JCL edited and select **Submit to > zos.dev**



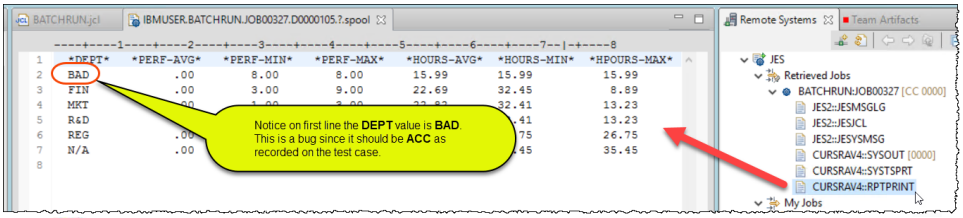

5.1.3 ⏭ Click **Notify** to be notified when the execution is complete.




5.1.4 Under *Remote Console*, you will be notified when execution is completed.
⏭ Once the execution ends, **click** on the link **BATCHRUN:JOB00xxx**
(where 00x**xx** is a number) to point to the Job output. This number vary depending on z/OS.




5.1.5 ⏭ Under *Remote Systems* view **scroll down**, expand **BATCHRUN:JOB00xxx** and **double click CURSRAV4::IRPTPRINT** step
and you will see the report produced by the ***DB2PRT*** called COBOL subprogram.




5.1.6 ⏭ Close all the opened editors using **Ctrl + Shift + F4,**

## Section 6. Use IDz to fix the bug, recompile the COBOL/DB2 program.

You will fix the using IDz, and rebuilt the executable

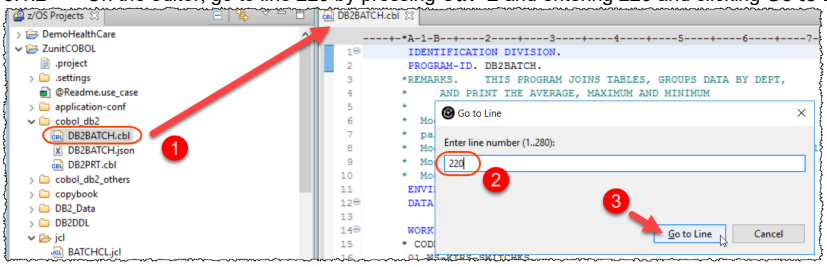
### ___ 6.1 Modifying the program to eliminate the bug

.

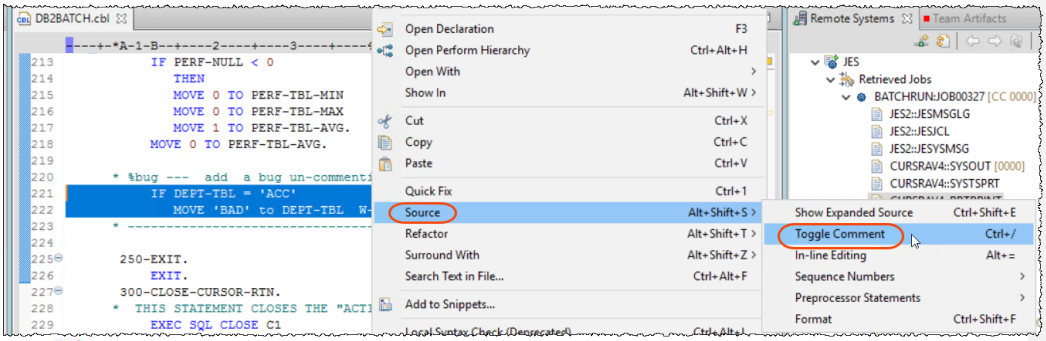
6.1.1 ⏭ Using *z/OS Projects* view open **DB2BATCH.cbl** under **cobol_db2** by double clicking on it

.

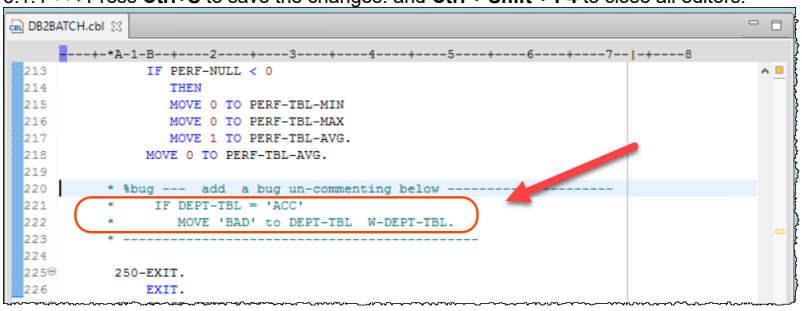
6.1.2 ⏭ On the editor, go to line 220 by pressing **Ctrl+L** and entering **220** and clicking **Go to Line**.




6.1.3 ⏭ Change the lines 221 and 222 adding an * on the column 7.
The easiest way to do that is **selecting those 2 lines**, right click and select **Source > Toggle Comment**




6.1.4 ⏭ Press **Ctrl+S** to save the changes. and **Ctrl + Shift + F4** to close all editors.



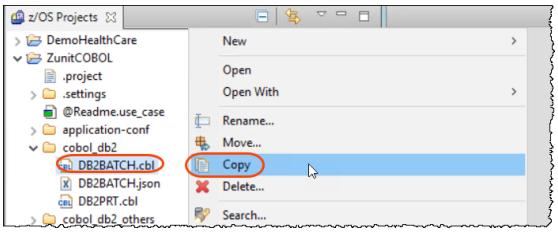

### ___ 6.2 Re-compile and link the changed program

You could use the IBM DBB (part of IDz) to do the compile and link as we already mentioned before, but on this exercise we decided to do it using the “old way, via JCL. Notice that the change was made at the local IDz project

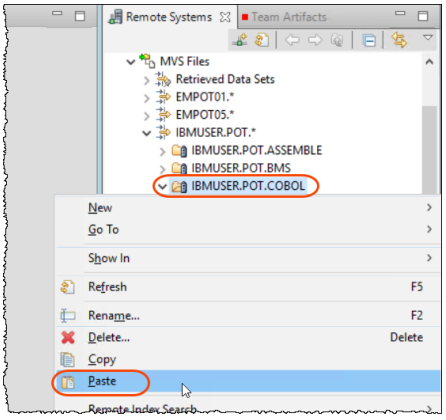
 

and the code must be moved to z/OS to compile it there. We had done similar task when we compiled/link the generated test case.

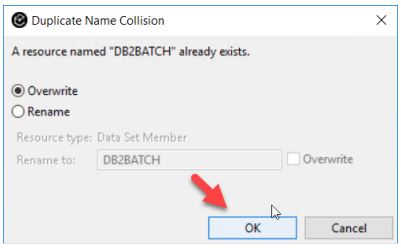
6.2.1 ⏭ Right click on **DB2BATCH.cbl** and select **Copy**




6.2.2 ⏭ Using the Remote systems (on right),
⏭ Right click on **IBMUSER.POT.COBOL** and select **Paste**

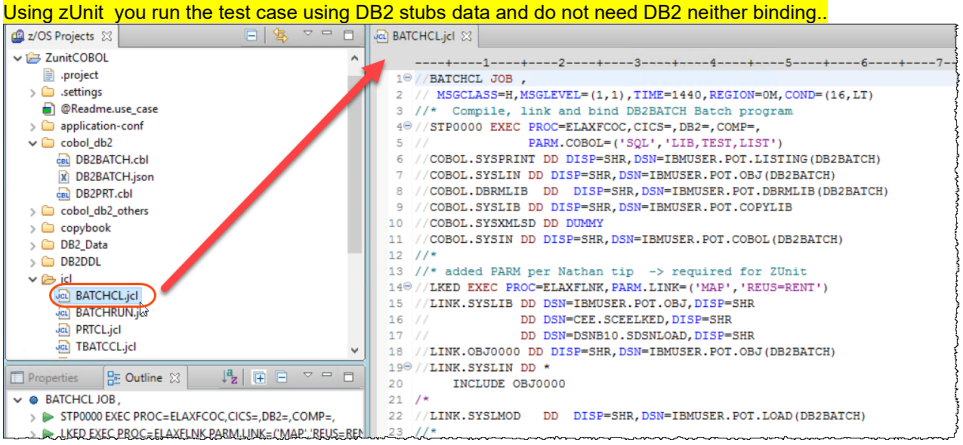



6.2.3 ⏭ Select **OK** to overwrite this member since it already existed on z/OS

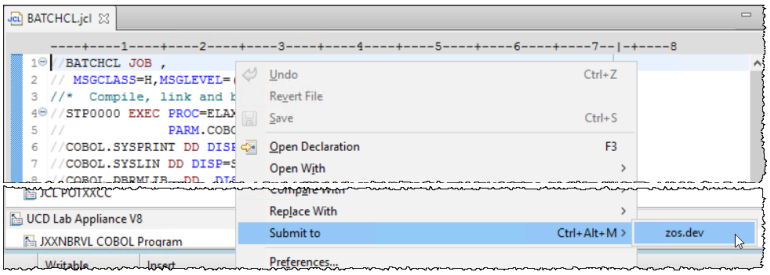


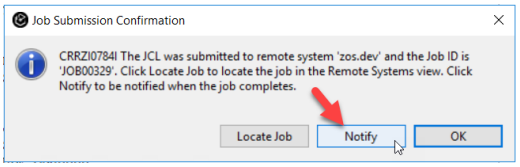
6.2.4 ⏭ Under z/OS Projects expand **ZunitCOBOL/jcl**, and double click on **BATCHCL.jcl** .
This JCL will compile, link and bind the program being tested.
Notice that the DB2 bind would not be necessary unless you will really execute it.
Using zUnit you run the test case using DB2 stubs data and do not need DB2 neither binding..



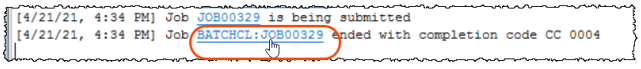

6.2.5 ⏭ Right click on the JCL being edited and select **Submit to > zos.dev** for execution on z/OS




6.2.6 ⏭ Click **Notify** to be notified when the execution is complete.

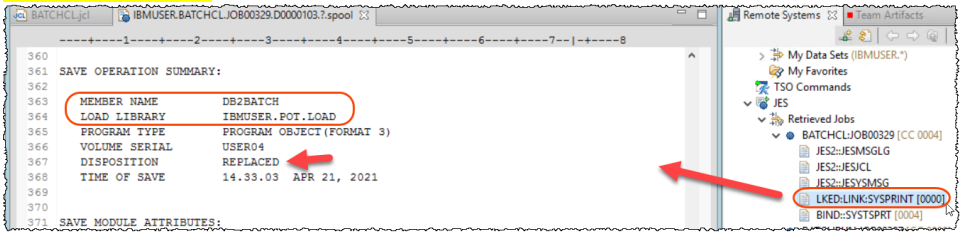



6.2.7 Under *Remote Console*, you will be notified when execution is completed.
The completion code must be 4.
⏭ Once the execution ends, **click** on the link **BATCHJCL:JOB00xxx**
(where 00**xx** is a number) to point to the Job output. This number vary depending on z/OS.



6.2.8 ⏭ Under *Remote Systems* view expand **BATCHJCL:JOB00xxx** and **double click LKED:LINK:SYSPRINT** step.
⏭ Scroll down the report displayed and you will see the load module **DB2BATCH** is replaced at **IBMUSER.POT.LOAD** .




6.2.9 ⏭ Use **Ctrl + Shift + F4** to close all opened editors.

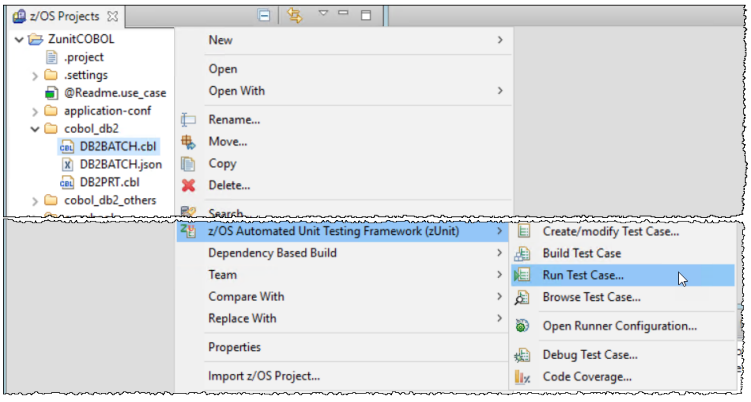
## Section 7. Rerun the zUnit and verify that the bug is eliminated.

You will run the zUnit test case and verify that the program is fixed.

### ___ 7.1 Running the test case again

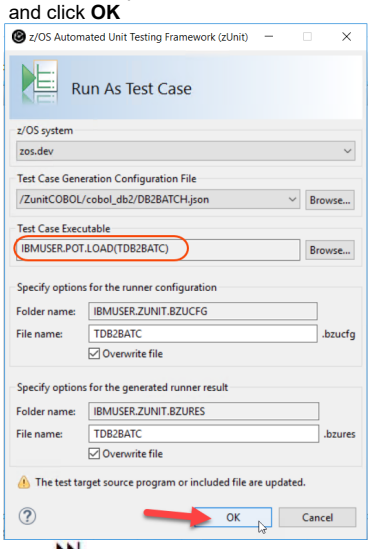
Since you fixed the bug the test case now should pass it.

7.1.1 ⏭ On the z/OS Projects view, select **DB2BATCH.cbl**, right mouse click, and select the **z/OS Automated Unit Testing Framework (zUnit)->Run Test Case..**

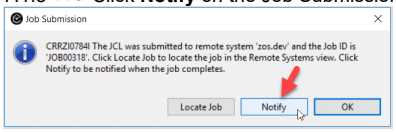


7.1.2 ⏭ Verify that the test case load module **IBMUSER.POT.LOAD(TDB2BATC)** is selected and click **OK**

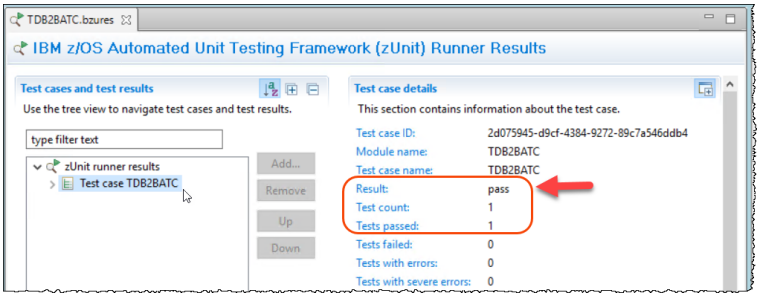



7.1.3 ⏭ Click **Notify** on the Job Submission Confirmation dialog.




7.1.4 When the test run completes, the test results will be displayed, and it showed that one test ran and it failed. Also notice that the completion code for this JCL execution is now 0000 since the test case passed

⏭ Click on **Test case TDB2BATC.** The test case now passed since the bug is fixed and results match what was originally recorded.




**Congratulations!** You have completed the Exercise.