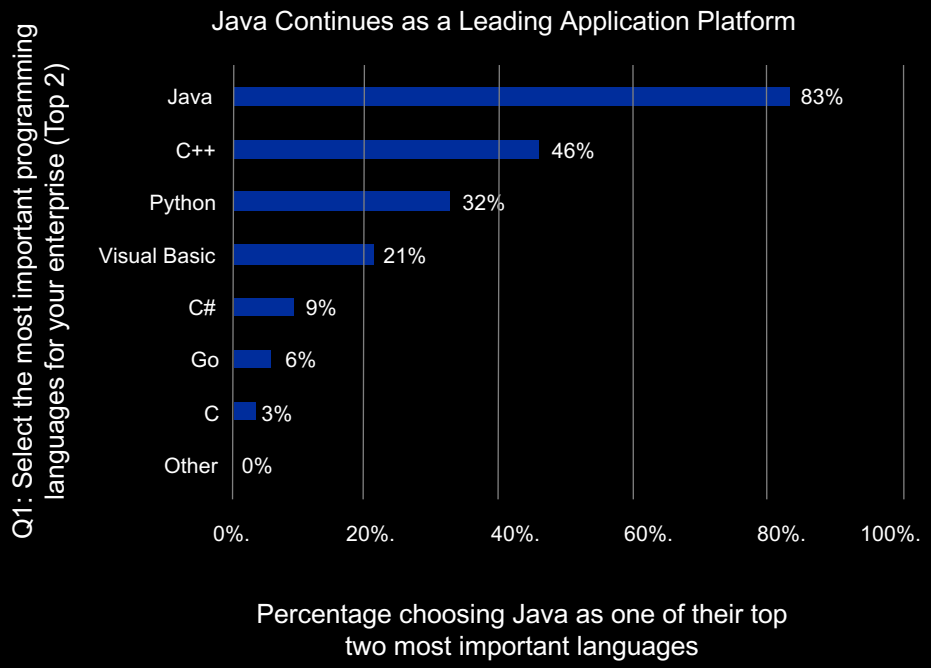
Transform your IMS™ Applications with COBOL and Java™ Interoperability

Haley Fung
Senior Product Manager
IMS and Ansible for Z
hfung@us.ibm.com

Agenda

Java™ in the enterprise



83%

of enterprise companies regard Java as one of their top two most important languages

Java on z/OS

Let us help you build…

Integrate
z/OS® middleware WebSphere Application Server, CICS® Transaction Server for z/OS, IMS, & Db2®

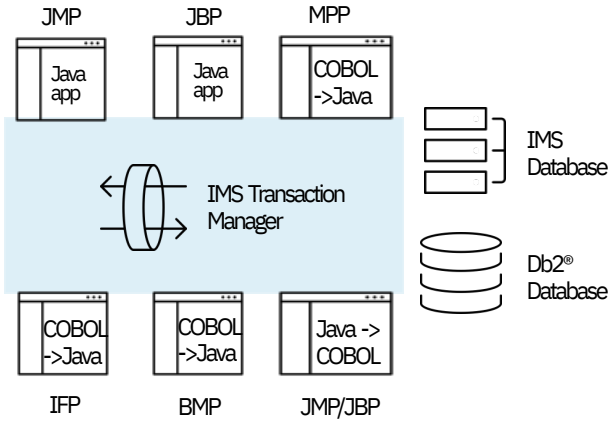
Develop
Efficiently build batch and transactional applications, microservices and more

Leverage
An abundance of APIs, libraries, and frameworks

Build
Design the architecture that's ideal for your business

Save
Run eligible Java™ work on IBM Z Integrated Information Processors (zIIPs)

How do you use Java on Z with IMS?



*Java can interoperate with COBOL or PLI

Extend existing IMS applications with Java

- Extend new business logic in Java
- Call Java code as subroutines from COBOL or PLI application or program-switch to Java transaction
- Convert heavy CPU consumption routines to Java
- Transitions development from COBOL to Java

New IMS applications in Java

- Write new IMS application in Java (JMP/JBP)
- Allows for easier maintenance
- Can still interoperate with existing COBOL or PL/I code

31-bit COBOL and 64-bit Java™ Interoperability

New enhancement

An IMS application developer can extend existing 31-bit COBOL applications that run in the IMS Message Processing Program (MPP), Batch Messaging Processing (BMP), and IMS Fast Path (IFP) regions to call 64-bit Java code to expand memory and process more data without a full rewrite.

Enable in place modernization on IMS

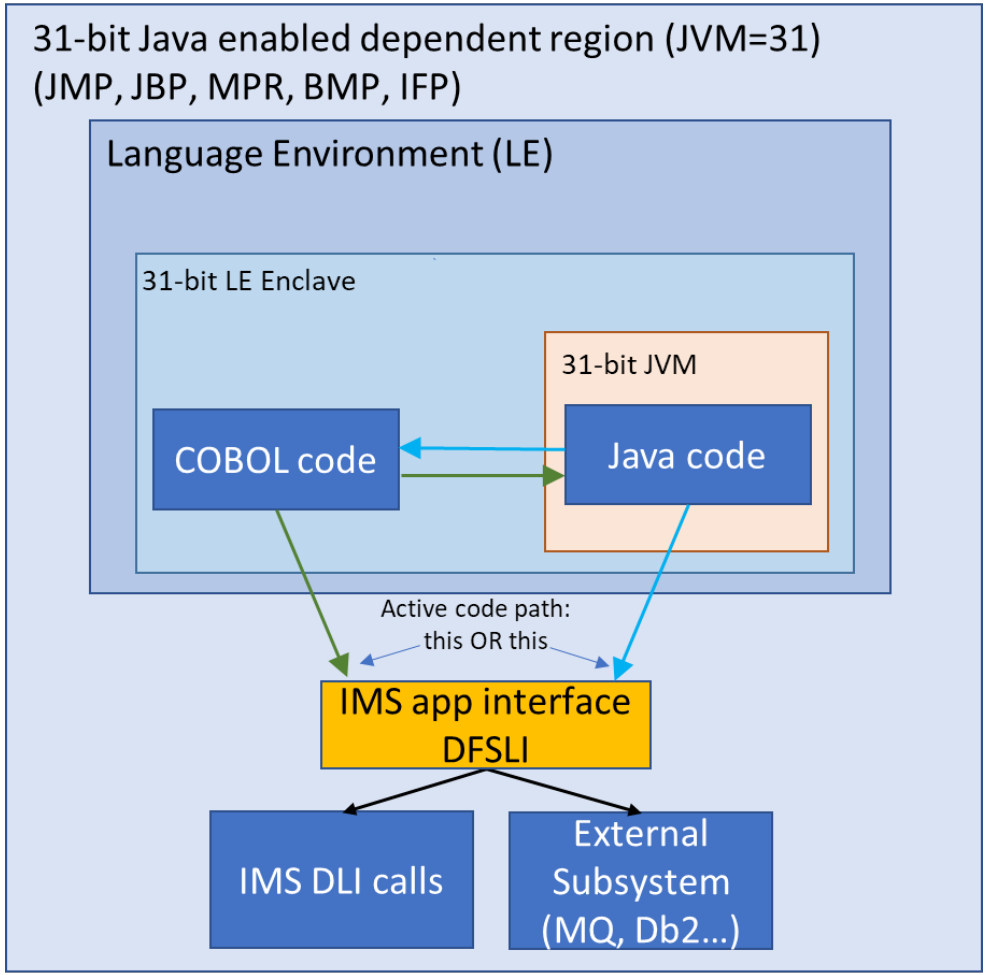
Leverage existing COBOL apps which interoperate with Java to ease Java adoption in IMS

Expand memory usage to process large loads of data without a rewrite

COBOL & Java Interoperability
Existing support

What is COBOL and Java interoperability?

- One 31-bit Language Environment (LE) created by the dependent region to run the COBOL code and the Java code
- Customers have been leveraging this interoperability support to reuse assets and modernize their applications
- But customers need 64-bit Java support



COBOL & Java Interoperability
New support

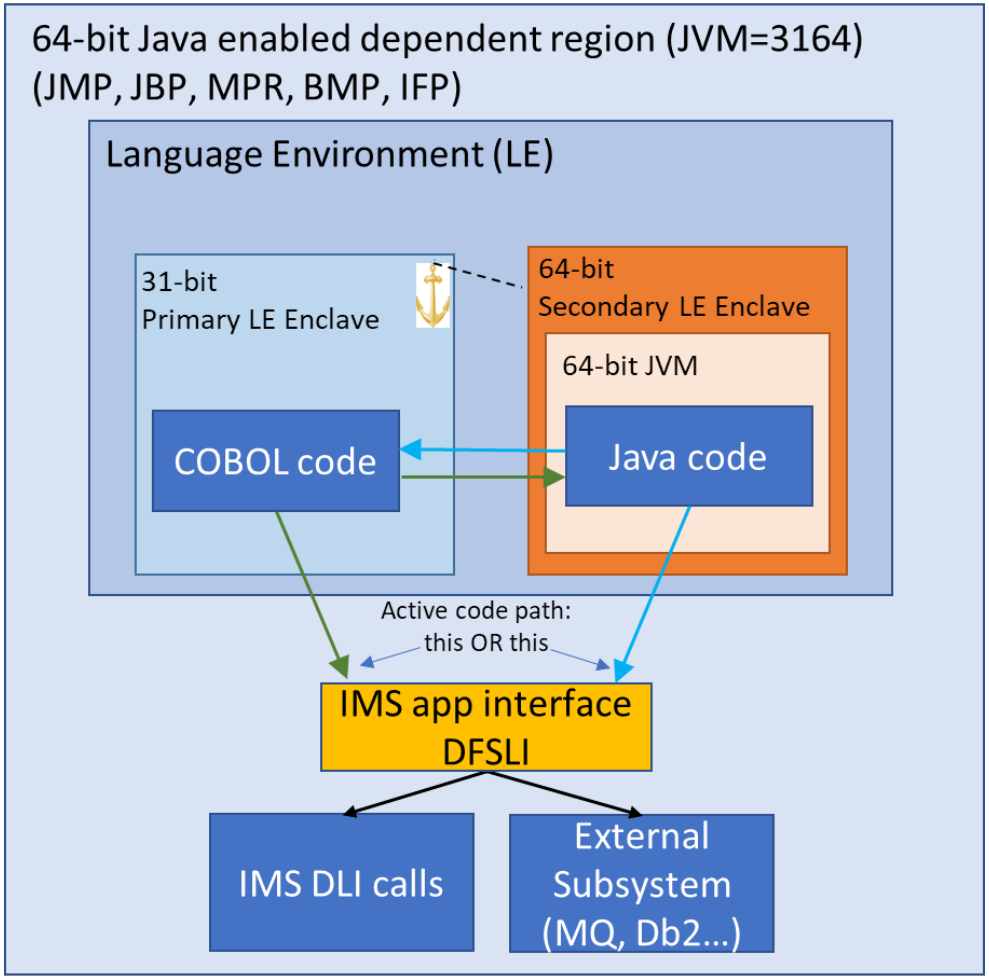
Better application modernization support

- Customers can now leverage the IMS dependent regions' support for 31-bit COBOL and 64-bit Java interoperability

Two Language Environments:

- One primary 31-bit LE to run the COBOL code
- One secondary 64-bit LE to run the Java code

Support possible because of the features added by LE, COBOL, and Java



COBOL & Java Interoperability
Activation

1. Install the APAR maintenance that provides the support: IMS(APARs PH37140 and PH47438), LE(latest maintenance), COBOL(latest maintenance), and Java SDK(V8.0.6.36 (latest FixPack)).
2. Update the dependent region's JCL to include the startup parameter JVM=3164 and add the SDFSJLIB to the STEPLIB concatenation.
3. Configure the dependent region to access the latest 64-bit Java 8 SDK
    - LIBPATH= in DFSJVMEV or STDENV DD

```
//        PROC SOUT=A,RGN=56K,SYS2=,
//             CL1=001,CL2=000,CL3=000,CL4=000,
//             OPT=N,OVLA=0,SPIE=0,VALCK=0,TLIM=00,
//             PCB=000,PRLD=,STIMER=,SOD=,DBLDL=,
//             NBA=,OBA=,IMSID=,AGN=,VSFX=,VFREE=,
//             SSM=,PREINIT=,ALTID=,PWFI=N,
//             APARM=,LOCKMAX=,APPLFE=,ENVIRON=,
//             JVMOPMAS=,PARDLI=,JVM=3164
//*
//REGION EXEC PGM=DFSRRC00,REGION=&RGN,
//             TIME=1440,DPRTY=(12,0),
//             PARM=(MSG,&CL1&CL2&CL3&CL4,
//             &OPT&OVLA&SPIE&VALCK&TLIM&PCB,
//             &PRLD,&STIMER,&SOD,&DBLDL,&NBA,
//             &OBA,&IMSID,&AGN,&VSFX,&VFREE,
//             &SSM,&PREINIT,&ALTID,&PWFI,
//             '&APARM',&LOCKMAX,&APPLFE,
//             &ENVIRON,&JVMOPMAS,&PARDLI,&JVM)
//*
//STEPLIB  DD DSN=IMS.&SYS2.PGMLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.SDFSJLIB,DISP=SHR
//         DD DSN=IMS.&SYS2.SDFSRESL,DISP=SHR
```

```
*****************************
HLQ.PROCLIB(DFSJVMEV)
*****************************
LIBPATH=/tmp:>
javahome/J8.0_64/lib/s390x/j9vm/:>
```

COBOL & Java Interoperability
Application changes

Manual invocation using JNI function calls:

Java Native Interface Specification Contents:

https://docs.oracle.com/javase/8/docs/technotes/guides/jni/spec/jniTOC.html

FindClass, GetStaticMethodID, NewString, GetStaticFieldID

*Java object references are 64-bit and must be declared as PIC 9(18) COMP-5 in COBOL.

For 31-bit Java:

02 MySimpleClass object reference jclass.

For 64-bit Java must change to:

02 MySimpleClass pic 9(18) comp-5.

COBOL & Java Interoperability
Application changes

Simplified syntax using CALL statement:

The latest COBOL compiler support a new flavor of the CALL statement with the following syntax:

```
CALL 'Java.class-name.static-method-name' USING …
```

CALL statement

https://www.ibm.com/docs/en/cobol-zos/6.4?topic=statements-call-statement

Mapping between COBOL and Java types

https://www.ibm.com/docs/en/cobol-zos/6.4?topic=interoperability-mapping-between-cobol-java-types

COBOL & Java Interoperability
Application changes

Simplified syntax using JAVA-CALLABLE and JAVA-SHAREABLE directives:

```
>>JAVA-CALLABLE
```

The JAVA-CALLABLE directive instructs the compiler to make the COBOL program automatically callable from Java™

https://www.ibm.com/docs/en/cobol-zos/6.4?topic=interoperability-java-callable

```
>>JAVA-SHAREABLE ON|OFF
```

Used around WORKING-STORAGE data items to indicate that they are to be made read/write accessible from Java™ applications interoperating with this COBOL program.

https://www.ibm.com/docs/en/cobol-zos/6.4?topic=interoperability-java-shareable

COBOL & Java Interoperability
Compiling the application

For example, use the cob2 script from the /bin folder of the COBOL install directory.

Use the DLL side-deck files:

- `libjvm31.x provided by the Java SDK`
- `igzxjni2.x provided by COBOL`

Use the cjbuild utility if using the CALL ‘Java….’ statement or the JAVA-CALLABLE or JAVA-SHAREABLE directives.

COBOL & Java Interoperability
Demo

COBOL & Java Interoperability
Blog and Tutorials

Blog:

Good news! 31-bit COBOL can now talk to 64-bit Java in IMS:

https://community.ibm.com/community/user/ibmz-and-linuxone/blogs/carlos-alvarado1/2022/02/16/31-bit-cobol-can-now-talk-to-64-bit-java

Tutorials:

Java to COBOL tutorial

https://community.ibm.com/community/user/ibmz-and-linuxone/viewdocument/leveraging-cobol-business-logic-fro?CommunityKey=eba3ada3-db89-4dca-9154-328195f5e560&tab=librarydocuments

COBOL to Java tutorial

https://community.ibm.com/community/user/ibmz-and-linuxone/viewdocument/modernizing-an-ims-cobol-applicatio?CommunityKey=eba3ada3-db89-4dca-9154-328195f5e560&tab=librarydocuments

Thank you!

Haley Fung
Senior Product Manager
IMS and Ansible for Z
hfung@us.ibm.com