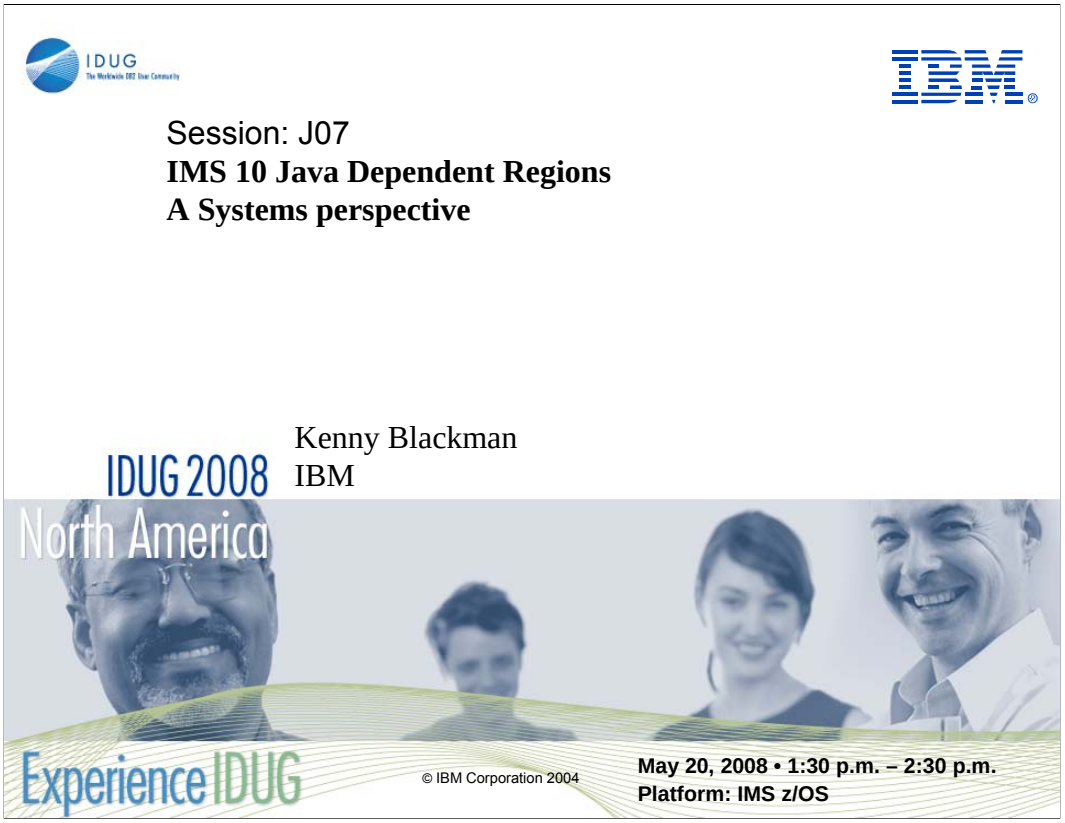


This presentation describes the latest developments in IMS that allow the Java world to be used with your IMS applications and data. This includes IMS V10 updates and the use of SDK V5 JVM.In this presentation you will learn about the IMS Java Dependent Regions, how they can be used for new IMS Java applications and to leverage existing COBOL business logic. You will also learn about the Shared Class Cache Java Virtual Machine.

Contents

- IMS Background
- Java Background
- Java Virtual Machine
- IMS Java Dependent Regions
- DB2 and IMS Java processing

In this presentation we will give a quick background review of IMS and Java. We will describe and compare the two types of JVMs supported by IMS V9 and V10 Java Dependent Regions. Then we will cover DB2 and Java processing. The presentation will meet the following objectives.

Objective 1: Learn the IMS Java programming model

Objective 2: Learn IMS Java Dependent Regions setup

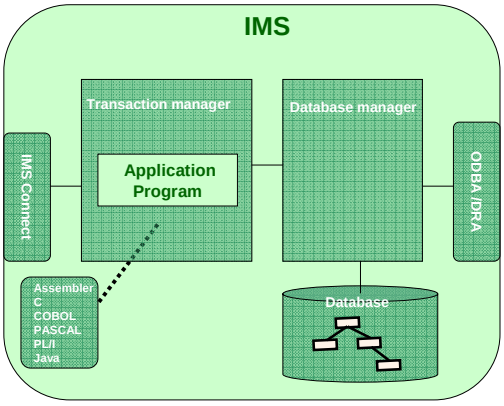
Objective 3: Learn Enterprise COBOL and Java processing

Objective 4: Learn DB2 from IMS Java processing

Objective 5: Learn z/OS Java JVM

What is IMS?

- Transaction Management System
- Multilingual Application Program server
- Database Management System



Services accessed without knowledge of underlying implementation

Services written in C# running on .Net platforms and services written in Java running on Java EE platforms can both be consumed by a common composite application

According to IBM's own estimates, SOA has a market opportunity of $65 billion this year. Gartner says that by 2008, SOA will provide the basis for 80 percent of new IT transformation projects.

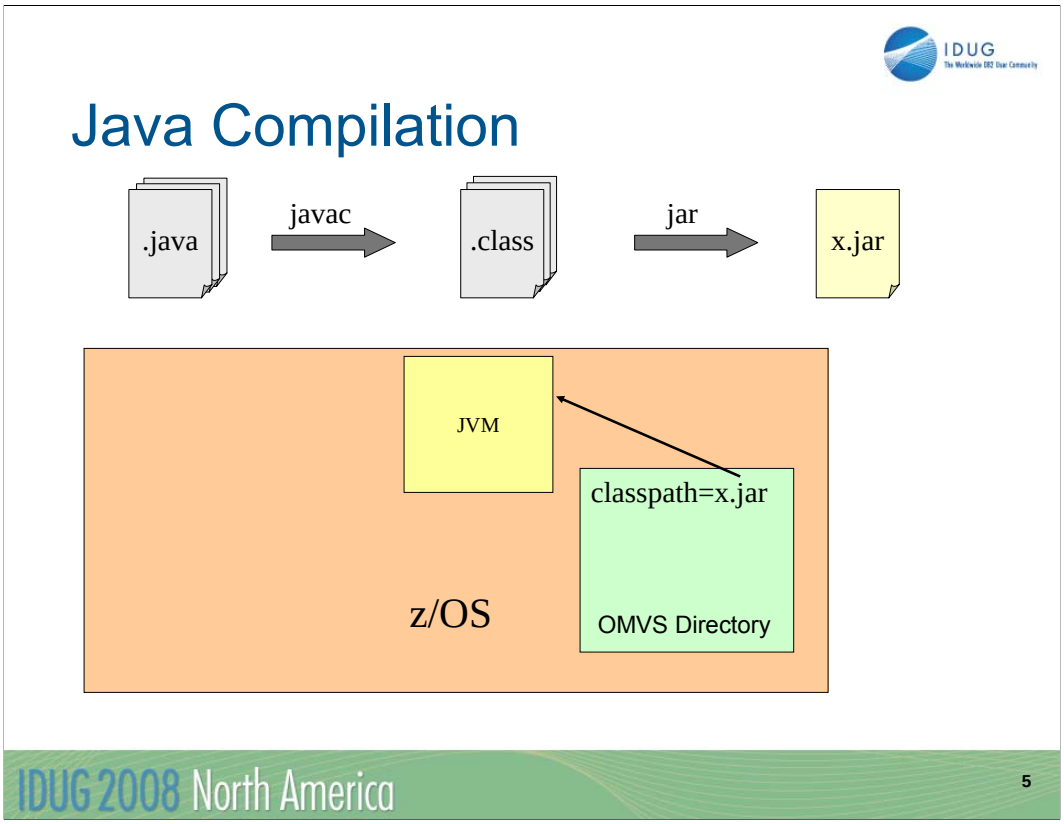
- IMS TM resource adapter
- IMS distributed DB resource adapter
- IMS DB resource adapter
- IMS Java dependent region (JDR) resource adapter

IMS TM resource adapter - download from the Web (previously known as IMS Connector for Java or IC4J)

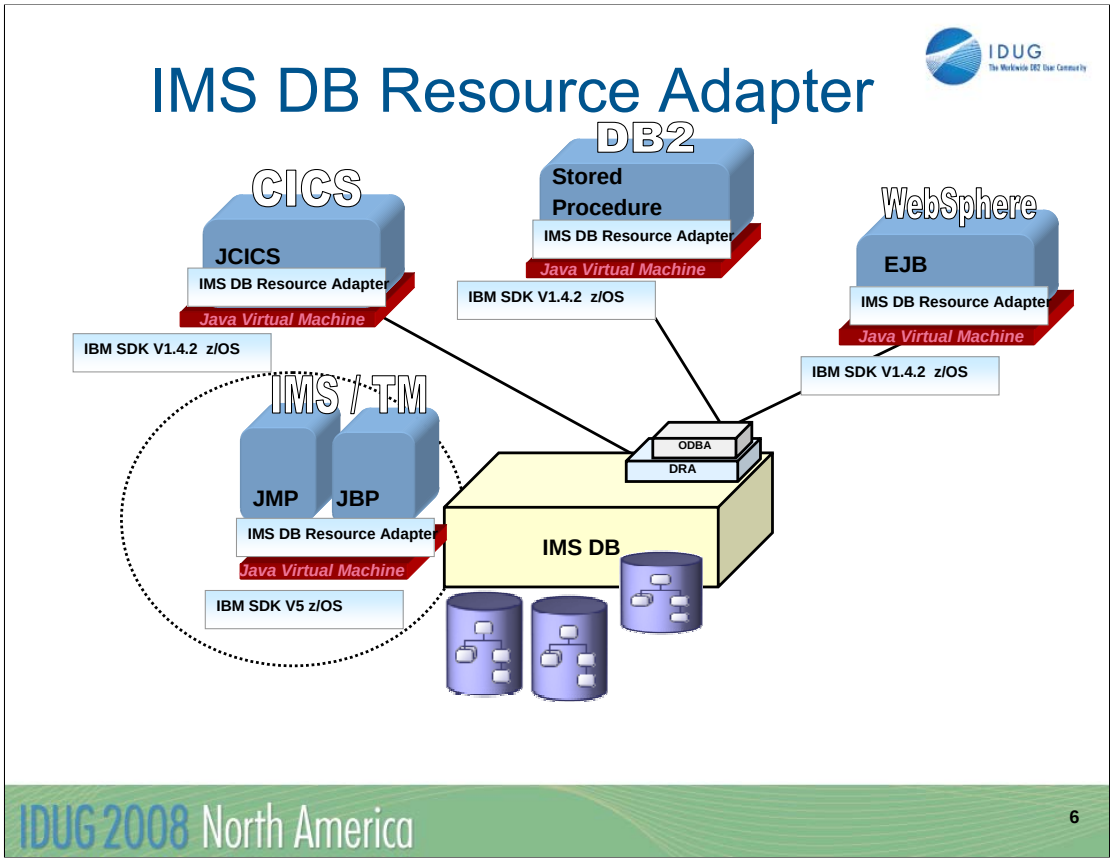
IMS distributed DB resource adapter - download from the Web (previously known as IMS Java remote database services)

IMS DB resource adapter - installed via SMP/E with IMS (previously know as IMS JDBC Connector).

IMS Java dependent region (JDR) resource adapter - installed via SMP/E with IMS (new name).



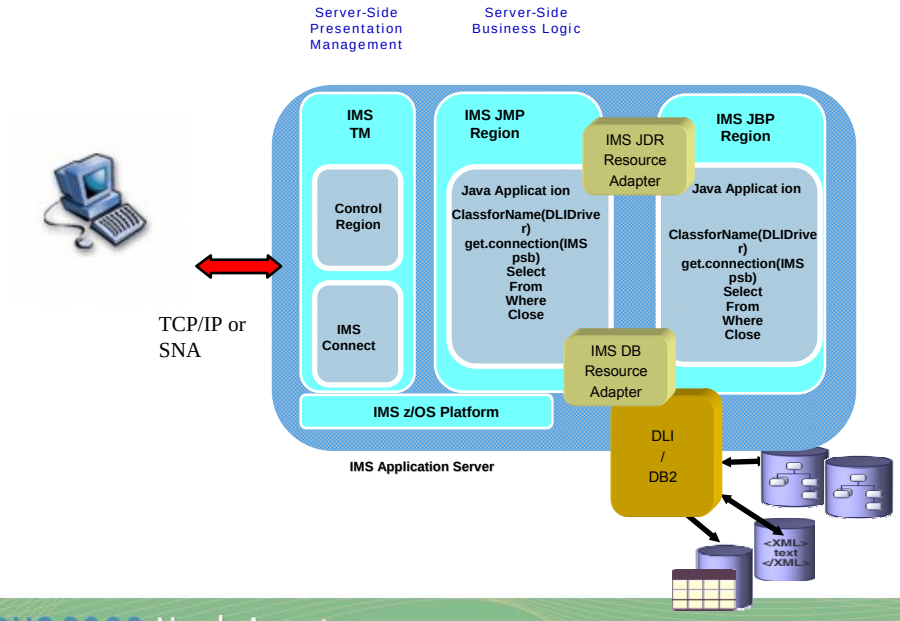
OMVS Directory



The IMS DB Resource Adapter enables JDBC access to IMS DB from IMS TM JMP/JBP environments, CICS Java application, DB2 Java Stored procedure, and Enterprise Java Beans running on WebSphere distributed and z/OS environments.

IMS V10 requires SDK V5 for JMP and JBP regions, IMS DB Resource Adapter for CICS, DB2 or WAS requires SDK V1.4.2 or higher.

IMS TM JMP/JBP Access to Data



Java Dependent Regions

/DIS ACTIVE command:

```
REGID JOBNAME  TYPE  TRAN/STEP PROGRAM  STATUS   CLASS
  1        JMPRGN1        JMP                          WAITING  1, 2, 3, 4

  2        JBPREGA         JBP    JBP                    JBPPSB
           BATCHREG   BMP   NONE
           FPRGN          FP      NONE
           DBTRGN       DBT   NONE
          DBRMCTAB  DBRC
```

IMS Definition Changes

APPLCTN macro changes

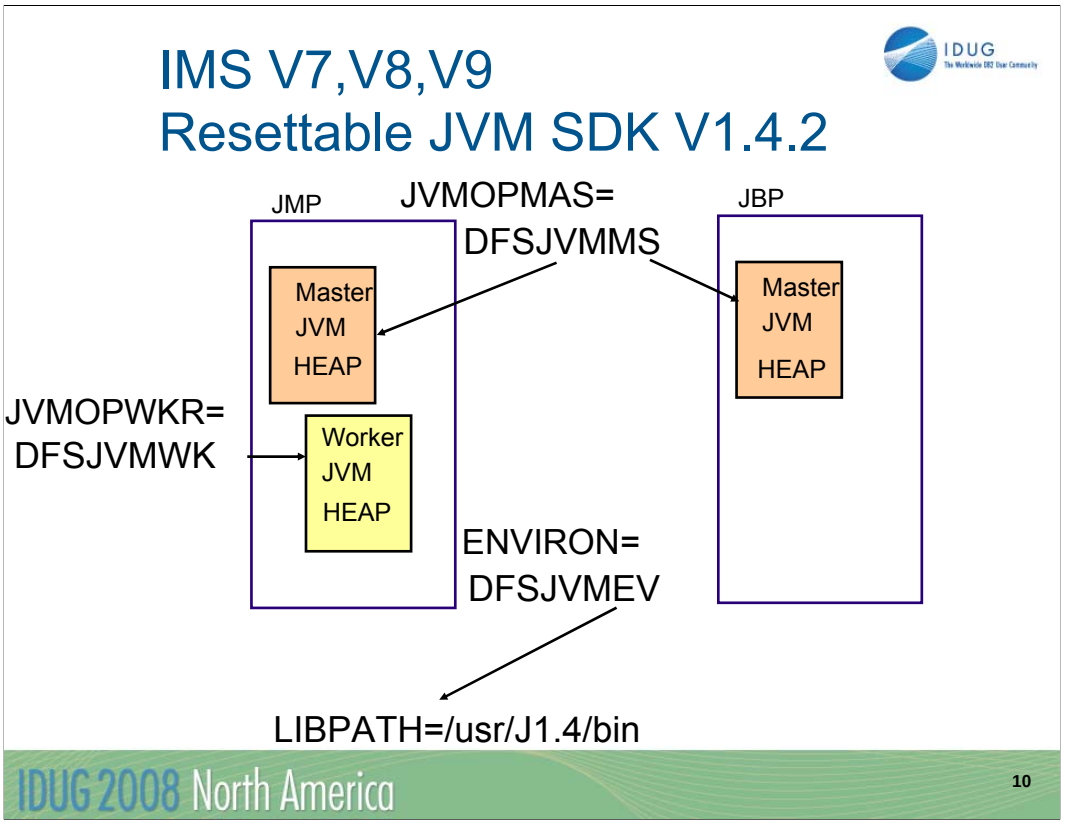
- Added support for LANG=JAVA if GPSB= specified
- Following error message issued if LANG=JAVA and FPATH=YES:
  - G220 LANG=JAVA INVALID WHEN FPATH=YES.

PSBGEN macro changes

- Added support for LANG=JAVA
  - JMP requires LANG=JAVA

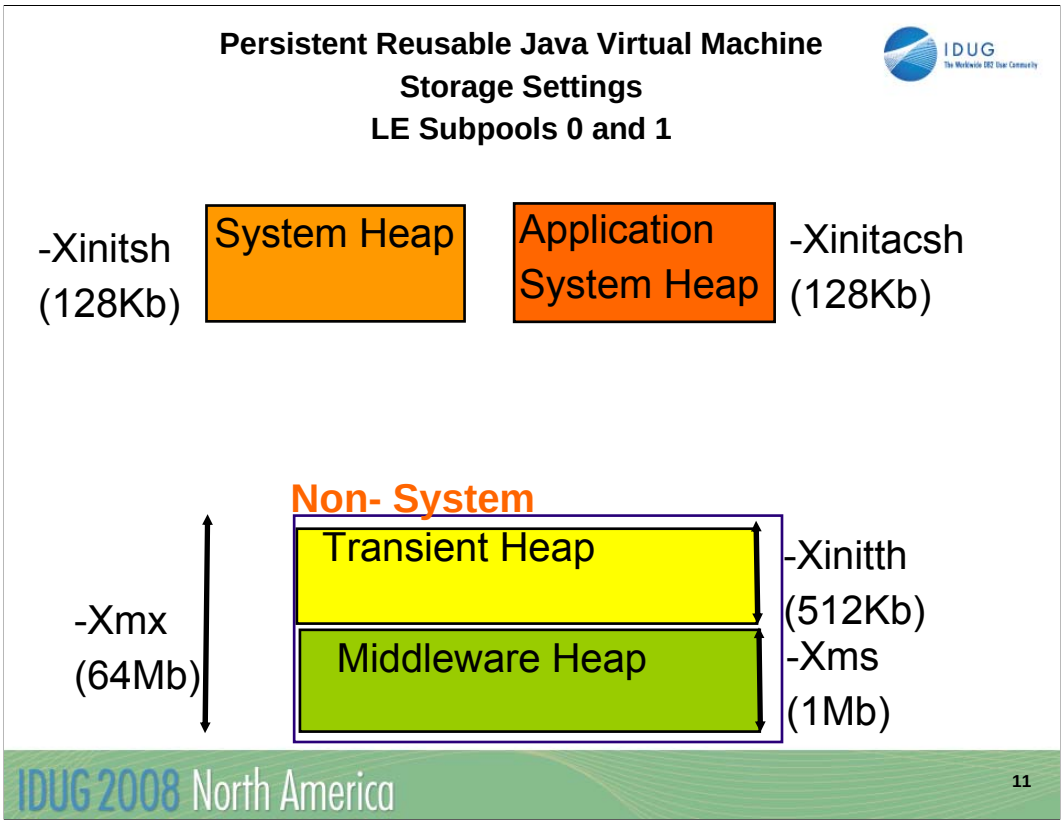
IMSGEN macro changes

- Added SCEERUN= parameter
  - Specifies the name of the C Runtime Library
  - STEPLIB concatenation for DFSJMP and DFSJBP
  - Max of 44 alphanumeric characters for the name
    - Default is CEE.SCEERUN



This visual shows the Persistent Reusable JVM supported in the IMS Java Dependent Regions and the procedure parameters provided in IMS sample library

JVMOPMAS=DFSJVMMS
Specifies the JVM options for the master JVM
JVMOPWKR=DFSJVMWK
Specifies the JVM options for the worker JVM
ENVIRON=DFSJVMEV
Must contain the pathname to JVM
Must contain the pathname to the IMS Java native code
HEAP - contiguous piece of storage obtained at JVM initialization.



runopts(ALL31(ON))

runopts(ANYHEAP(768K,128K,ANYWHERE,FREE))

runopts(BELOWHEAP(24K,2K,FREE))

runopts(HEAP(4M,512K,ANYWHERE,FREE))

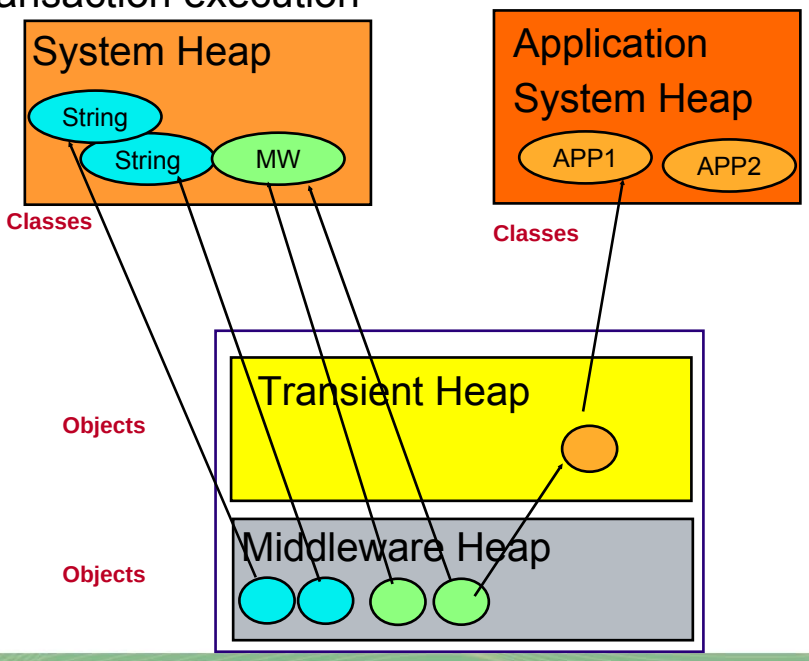
runopts(LIBSTACK(1K,1K,FREE))

runopts(STACK(16K,4K,ANYWHERE,FREE,64K,16K))

runopts(STORAGE(NONE,NONE,NONE,1K))

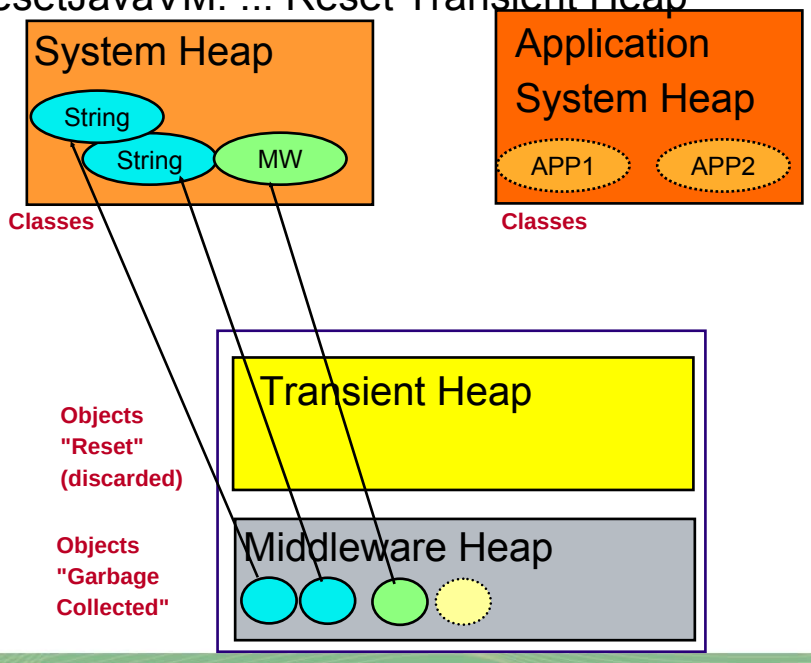
JVM Lifecycle

Transaction execution



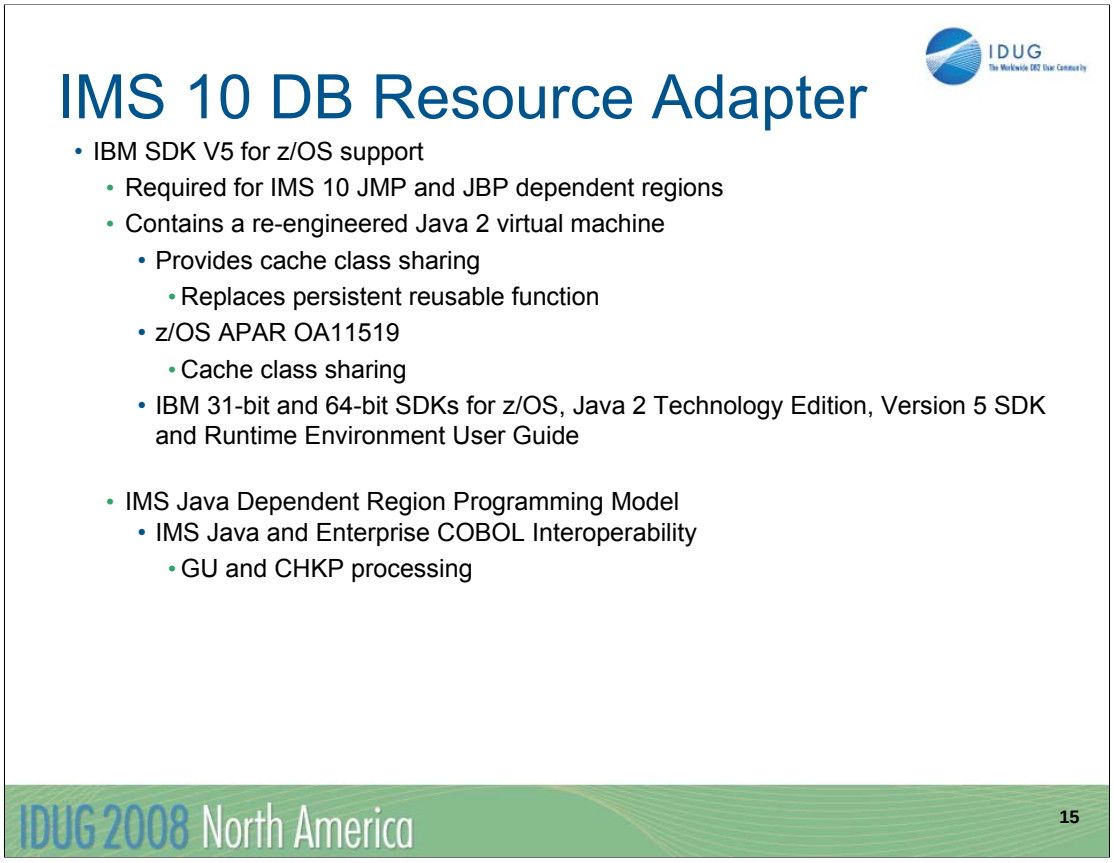
JVM Lifecycle

ResetJavaVM: ... Reset Transient Heap



Persistent Reusable Java Virtual Machine restrictions

- Anything that doesn't make a JVM look as if it isn't being used for the first time:
  - Modifying system properties
  - Changing static variables
  - Starting threads
  - Creating processes
- If performed by application it will make JVM unresettable



z/OS delivers a complete Java 2 Software Developer Kit (SDK).

Previous versions of the SDK provided a reusable function to support transactional runtime environments like IMS. This capability allowed the Java Virtual Machine (JVM) to be initialized during IMS Java dependent region startup and to be “re-set” after the IMS application program completed processing. This avoided the overhead of loading the JVM for each IMS appplication program schedule.

The new SDK provides a Class Sharing capability to replace the persistent reusable function.

z/OS APAR OA11519 is recommended for cache class sharing.

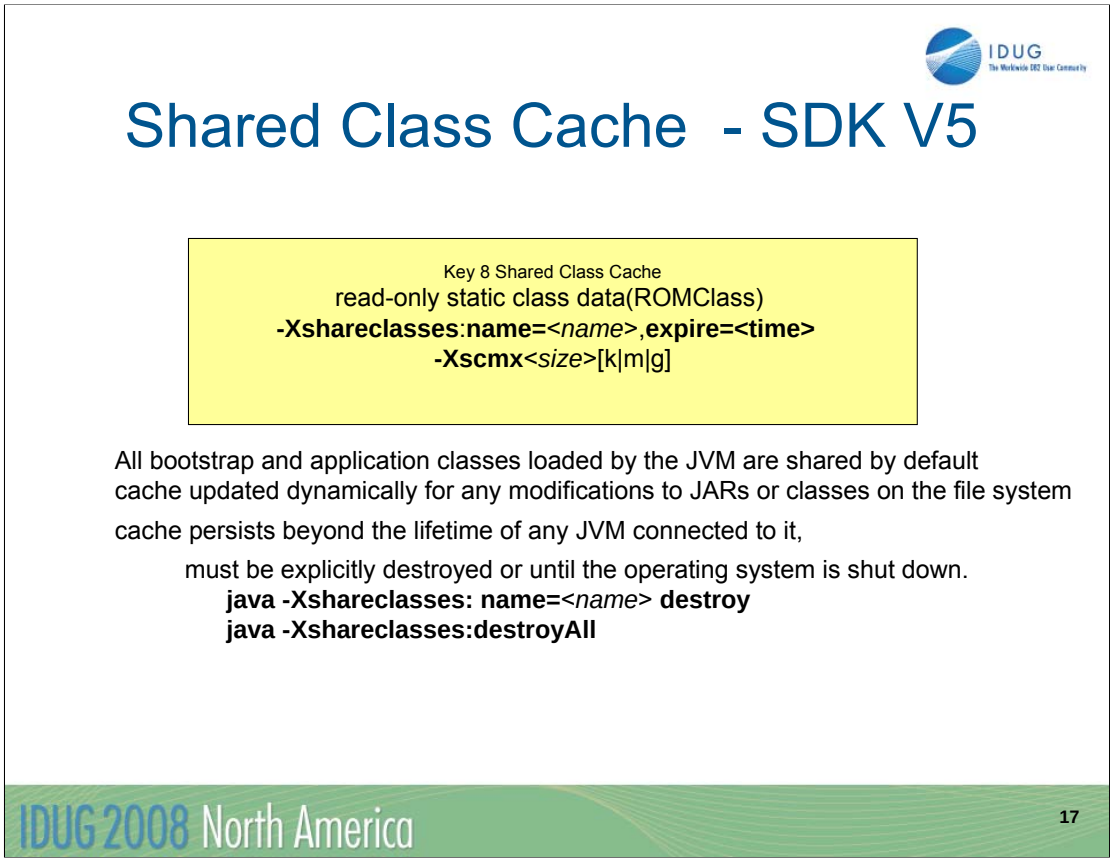
The User Guide can be downloaded from the specified URL.

IMS message GU/CHKP processing was altered for Java Dependent Regions (JDRs) when JDRs were initially implemented. For JDRs, an IMS message GU meant to get the message only, do not perform sync point processing. CHKP processing meant perform sync point processing only, do not get the next message. In the standard IMS model, message GU and CHKP meant perform sync point and retrieve the next input message.

For IMS V10, IMS JDRs GU/CHKP processing will be consistent with the IMS standard model.

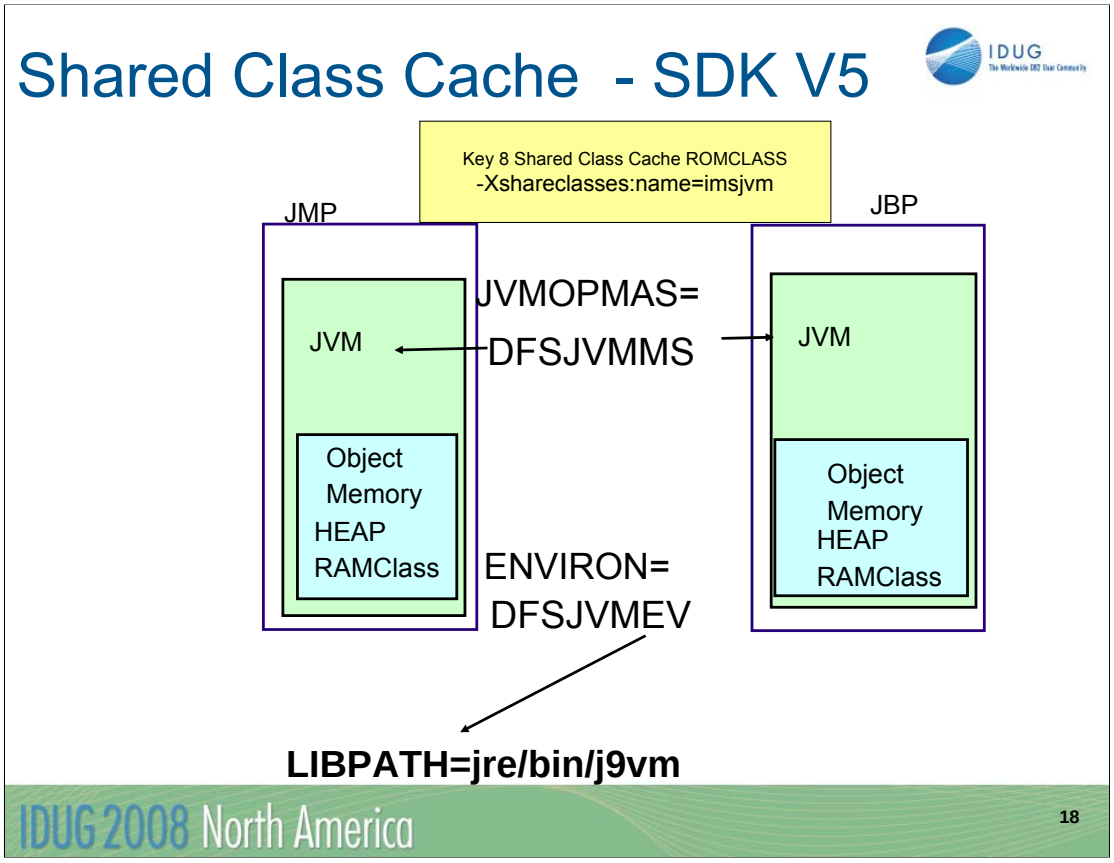
Where is SDK V5 for z/OS?

http://www-03.ibm.com/servers/eserver/zseries/software/java/



The shared class cache contains read-only static class data and metadata that describes the classes.

To enable class sharing use –Xshareclasses:-name= option when starting a JVM.See the Diagnostics Guide for more information.- **printAllStats** (Utility option) Displays detailed information about the contents of the cache specified in the **name=**<*name*> suboption.



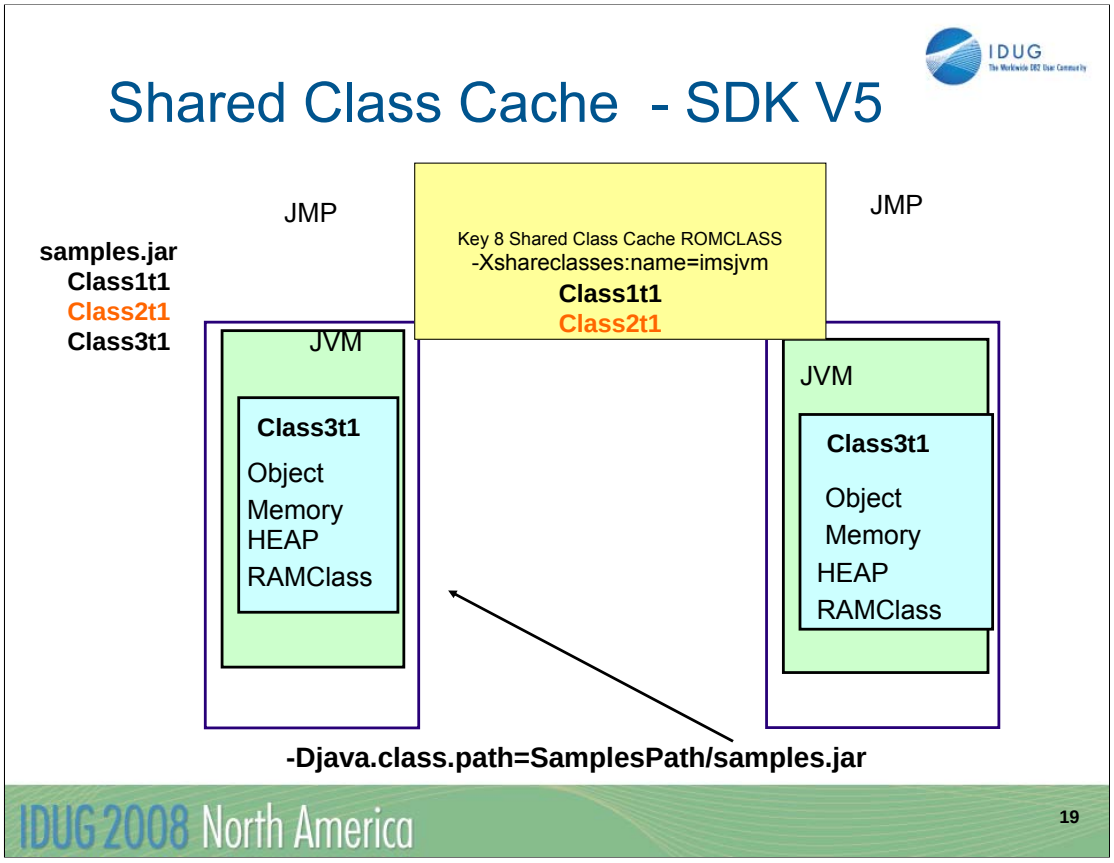
JVMOPMAS=DFSJVMMS

Specifies the JVM options

ENVIRON=DFSJVMEV

Must contain the pathname to JVM

Must contain the pathname to the IMS Java native code



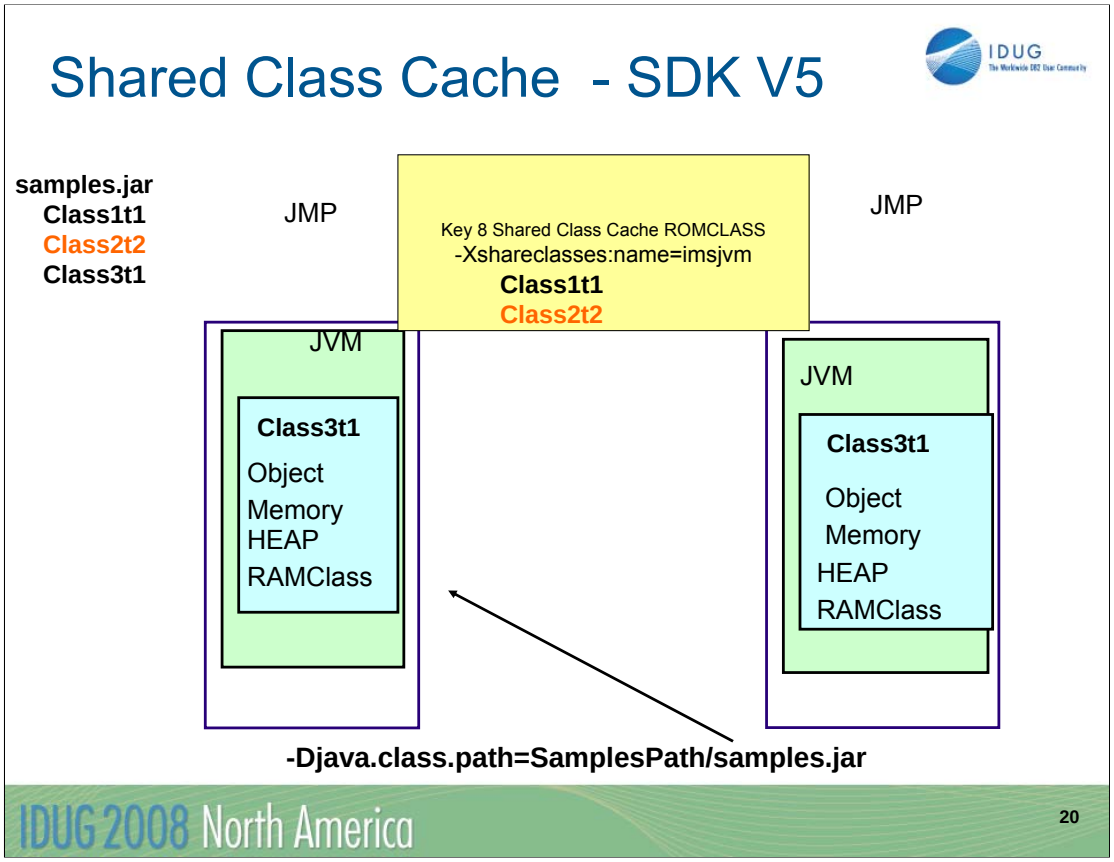
JVMOPMAS=DFSJVMMS

Specifies the JVM options

ENVIRON=DFSJVMEV

Must contain the pathname to JVM

Must contain the pathname to the IMS Java native code



JVMOPMAS=DFSJVMMS

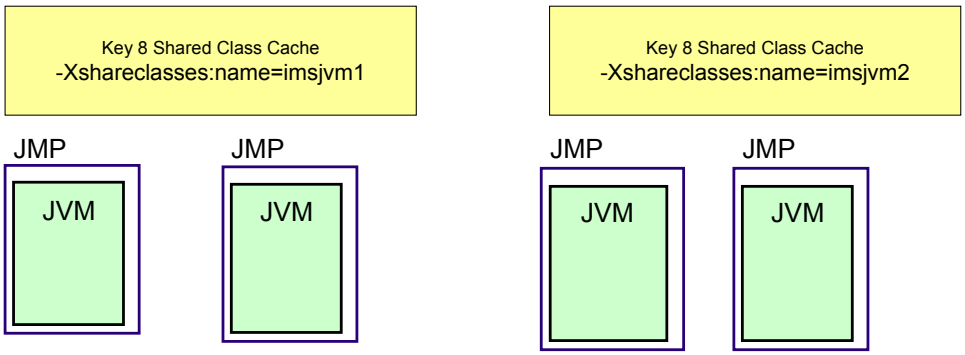
Specifies the JVM options

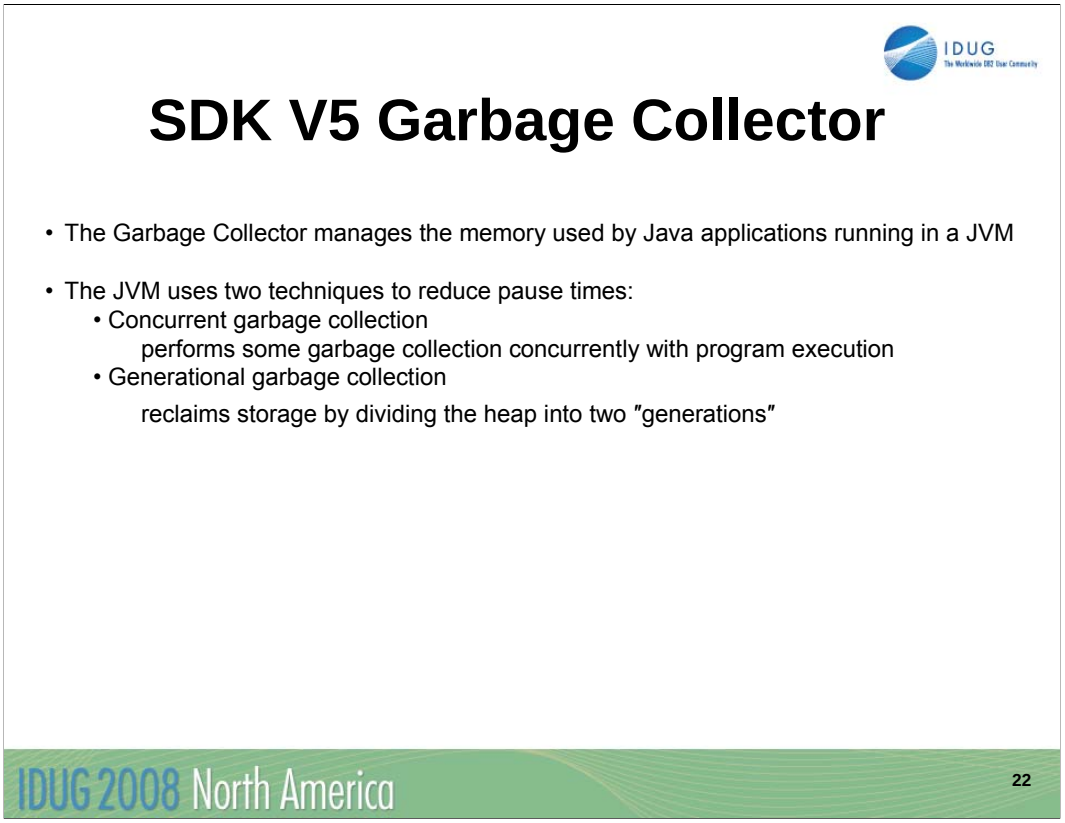
ENVIRON=DFSJVMEV

Must contain the pathname to JVM

Must contain the pathname to the IMS Java native code

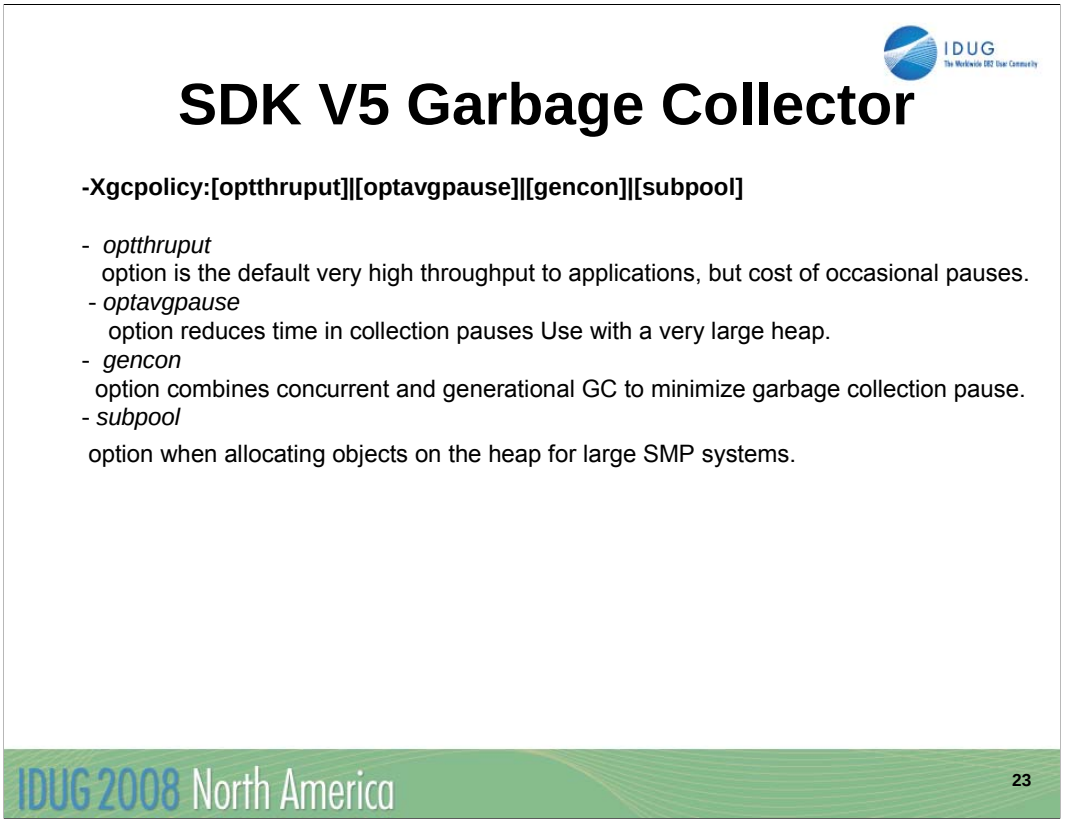
Multiple Shared Class Cache





The Garbage Collector manages the memory used by Java and by applications running within the JVM.

When an application's attempt to create an object cannot be satisfied immediately from the available space in the heap, the garbage collector is responsible for identifying unreferenced objects (garbage), deleting them, and returning the heap to a state in which the immediate and subsequent allocation requests can be satisfied quickly. Such garbage collection cycles introduce occasional unexpected pauses in the execution of application code.



-Xgcpolicy:[optthruput]|[optavgpause]|[gencon]|[subpool]

Controls the behavior of the Garbage Collector.

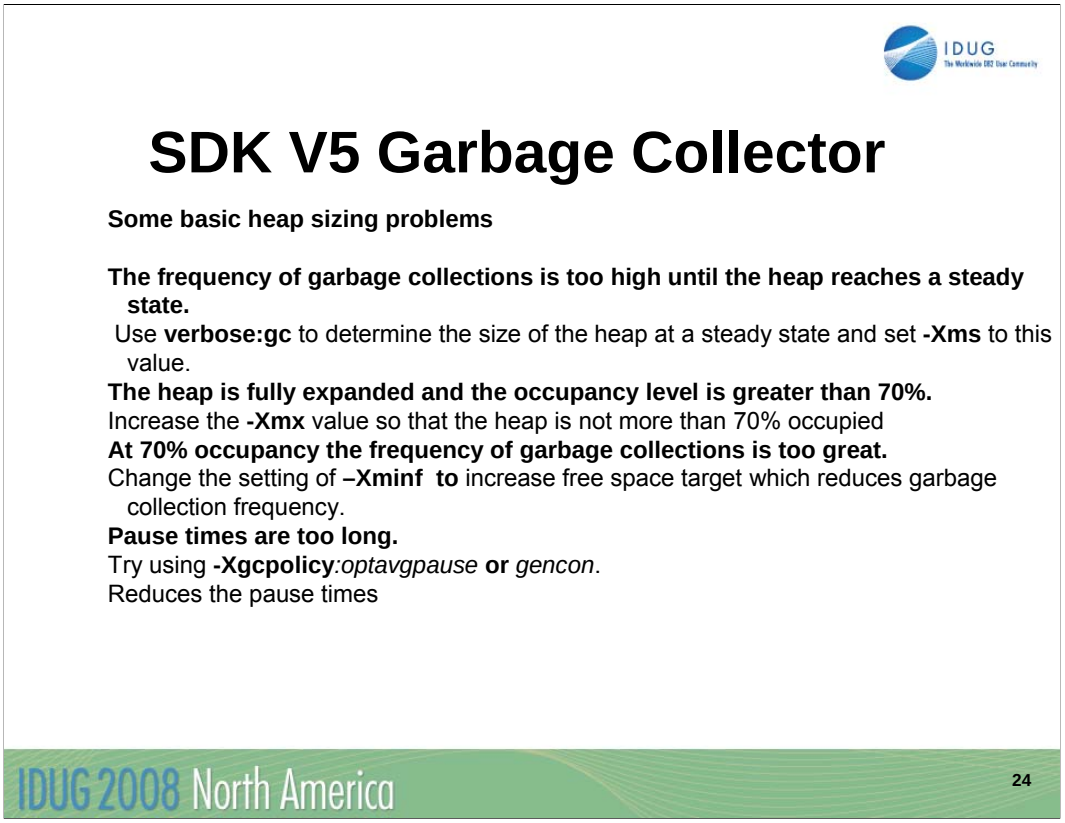
- The *optthruput* option is the default and delivers very high throughput to applications, but at the cost of occasional pauses.

- The *optavgpause* option

Enables concurrent mark with its default values. If you are having problems with erratic application response times that are caused by normal garbage collections, you can reduce those problems at the cost of some throughput, by using the *optavgpause* option.

- The *gencon* option requests the combined use of concurrent and generational GC to help minimize the time that is spent in any garbage collection pause.

- The *subpool* option uses an improved object allocation algorithm to achieve better performance when allocating objects on the heap. Disables concurrent mark. It uses an improved object allocation algorithm to achieve better performance when allocating objects on the heap. This option might improve performance on SMP (**Symmetric multiprocessing**) systems with 16 or more processors.



For the majority of applications, the default settings work well. The heap expands until it reaches a steady state, then remains in that state, which should give a heap occupancy (the amount of live data on the heap at any given time) of 70%. At this level, the frequency and pause time of garbage collection should be acceptable. For some applications, the default settings might not give the best results. Listed here are some problems that might occur, and some suggested actions that you can take. Use **verbose:gc** to help you monitor the heap.

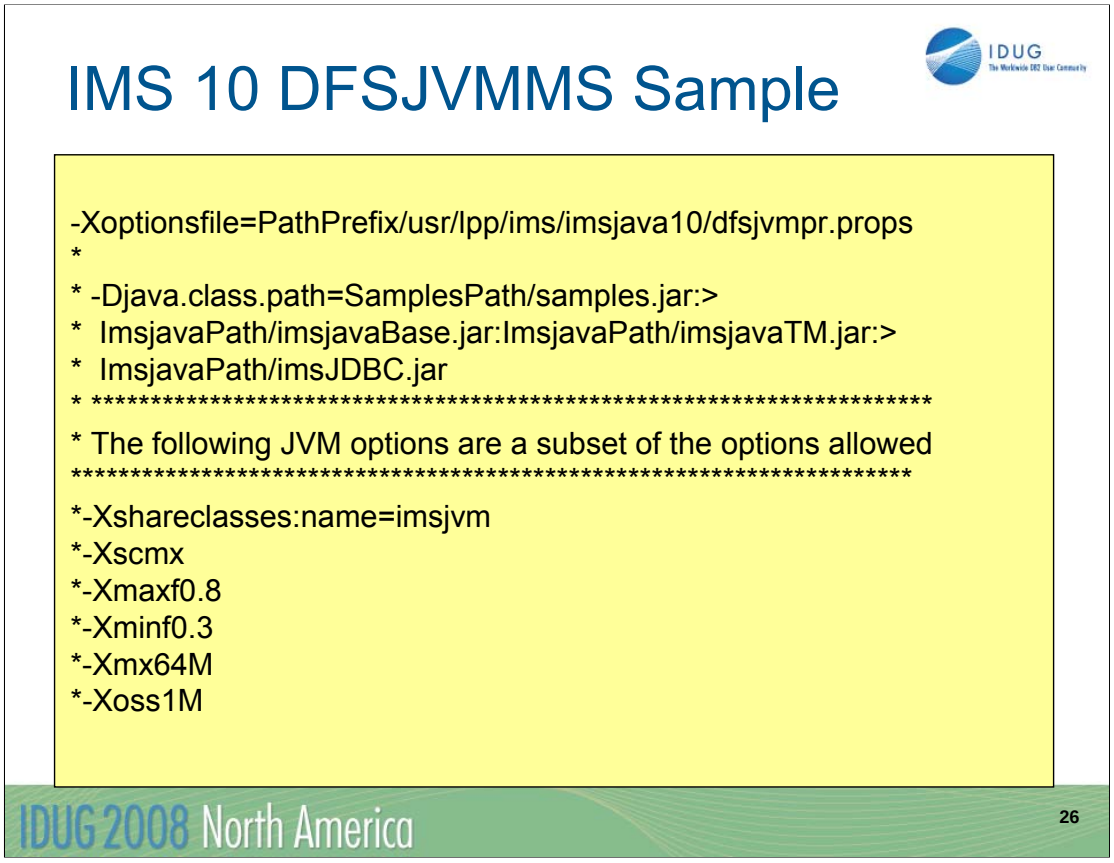
-Xms<*size*> Sets the initial Java heap size.

-Xmx<*size*> Sets maximum Java heap size.

-Xminf<*size*> Specifies the minimum percentage of heap that should be free after a garbage collection. If the free space falls below this amount, the JVM attempts to expand the heap

Migration to SDK V5

- IMS V9 APAR - **PK37843** provides SDK V5 support
- The shared library libjvm.so is installed in directory jre/bin/j9vm
  - Previous versions directory jre/bin/classic
- **-verbose:gc** data is now formatted as XML
- **-Xresettable**, is not supported



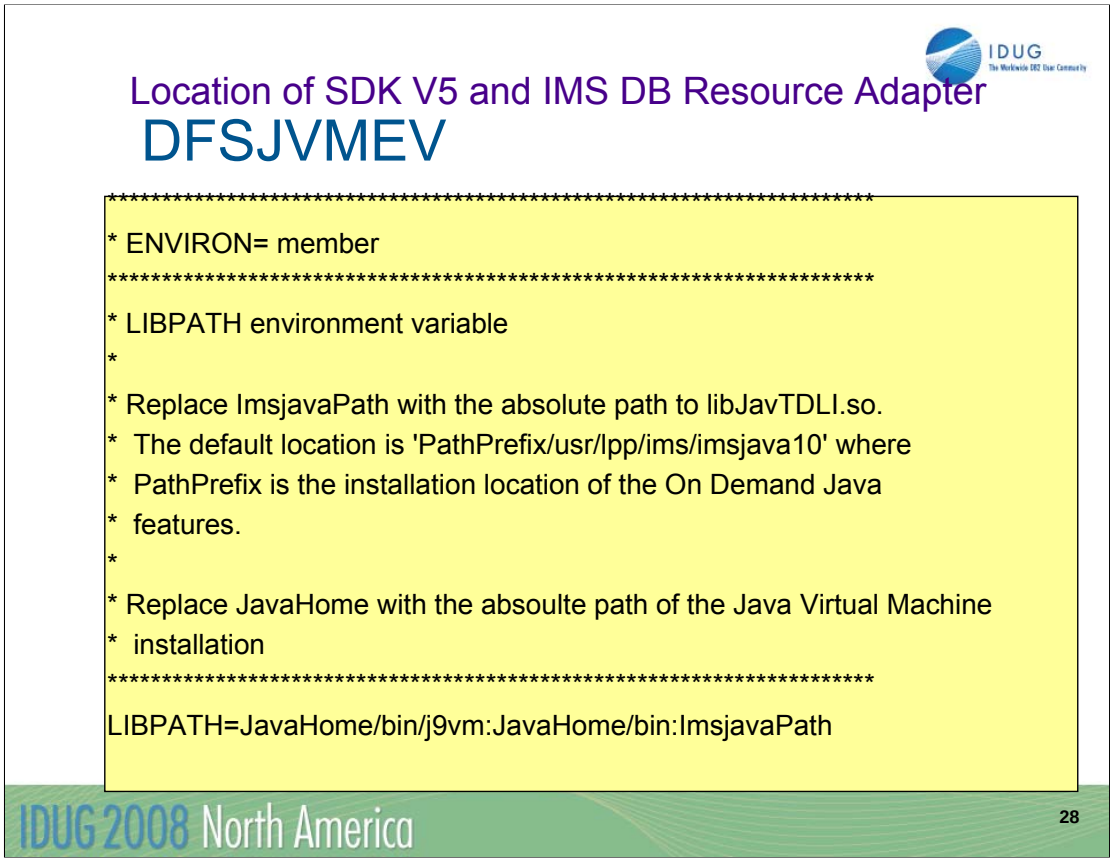
IMS provides the DFSJVMMS proclib member for the JVMOPMAS= parameter.

IMS 10 dfsjvmpr.props

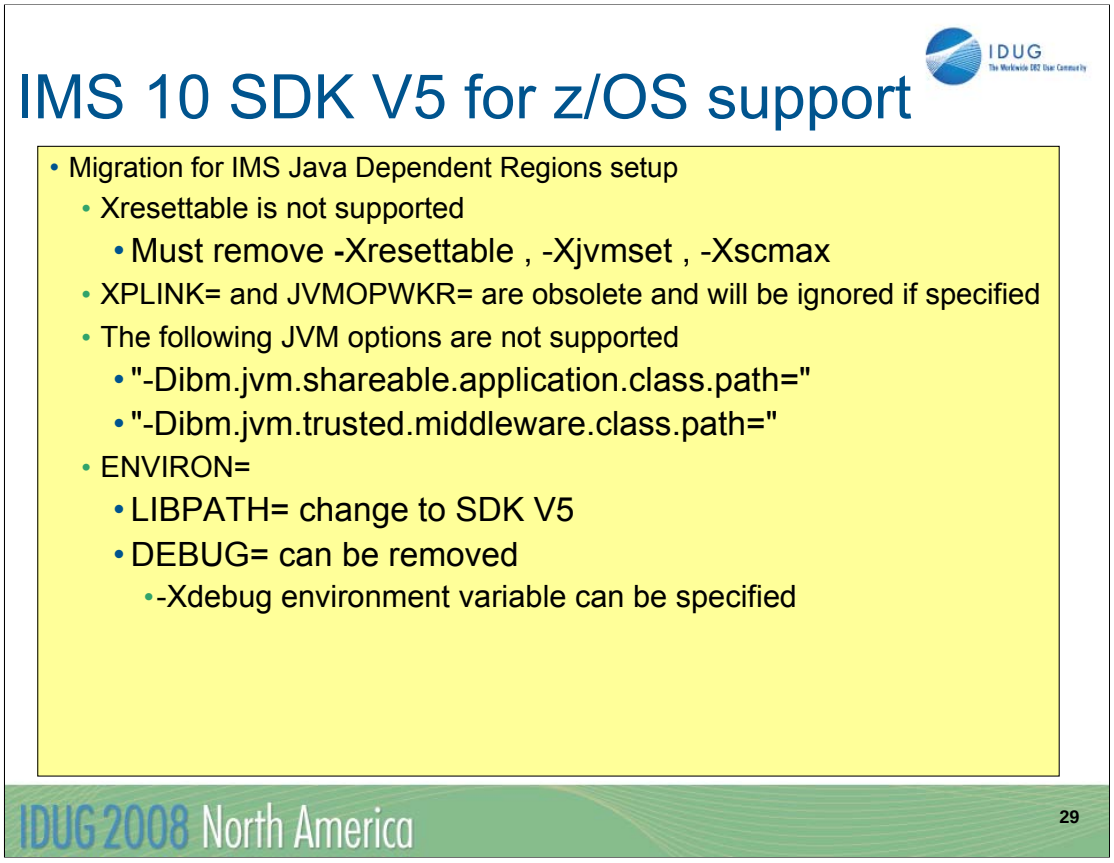
```
-Xoptionsfile=PathPrefix/usr/lpp/ims/imsjava10/dfsjvmpr.props
#   IMS JDR options file
# IMS DB Resource Adapter and IMS Applications path
-Djava.class.path=SamplesPath/samples.jar:>
  ImsjavaPath/imsjavaBase.jar:ImsjavaPath/imsjavaTM.jar:>
  ImsjavaPath/imsJDBC.jar
# JVM options
-Xshareclasses:name=imsjvm
-Xscmx32M
-Xmaxf0.8
-Xminf0.3
-Xmx64M
-Xoss1M
```


In this example dfsjvmpr.props is the file that specifies the -Djava.class.path to specify the path to the IMS DB Resource Adapter, the IMS Java sample application and user written IMS Java application programs.

Note –classpath cannot be specified



IMS provides the DFSJVMEV proclib member for the ENVIRON= parameter. The IMS V10 member has been updated to specify the LIBPATH= value for the to IBM SDK V5 for z/OS JVM and the IMS DB Resouce Adapter.



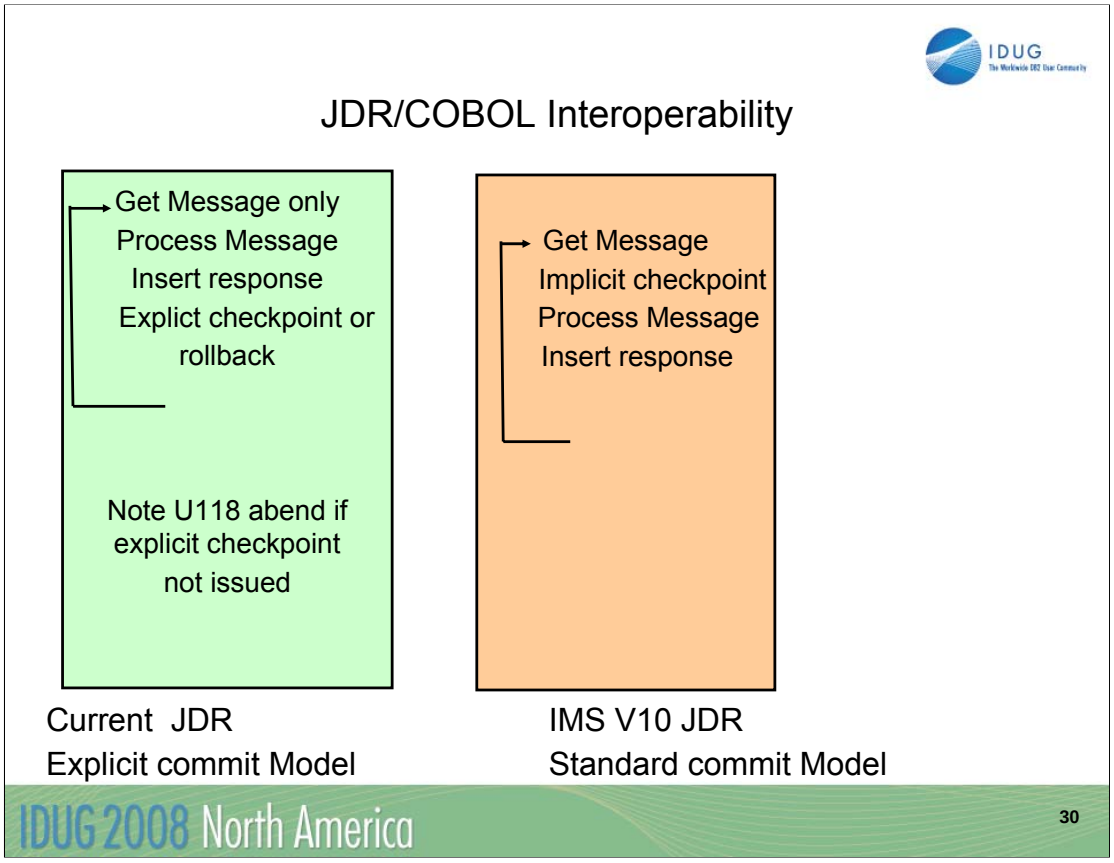
The serial reusability feature of the IBM SDK for z/OS, version 1.4.2 (31-bit) and earlier is not supported. If you specify –Xresettable -Xjvmset or -Xscmax the JVM will issue an error message and will not start.

The JVM shared library libjvm.so is installed in directory jre/bin/j9vm.

The IMS Java dependent region ENVIRON= member needs to have the LIBPATH changed to LIBPATH=JavaHome/bin/j9vm:JavaHome/bin:ImsjavaPath

ENVIRON= can now be used to specify any Java environment variables. This can be used to replace the DEBUG= parameter. The Java environment variables are presented as Java system properties at runtime and are therefore accessible by a Java application running in the JDR.

-Xdebug starts the JVM with the debugger enabled. By default, the debugger is disabled



The current IMS Java Dependent Region programming requires an explicit checkpoint/rollback call using the IMS DB Resource Adapter Transaction class

IMSTransaction.getTransaction().commit()

IMSTransaction.getTransaction().rollback().

For IMS V10 the IMS Java application does not need to perform an explicit commit before obtaining the next input message.

IBM SDK V5 for z/OS support

- Migration for IMS Java Dependent Regions programming model
  - Current applications do not need to change unless:
    - Applications that use U118 Abend for ROLLBACK
    - Applications that use explicit CHKP call
      - Now receives next input message after CHKP call

An IMS Java application that does not issue explicit ROLLBACK but uses U118 abend for rollback processing must be changed. An IMS Java application that uses the IMS Java hierarchical database interface for CHKP call processing must be able to process the next message off the message queue.

IBM SDK V5 for z/OS support

- Migration for IMS Java Application Programs
  - com.ibm.ims.application.IMSApplication is deprecated
    - Recommend begin changing existing IMS Java applications
  - com.ibm.ims.base.DLISecondaryIndexInfo class has been removed
    - DLIModel generated metadata
      - No impact unless application explicitly used the class
  - com.ibm.ims.db.SecondaryIndexInfo class has been renamed
      - com.ibm.ims.base.SecondaryIndexInfo class
        - DLIModel generated meta data
        - No impact unless application explicitly used the class

The IMS DB Resource Adapter class com.ibm.ims.application.IMSApplication is deprecated in IMS Version 10.

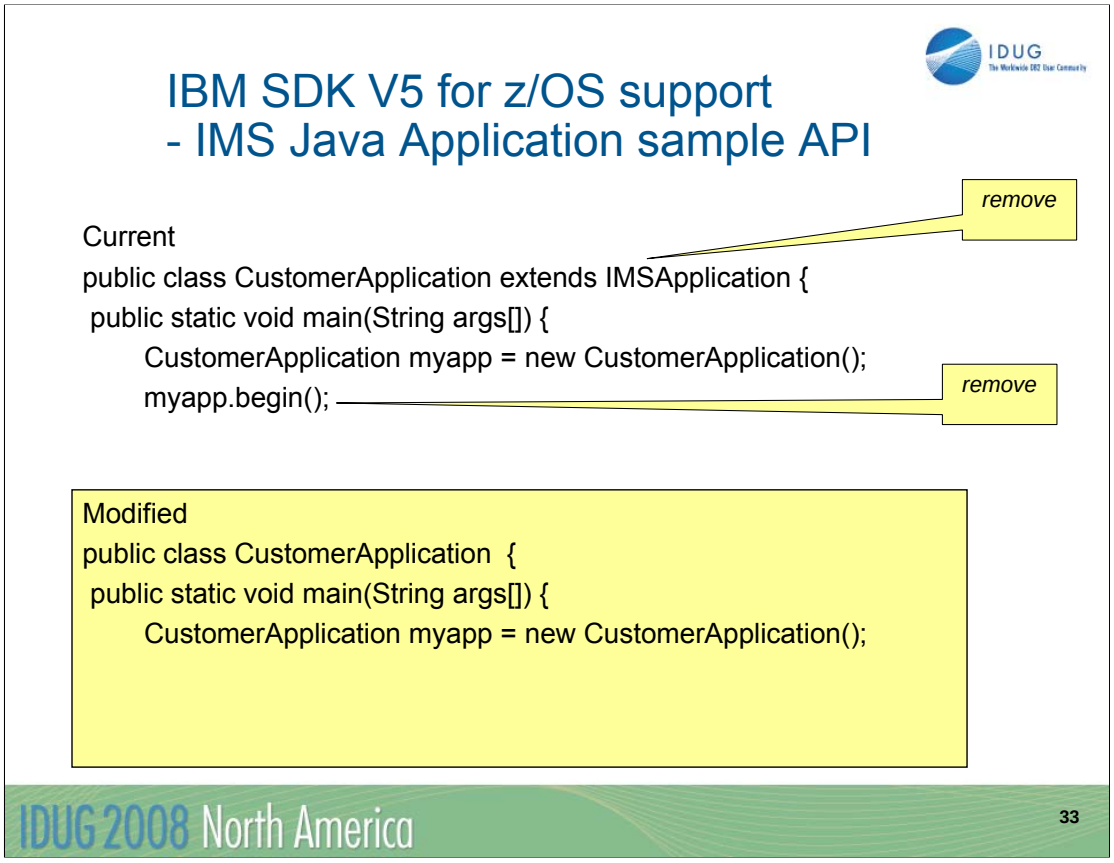
A class marked as deprecated is considered obsolete but still works in IMS Version 10.

You do not have to change your applications, but it is highly recommended.

The com.ibm.ims.base.DLISecondaryIndexInfo class has been removed from the library. This will only impact you if you did not use DLIModel to generate the meta data classes (the database view) or if you use the DLISecondaryIndexInfo class explicitly in your code.

The com.ibm.ims.db.SecondaryIndexInfo class has been renamed to com.ibm.ims.base.SecondaryIndexInfo.

You will be impacted only if you use the class directly in your code. The metadata that is generated by the DLIModel utility is not affected.



The applications that subclass IMSApplication can be modified as follows:

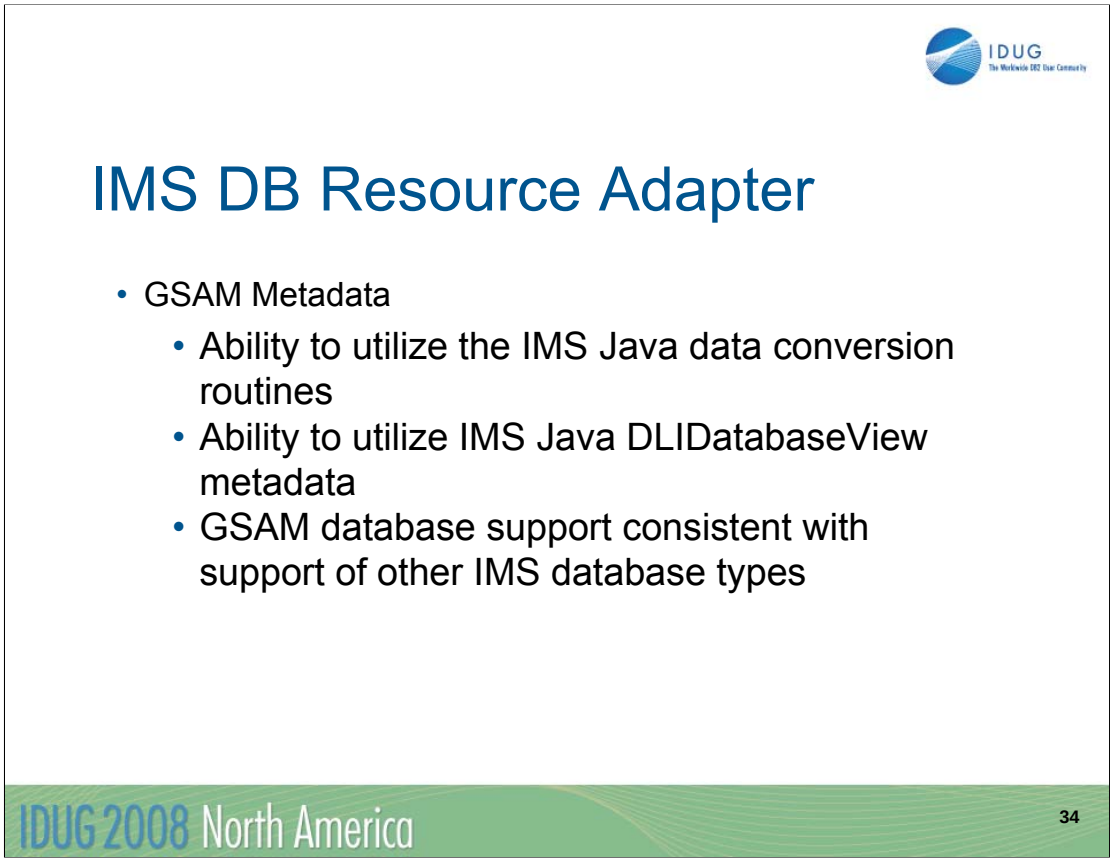
Remove the "extends IMSApplication" from the class declaration line.

For example, "public class CustomerApplication extends IMSApplication"

becomes "public class CustomerApplication".

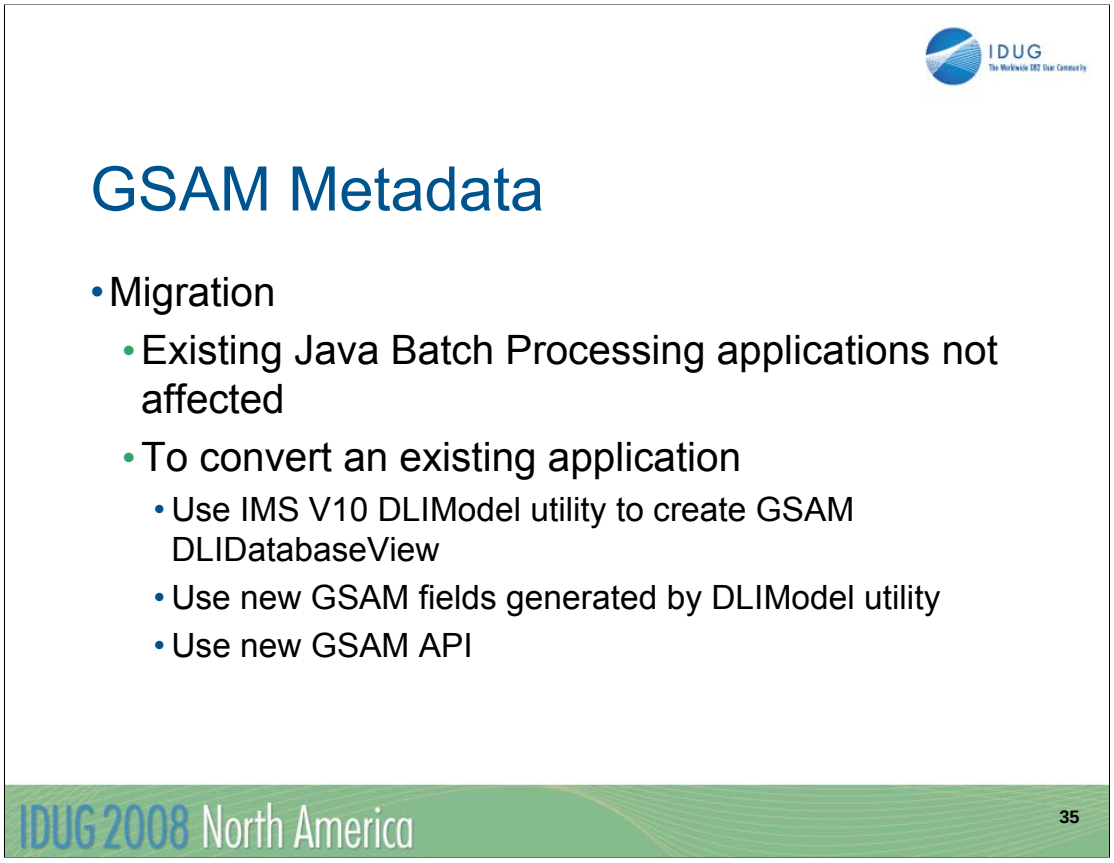
The main method of the application no longer needs to call the IMSApplication.begin() method.

Instead, the main method can directly call the public void doBegin() method or simply move the logic from the doBegin() method to the main method and delete the doBegin() method.



IMS DB Resource Adapter support for GSAM was enabled by Small Programming Enhancement PQ93785/UQ93241. This provided a Java Batch Processing application program with a Java API to open read write and close GSAM databases. However , the application was required to know the complete details of the GSAM database

Given a DLIDatabaseView name, the IMS Java class library will use the metadata information to capture the correct sequence of bytes in the record, read that sequence of bytes, and convert that sequence of bytes into the appropriate data type as defined by the metadata



Java Batch Processing applications that use the IMS V9 GSAM database support will not be affected

The DLIModel utility for IMS V10 provides new GSAM record & field options. This will be described in the DLIModel utility section.

DFSJMP Procedure

- Starts a JMP region
- The existing APPLFE=, DBLDL=, **PRLD=**,VSFX=, and VFREE= parameters on the DFSMPR procedure are not supported on the DFSJMP procedure.
  - **JVMOPMAS=**
    - Specifies name of IMS.PROCLIB member that contains the JVM options
    - Required. If not present, region abends with ABENDU0101
  - **XPLINK=** this parameter is ignored and internally set to XPLINK=Y.
  - **ENVIRON=**
    - Specifies name of IMS.PROCLIB member that contains the LIBPATH= environment variable specification
    - Required. If not present, region abends with ABENDU0101
  - **DFSJVMAP**
    - optional PDS member
    - maps 1-8 byte uppercase name to OMVS path name
    - read during IMS Java application program scheduling
      - do not need to shut down region to make changes effective

XPLINK=

Specifies whether (Y) or not (N) to initialize the JVM LE enclave with XPLINK(ON). You must specify XPLINK=Y if you are running SDK 1.4.1 or above. XPLINK=N is the default. For Java dependent regions, this parameter is ignored and internally set to XPLINK=Y.

Attention: For performance improvement purposes, a DFSINTxx member might include the Language Environment (LE) Library Routine Retention feature. Library Routine Retention is not supported when XPLINK=Y is specified. For more information about Library Routine Retention, see the *z/OS Language Environment Programming Guide, SA22–7561*.

DFSJBP Procedure

- Starts a JBP region
- The existing IN= and PRLD= parameters on the IMSBATCH procedure are not supported on the DFSJBP procedure.
  - **JVMOPMAS=**
    - Specifies name of IMS.PROCLIB member that contains the JVM options for the master JVM
    - Required. If not present, region abends with ABENDU0101
  - **XPLINK=** this parameter is ignored and internally set to XPLINK=Y
  - **ENVIRON=**
    - Specifies name of IMS.PROCLIB member that contains the LIBPATH= environment variable specification
    - Required. If not present, region abends with ABENDU0101
  - **MBR=**
    - actual name of the Java application or a symbolic for the actual name of the Java application
  - **DFSJVMAP**
    - optional PDS member
    - maps 1-8 byte uppercase name to OMVS path name
    - read during IMS Java application program scheduling

IMS JVMs

REGION=0M

//* HFS path for Java stdout System.out.print()
//JAVAOUT DD PATH='/region/JVM.out'

//* HFS path for Java stderr System.err.print()
//JAVAERR DD PATH='/region/JVM.err'

APPLFE=, DBLDL=, PRLD=, VSFX=, and VFREE= are not supported for JMP

IN= and PRLD= are not supported for JBP

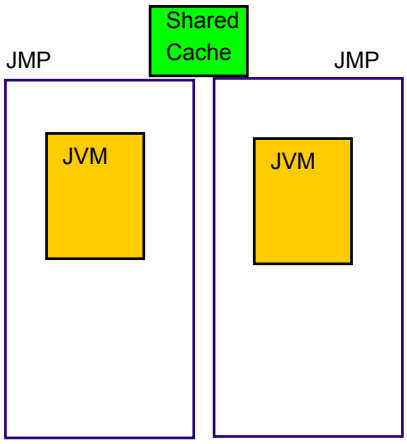
If you do not specify the JAVAOUT DD or the JAVAERR DD statements, System.out.print() and System.err.print() calls in the IMS Java application will not generate any output.

IMS JVMs

IMS.PROCLIB(DFSJVMAP)

```
IMSJavaPgm1.java
package imsjava.appl.jmp;
import com.ibm.ims.base.*;
import com.ibm.ims.application.*;
import com.ibm.ims.db.*;
import java.sql.*;
public class IMSJavaPgm1 extends IMSApplication {

IMSJavaPgm2.java
package imsjava2.appl.jmp;
import com.ibm.ims.base.*;
import com.ibm.ims.application.*;
import com.ibm.ims.db.*;
import java.sql.*;
public class IMSJavaPgm2 extends IMSApplication {
```



OMVS application path

ims/java/applications/imsjava/appl/jmp/IMSJavaPgm1.class

ims/java/applications/imsjava2/appl/jmp/IMSJavaPgm2.jar

IMS JVMs

DFSJVMAP is a supplied member in the IMS sample library and specifies the path to an IMS Java application

APPLCTN PSB=JAVAPGM1

PSBGEN LANG=JAVA,PSBNAME=JAVAPGM1

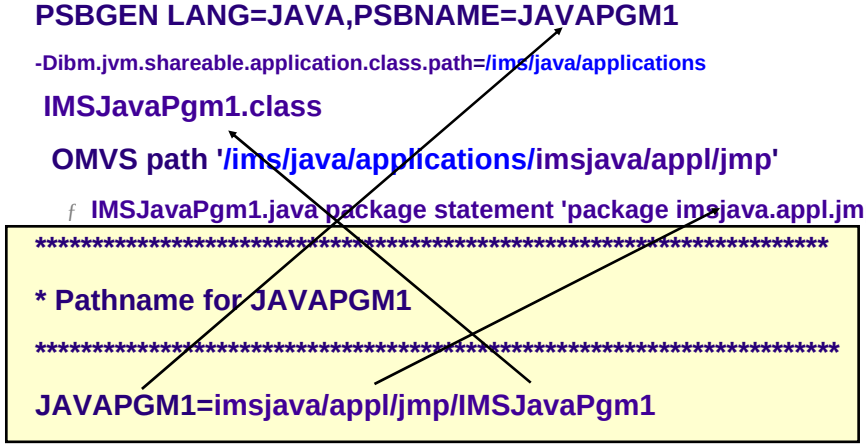
-Dibm.jvm.shareable.application.class.path=/ims/java/applications

IMSJavaPgm1.class

OMVS path '/ims/java/applications/imsjava/appl/jmp'

- **IMSJavaPgm1.java package statement 'package imsjava.appl.jmp ':**

```
**************************************************************************
* Pathname for JAVAPGM1
**************************************************************************
JAVAPGM1=imsjava/appl/jmp/IMSJavaPgm1
```



IMS JVMs

DFSJVMAP is a supplied member in the IMS sample library and specifies the path to an IMS Java application

APPLCTN PSB=JAVAPGM2

PSBGEN LANG=JAVA,PSBNAME=JAVAPGM2

-Dibm.jvm.shareable.application.class.path=/ims/java/applications/IMSJavaPgm2.jar

OMVS path '/ims/java/applications/imsjava2/appl/jmp'

- IMSJavaPgm2.java package statement 'package imsjava2.appl.jmp ':

```
**************************************************************************
* Pathname for JAVAPGM2
**************************************************************************
JAVAPGM2=imsjava2/appl/jmp/IMSJavaPgm2
```

Java Dependent Regions - ABENDU0101

Description

- An error occurred during Java dependent region processing

Analysis

- For all instances of this abend, the user should examine the dependent region JOB output for the cause of the failure by searching on the character string "DFSJVM00:" which can indicate:
  - LE error messages
  - Caught thrown exceptions from the IMS Java application
  - JVM error messages

Java Dependent Regions - COBOL/JAVA

A combination COBOL and Java application that runs in an IMS Java dependent region:

- Call a COBOL method from an IMS Java application
  A Java program cannot call procedural COBOL programs directly.
  To reuse existing COBOL IMS code:

    Restructure the COBOL code as a method in a COBOL class
    or
    Write a COBOL class definition and method that serves as a wrapper code
    that can use COBOL CALL statements to access procedural COBOL programs

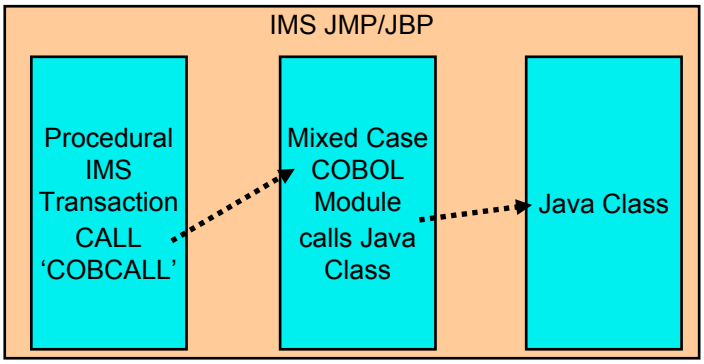
- Build a mixed COBOL and Java application
  start main method of a COBOL class that invokes Java routines.

Note: COBOL class methods are implemented in native code, the JVM cannot be reset after a transaction that uses COBOL routines runs

Note SDK V5 does not provide resettable JVM

COBOL calls Java in JMP/JBP

- IMS Transaction
  - PSB with LANG=ASSEM
- Scenario
  - Procedural Application Calls COBOL wrapper that calls Java



BTS

- **IMS Java applications**
- The IMS™ Java™ application can run under the JBP region by using standalone JVM. To do this specify KW=JBP in BTS.
- To run Java application programs with BTS:
- You must specify APARM=BTS on the EXEC parameter.
- You must specify ASM or CBL on the ./T LANG= keyword.

Java Dependent Regions - DB2

DB2 RRSAF

1) SSM member of IMS.PROCLIB for DB2 subsystem example:
SST=DB2,SSN=DB2E,COORD=RRS

2) IMS Control RRS=Y

2) Add DB2 to class path

-Djava.class.path= > /usr/lpp/db2/db2710/classes: > /usr/lpp/db2/db2710/classes/db2j2classes.zip

3) Add DB2 to libpath

LIBPATH=/usr/lpp/db2/db2710/lib

4) Add the DB2 library to JMP region with the DFSDB2AF DD (which must all be APF authorized libraries) example:

//DFSDB2AF DD DISP=SHR,DSN=IMS.SDFSRESL
// DD DISP=SHR,DSN=DSNxxx.DSNLOAD

DB2 JDBC/SQLJ 2.0 driver or JDBC/SQLJ 1.2 driver

DB2 Recoverable Resource Manager Services Attachment Facility (RRSAF)

DB2 JDBC/SQLJ 2.0 driver or the DB2 JDBC/SQLJ 1.2 driver

RRS required

COBOL cannot access DB2 in a JMP or JBP region.

Java Dependent Regions - DB2

IMS control region

1) prepares for access to DB2 for z/OS during initialization
2) does not affect the execution of other types of dependent regions
3) all Java dependent regions that start will build an access thread to DB2 for z/OS
   - DB2 for z/OS library is defined in the Java dependent region JCL
4) prevent access threads from being built by stopping access to the DB2 for z/OS
   IMS Command - /STO SUBSYS

Java application programs running in IMS dependent regions can access DB2 for z/OS under syncpoint control of RRS if a DB2 Attach Facility definition is included when the IMS control region is started. The initialization processing of the IMS control region prepares for access to DB2 for z/OS. When Java dependent regions are subsequently started, application programs in those regions can make direct calls to both DB2 for z/OS and IMS.

Initialization of the DB2 Attach Facility does not affect the execution of other types of dependent regions. The DB2 Attach Facility definition can be retained in the IMS.PROCLIB member, even if Java dependent regions are not used. If a DB2 Attach Facility definition exists in the IMS.PROCLIB member, and the DB2 for z/OS library is defined in the Java dependent region JCL, all Java dependent regions that start will build an access thread to DB2 for z/OS. If the DB2 Attach Facility definition exists, but the Java dependent regions do not require access to DB2 for z/OS, you can prevent access threads from being built by stopping access to the DB2 for z/OS system. Use the /STO SUBSYS command, which stays in effect until a /STA SUBSYS command is subsequently issued.

IBM SDK V5 for z/OS support

- Benefits
  - IMS participates in industry standards
  - Can be used with existing IMS Java applications
  - Reduces virtual memory consumption
  - Reduces JVM startup time

Java Dependent Regions - Summary

IMS 10 Requires SDK 5

- IMS V9 – PK37843
- XPLINK=Y is the default for a JMP region and a JBP region

APPLCTN and PSBGEN macros changed to support

- LANG=JAVA

Two new IMS Dependent Region types

- JMP region type for message driven IMS Java applications
  - similiar to MPP
- JBP region type for non-message driven IMS Java applications
  - similiar to non-message driven BMP
- Supports mixed COBOL/Java language application program

DB2

- RRS required

J07
IMS 10 Java Dependent Regions
A Systems perspective

Kenny Blackman

IBM

kblackm@us.ibm.com