E33

IMS Java Dependent Regions An externals to internals perspective

Ken Blackman
kblackm@us.ibm.com

IMS technical conference

Las Vegas, NV **September 15 - September 18, 2003**

cm650b

RETURN TO INDEX

Terminology and Trademarks

Terminology

- JDK - Java Development Kit
- JVM - Java Virtual Machine
- EAB -Enterprise Access Builder
- EJB - Enterprise Java Bean
- HPJ - High Performance Java
- JAR - Java Archive
- JDBC - JDBC

Trademarks

MVS/ESA
IMS/ESA*
DB2*
S/390*
ESA/390
IBM*
IBM COBOL for MVS
System/390*
CICS
CICS/ESA
JDBC
Java

* Trademarks followed by an asterisk (*) are registered.

Contents

Background

Java Virtual Machine (JVM)

- Provides Runtime environment for Java Programs
- Inefficient for transaction processing

High Performance Java (HPJ)

- Java Program does not require JVM
- Inefficient for creating executable Java Program
  - Not supported in IMS V8

Persistent Reusable JVM

- Provides Runtime environment for transaction processing Java Programs
- Simplifies creation of executable Java Program

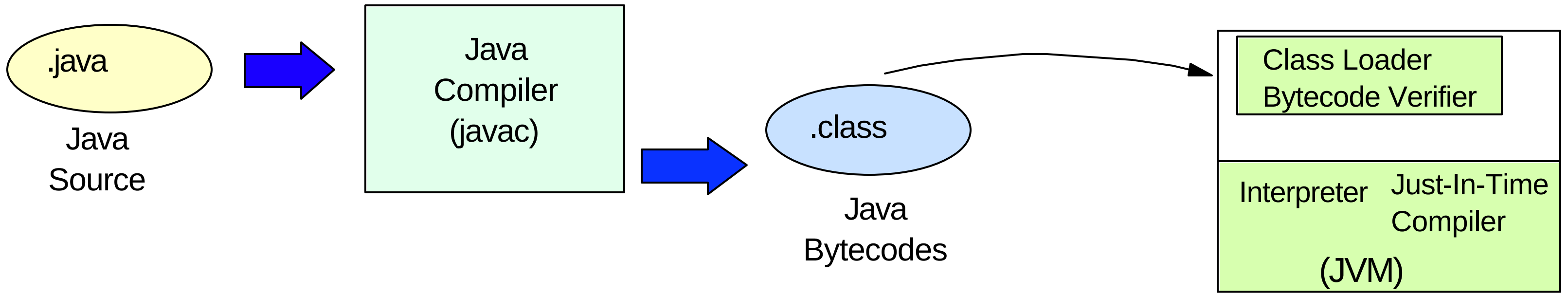
Runtime - JVM support

Persistent Reusable JVM - support added by APAR PQ53944

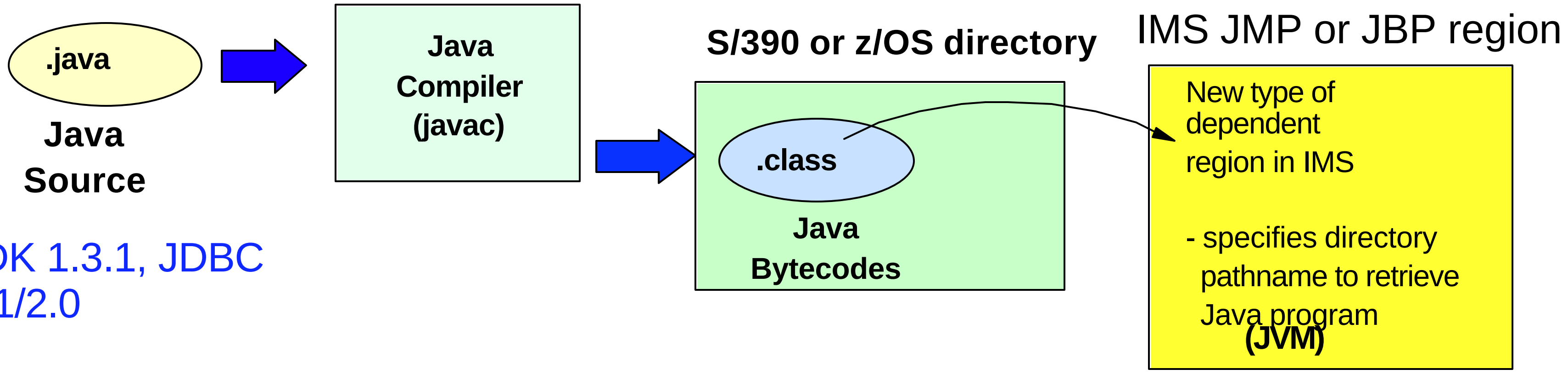
- Supports a Java Virtual Machine (JVM) in an IMS environment
  - Two new dependent regions in IMS - JMP and JBP

Value

- Supports the traditional Java execution model



Java bytecodes = machine code instructions for the JVM



JDK 1.3.1, JDBC 2.1/2.0

Persistent Reusable JVM

What is Persistent Reusable JVM?

- Enhancement to JDK 1.3 that implements:
  - Class cache storage management called a heap
  - Master JVM
    - establishes the JVM runtime environment
    - remains until IMS Dependent Region Termination
  - Worker JVM
    - Transaction processing JVM runtime environment
    - "reset" after transaction commit

      `IMS Java requires explicit commit`
- Dependent on JDK 1.3.1S
- Packaged in "IBM Developer Kit for OS/390, Java 2 Technology Edition"

Persistent Reusable Java Virtual Machine Classes

- System
  - System classes e. g. String, Thread
- Middleware
  - EJB container, JDBC driver
- Application
  - EJB, IMS program

Persistent Reusable Java Virtual Machine Classes

- Purpose:
  - Run multiple applications serially in the same JVM
  - Only load, verify & JIT classes once
  - Complementary to shared classes support
- System classes & objects persist
- Middleware classes & objects persist
- Shareable Application classes may persist
  - Loaded during region initialization
    - Similar to doing IMS MPP Preload
- Non-Shareable Application classes
  - Loaded during transaction scheduling
- Application objects cleared

Persistent Reusable Java Virtual Machine considerations

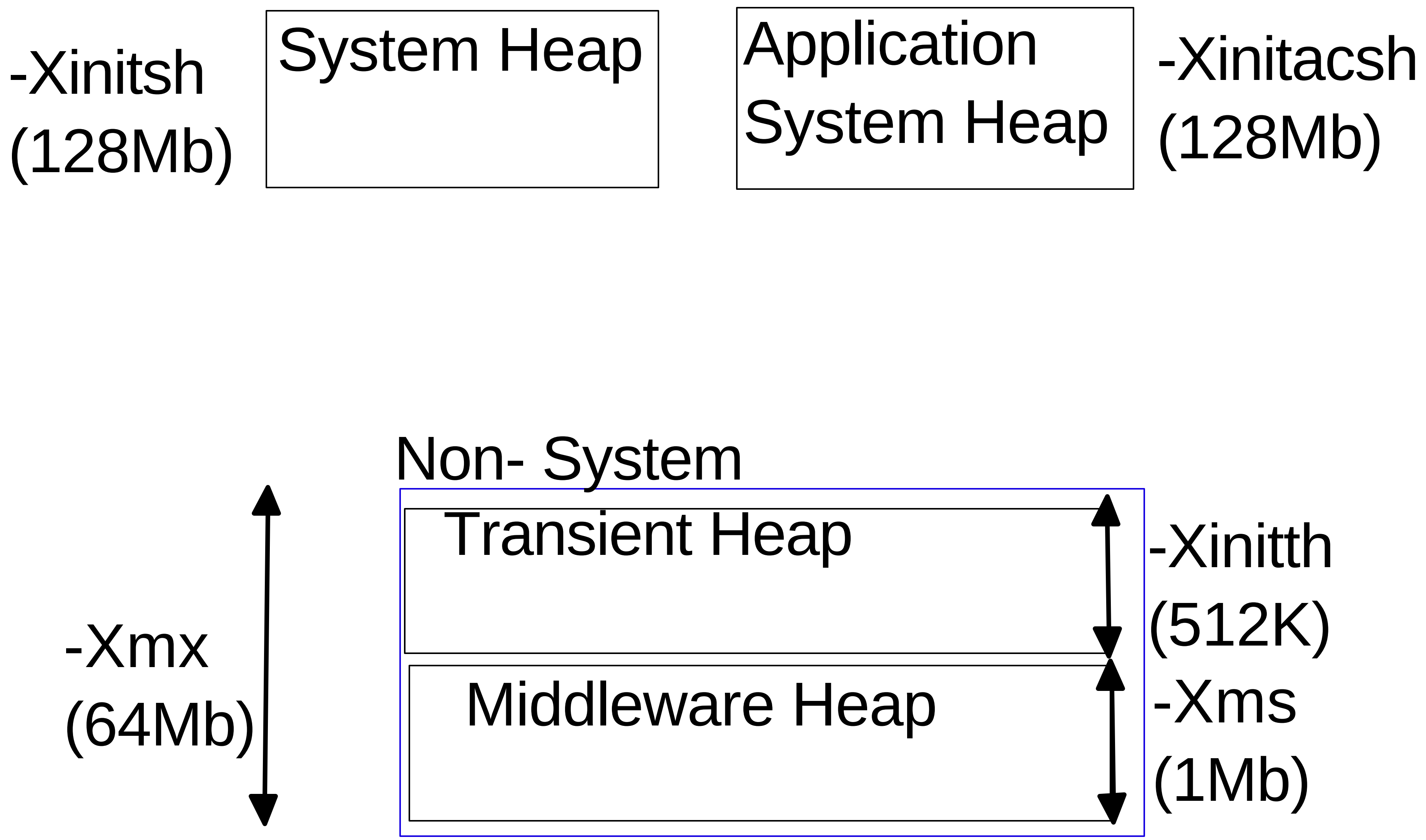
- Anything that doesn't make a JVM look as if it isn't being used for the first time:
  - Modifying system properties
  - Changing static variables
  - Starting threads
  - Creating processes
- If performed by application it will make JVM unresettable
- Rules align well with EJB coding standards

Persistent Reusable Java Virtual Machine Storage

- System Heap
  - Classes (System & Middleware)
  - Never Garbage Collected
- Shareable Application System Heap
  - Classes (Application)
  - Never Garbage Collected
- Non- system Heap/ Java Heap
  - Java objects
  - Normal Garbage Collection
- Middleware Heap
  - Middleware objects
  - Normal Garbage Collection
- Transient Heap
  - Application objects
  - Reset(discarded)

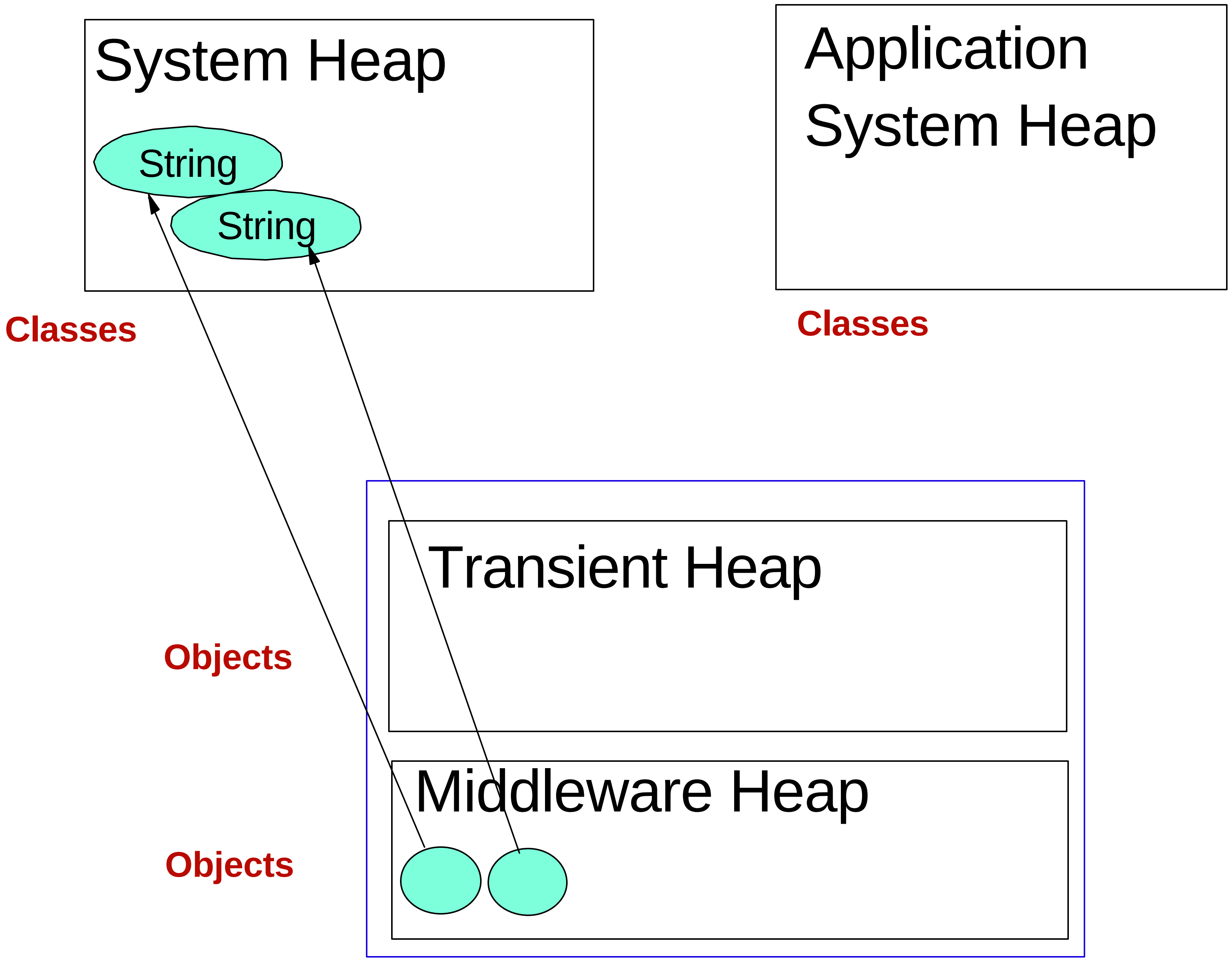
HEAP - contiguous piece of storage obtained at JVM initialization

Persistent Reusable Java Virtual Machine
Storage Settings
LE Subpools 0 and 1



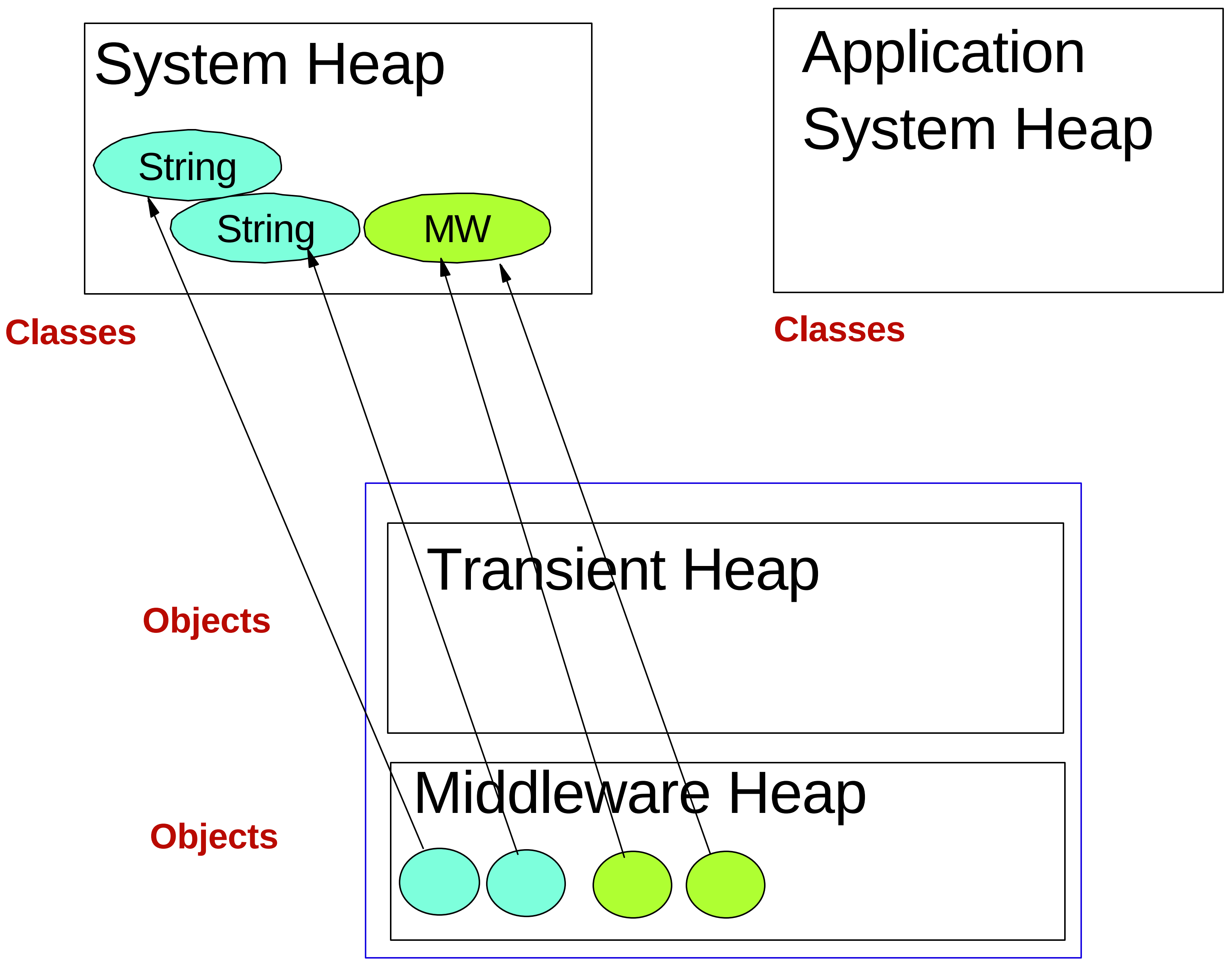
JVM Lifecycle

JVM initialization



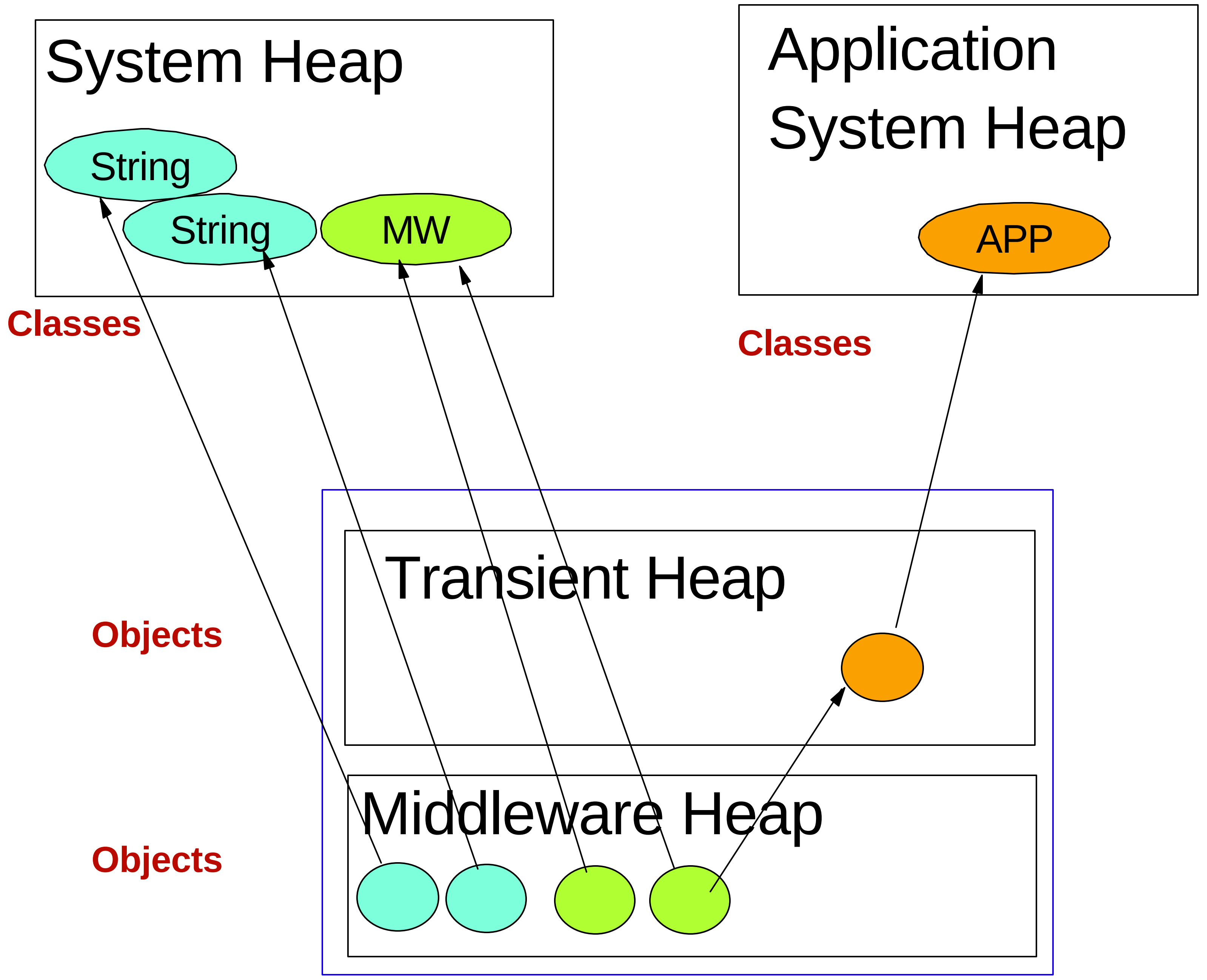
JVM Lifecycle

Middleware initialization



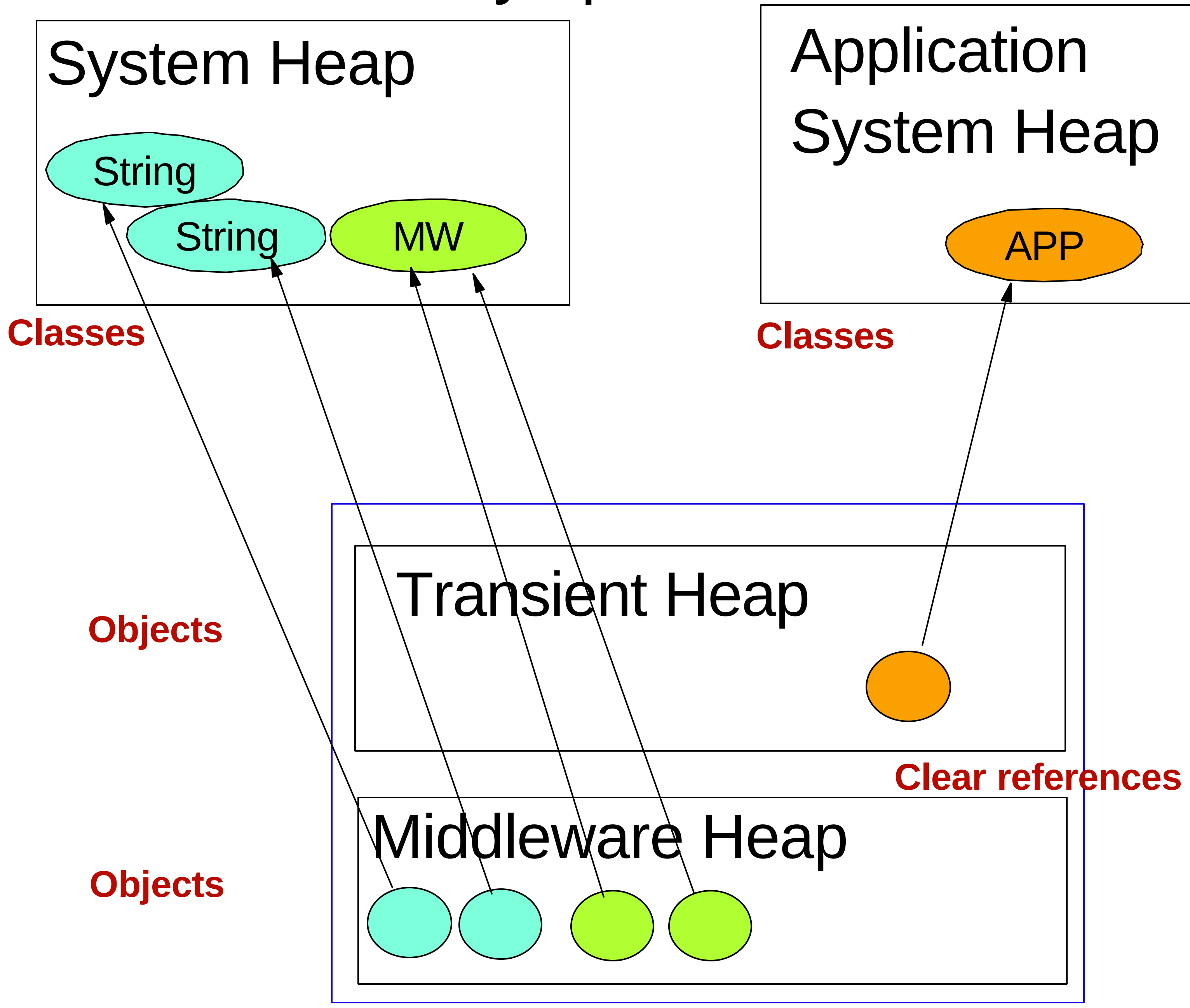
JVM Lifecycle

Transaction execution



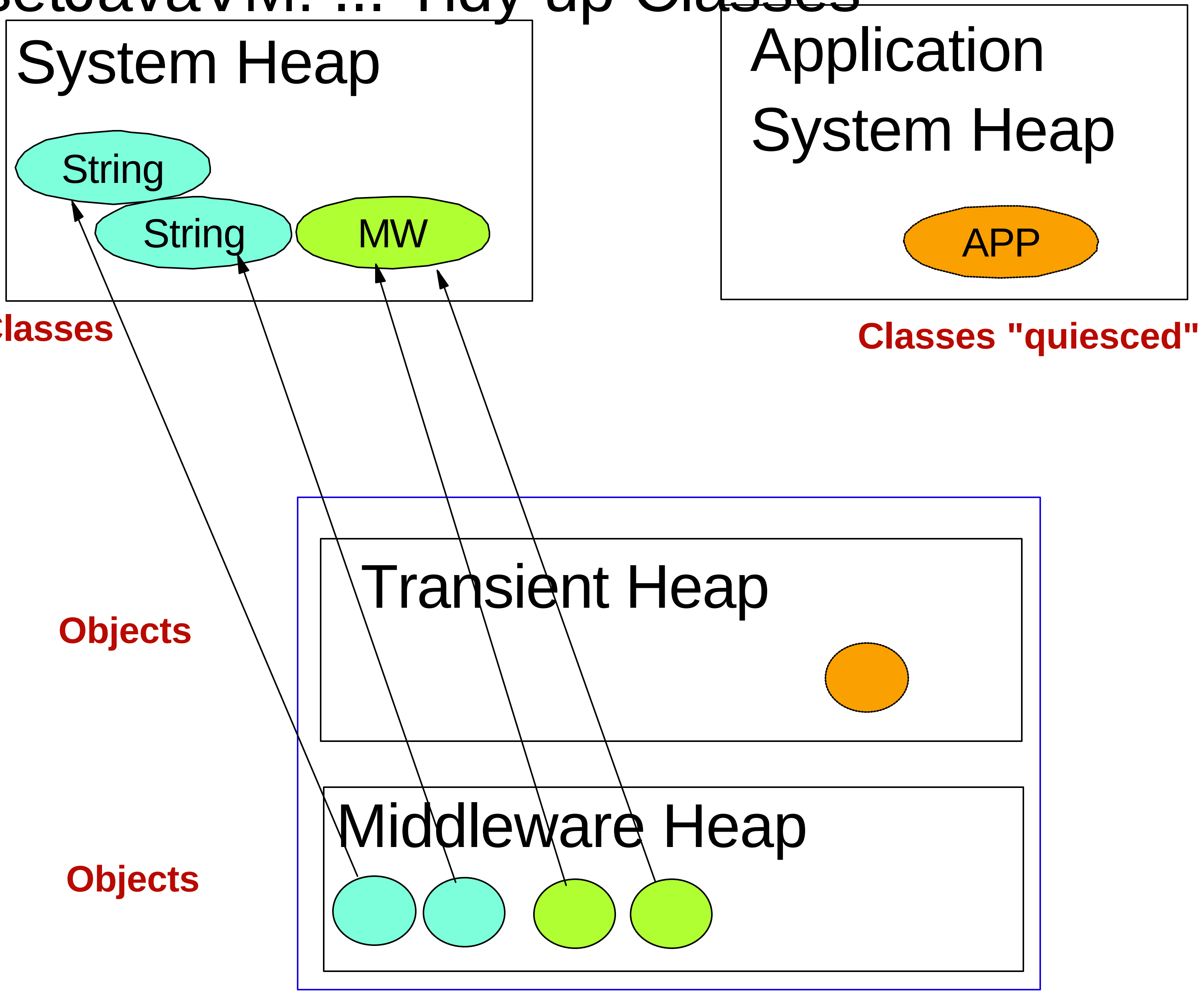
JVM Lifecycle

ResetJavaVM: Tidy up Middleware



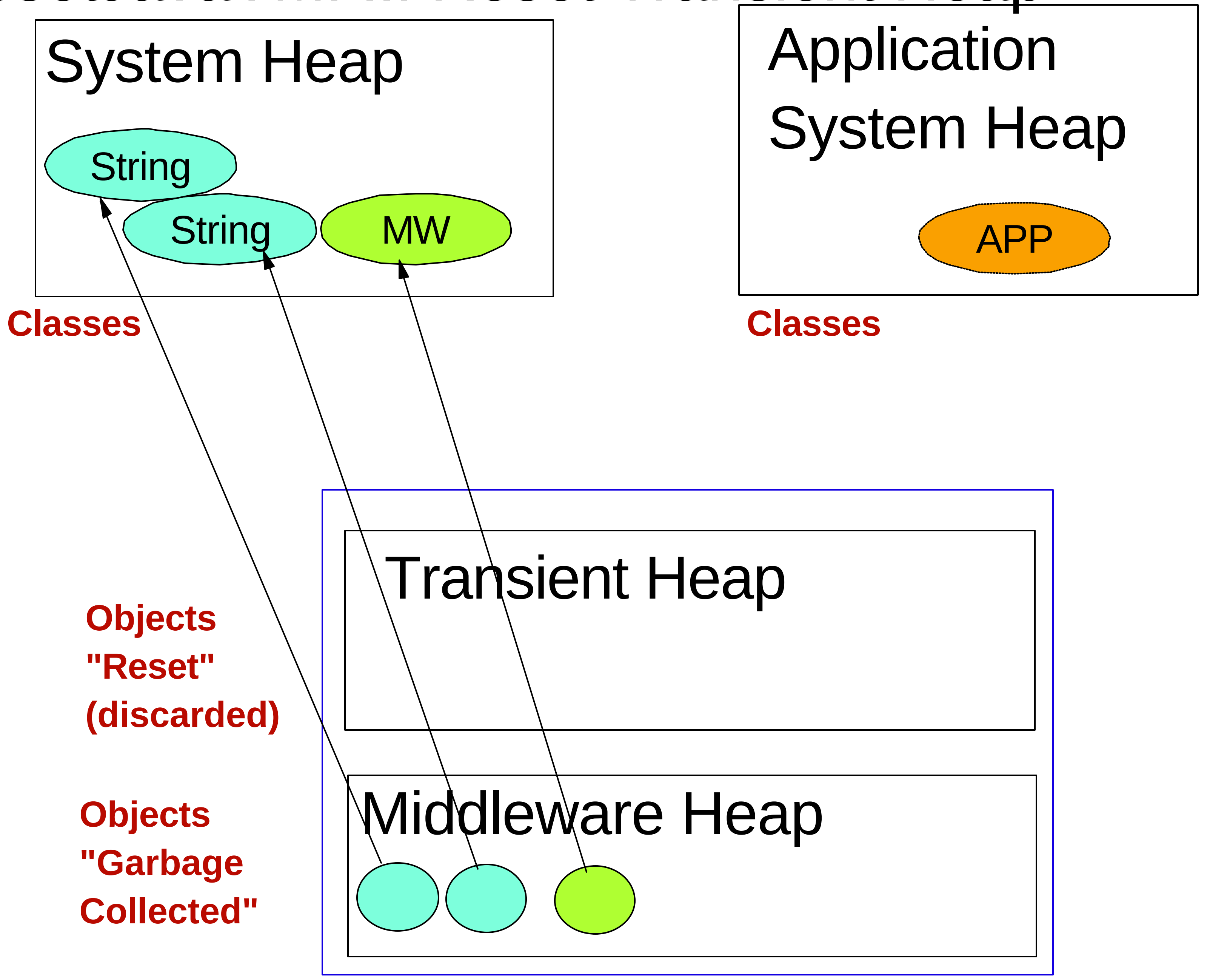
JVM Lifecycle

ResetJavaVM: ... Tidy up Classes

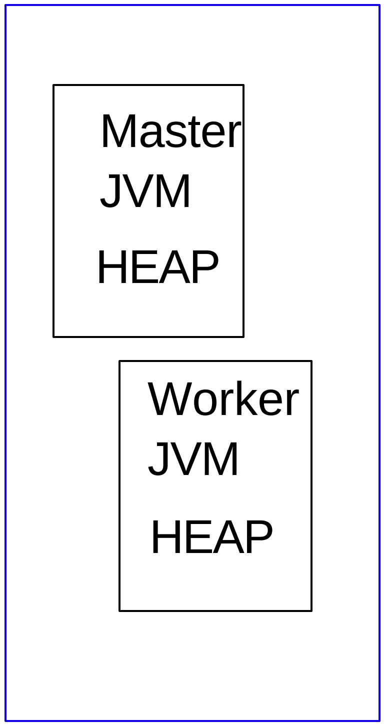


JVM Lifecycle

ResetJavaVM: ... Reset Transient Heap



Persistent Reusable Java Virtual Machine Structure

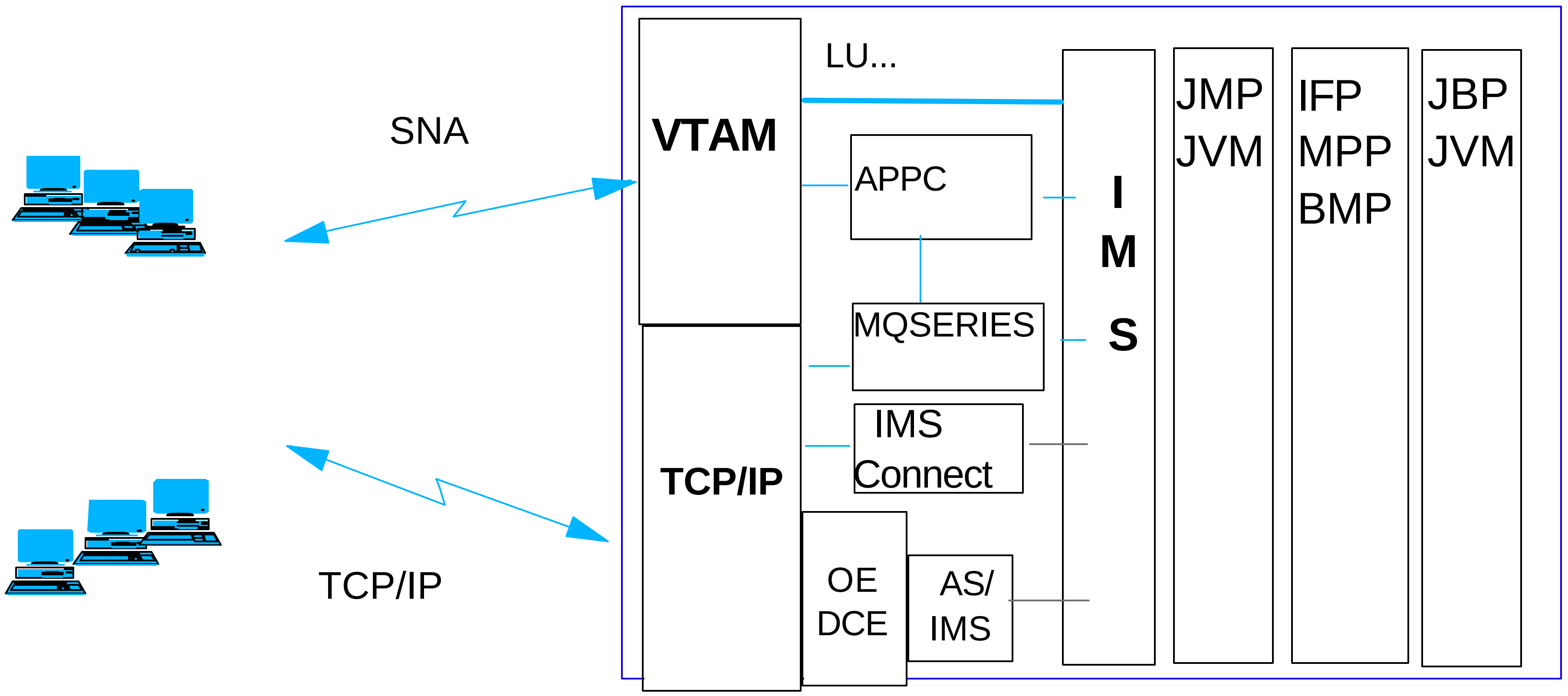


Middleware path
-Dibm.jvm.trusted.middleware.class.path= path

Shareable Application path
-Dibm.jvm.shareable.application.class.path= path

Non-Shareable Application path
-Djava.class.path= path

Access to IMS JVM Regions Java Application Programs



New IMS JVM Dependent Regions

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

IMS Java Dependent Regions

JMP region type (Java Message Processing region)

- For message-driven Java applications
- New IMSJMP JOB that EXECs the new DFSJMP procedure
- DFSJMP procedure added to IMS.PROCLIB

JBP region type (Java Batch Processing region)

- For non-message driven Java applications
- New IMSJBP JOB that EXECs the new DFSJBP procedure
- DFSJBP procedure added to IMS.PROCLIB
- Supported in DBCTL

DB2

- RRSAF interface used
  - RRS required

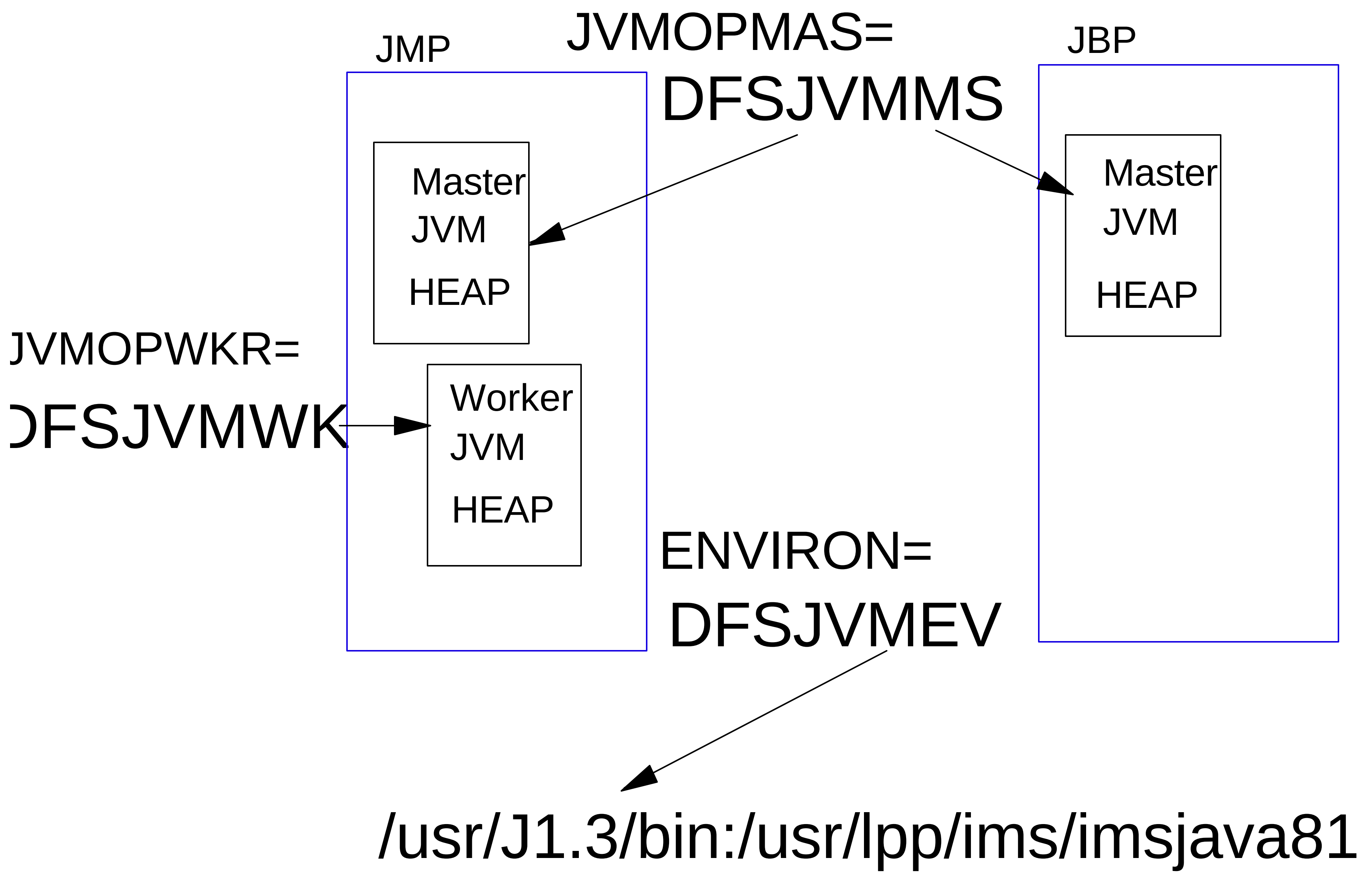
DFSJMP Procedure

- 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 for the master JVM
    - Required. If not present, region abends with ABENDU0101
  - JVMOPWKR=
    - Specifies name of IMS.PROCLIB member that contains the JVM options for the worker JVM
    - Optional
  - 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

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
  - ENVIRON=
    - Specifies name of IMS.PROCLIB member that contains the LIBPATH= environment variable specification
    - Required. If not present, region abends with ABENDU0101
  - MBR= parm
    - 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


JVMOPMAS=
DFSJVMMS
JMP
JBP
Master
JVM
HEAP
Master
JVM
HEAP
JVMOPWKR=
DFSJVMWK
Worker
JVM
HEAP
ENVIRON=
DFSJVMEV
/usr/J1.3/bin:/usr/lpp/ims/imsjava81

IMS JVMs
Master JVM

DFSJVMMS is a supplied JVMOPMAS= member in the IMS sample library

```
*************************************************************************************
*  The following two JVM options are required.
*************************************************************************************
-Dibm.jvm.shareable.application.class.path=/ims/java/applications
-Dibm.jvm.trusted.middleware.class.path=   >
 /usr/lpp/ims/imsjava71/imsjava.jar
*************************************************************************************
* -Xmaxf and -Xminf set max and min percent middlware heap free space
* -Xoss specifies Java Stack size use default 400K
* -Xms use default 1M
* -Xinitth use default of -Xms/2(1M/2=512K)
*************************************************************************************
-Xinitacsh128k
-Xinitsh128k
-Xmaxf0.6
-Xminf0.3
-Xmx64M
-Xoss400k
```

IMS JVMs
Worker JVM

DFSJVMWK is a supplied JVMOPWKR= member in the IMS sample library

```
*********************************************************************************************************
* The following JVM options are subset of the options allowed under
* JDK 1.3.1S
*********************************************************************************************************
-Xmaxf0.6
-Xminf0.3
-Xmx64M
-Xoss400k
```

IMS JVMs

DFSJVMEV is a supplied ENVIRON= member in the IMS sample library and contains path for JVM and IMS Java Classes:

```
**************************************************************************
* LIBPATH environment variable
* ----------------------------
* /usr/J1.3/bin/classic is path to libjvm.so
* /usr/J1.3/bin is path to libatoe.so
* /usr/lpp/ims/imsjava71 is path to libJavTDLI.so
**************************************************************************
LIBPATH=/usr/J1.3/bin/classic >
:/usr/J1.3/bin:/usr/lpp/ims/imsjava71
LEOPT=Y
```

IMS JVMs

```
REGION=0M

//* HFS path for Java stdin System.in.read()
//JAVAIN DD PATH=/tmp/javain,DISP=SHR

//* HFS path for Java stdout System.out.print()
//JAVAOUT DD PATH=/tmp/javaout,DISP=SHR

//* HFS path for Java stderr System.err.print()
//JAVAERR DD PATH=/tmp/javaerr,DISP=SHR
```

IMS JVMs

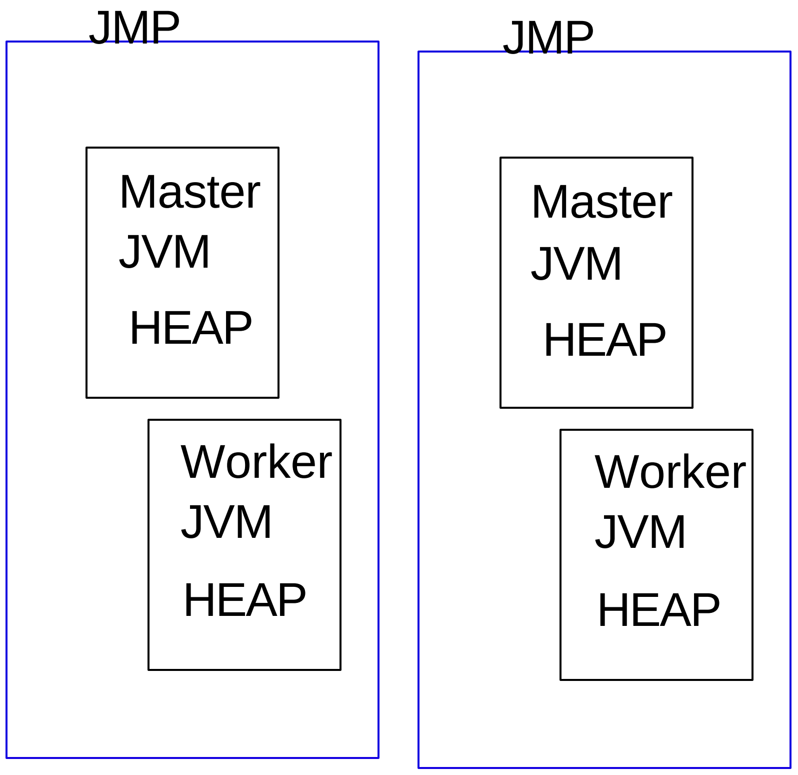
IMS.PROCLIB(DFSJVMAP)

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

/DIS ACTIVE command:

```
REGID JOBNAME  TYPE  TRAN/STEP PROGRAM  STATUS   CLASS
  1   JMP810     JMP                      WAITING  1, 2, 3, 4
      MSGRGN     TP    NONE
  2   JBP810     JBP   JBP        JBPPSB
      BATCHREG   BMP   NONE
      FPRGN      FP    NONE
      DBTRGN     DBT   NONE
      DBRMCTAB  DBRC
 *00200/132805*
```

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 - Messages

Message DFS551I

- For JBP:
  - DFS551I JBP REGION <jobname> STARTED ID=...
- For JMP:
  - DFS551I JMP REGION <jobname> STARTED ID=...

Message DFS552I

- For JBP:
  - DFS552I JBP REGION <jobname> STOPPED ID=...
- For JMP:
  - DFS552I JMP REGION <jobname> STOPPED ID=...

Message DFS554A

- For JBP:
  - DFS554A JBP810 00001 JBP   JBPPSB(6)   0C1,0000 PSB ...
- For JMP:
  - DFS554A JMP810 00001 JMP   JMPPSB(5) JMPTRAN 0C1,0000...

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 and that invokes Java routines.

Java Dependent Regions - COBOL/JAVA

For example, To make the implementation of a COBOL class available to an IMS Java program, do the following steps:

1. Compile the COBOL class with the Enterprise COBOL compiler to generate a Java source file (.java) and an object module (.o)

2. Compile the Java source file with the Java compiler to create a class file (.class)

3. Link the object code into a dynamic link library (DLL) in the HFS (.so).
   The HFS directory that contains the COBOL DLLs must be listed in the LIBPATH

```
ENVIRON= DFSJVMEV
    LIBPATH=/usr/J1.3/bin/classic >
    :/usr/J1.3/bin:/usr/lpp/ims/imsjava71:/usr/lpp/ims/cobol
```

4. Update the sharable application class path in the master JVM options member

```
JVMOPMAS=DFSJVMMS
-Dibm.jvm.shareable.application.class.path=/ims/java/applications >
/ims/cobol/applications
```

Java Dependent Regions - COBOL/JAVA

Wrapping procedure-oriented COBOL programs

A wrapper provides an interface to procedure-oriented code

To wrap COBOL code, do these steps:

1. Create COBOL class that contains a FACTORY paragraph

2. Code a factory method that uses a CALL statement to call the procedural program
   A Java program invokes the factory method to access the procedural program

Java Dependent Regions - Summary

Dependent on JDK 1.3.1S

- JDK 1.3 + Persistent Reusable JVM = JDK 1.3.1S
  - Resettable JVMs
  - Master/Worker JVMs

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