Developing IMS Java Transactions: A Practical Approach

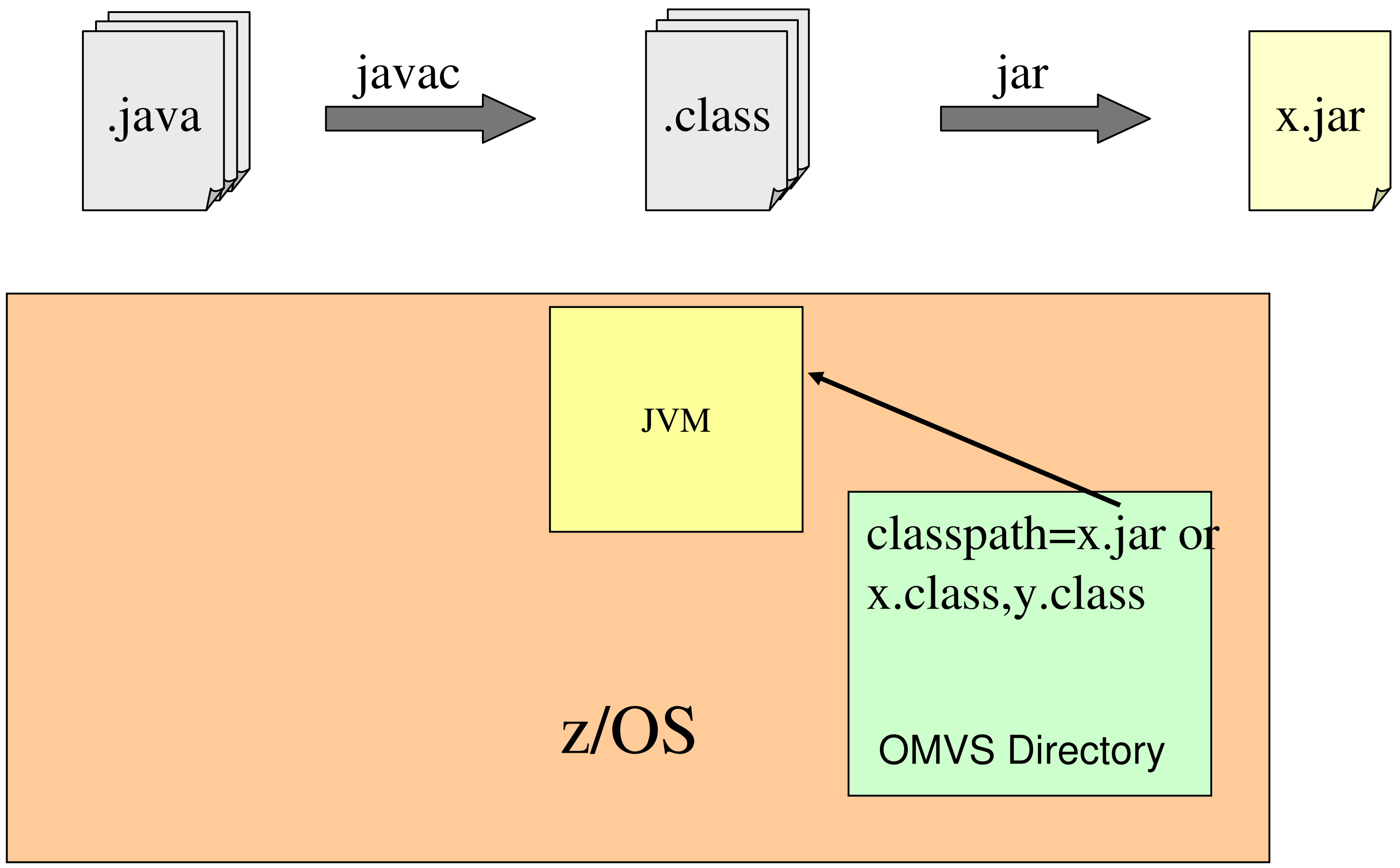
Kenny Blackman
kblackm@us.ibm.com

Information Management software

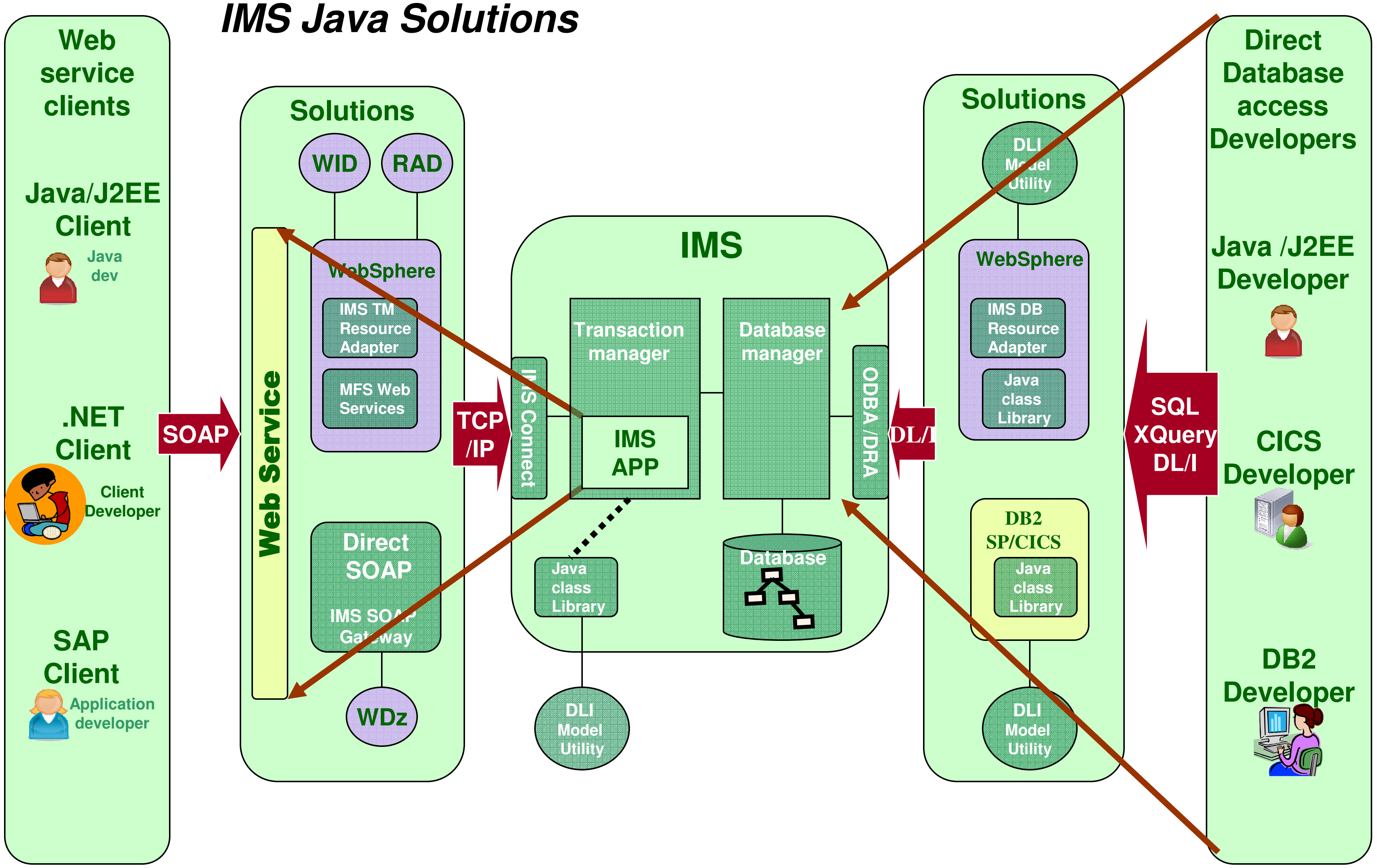
Agenda

- Java development solutions for IMS
  - Background Information
- IMS Java Class Library
  - What Is it?
  - Class Library Architecture
  - Java API for Dependent Region Processing
  - Java API for Database Access
    - DLIModel Utility
    - IMS 9 , 10 and 11 DB Access
  - Sample Application

Java Review


.java
javac
.class
jar
x.jar
JVM
classpath=x.jar or
x.class,y.class
OMVS Directory
z/OS

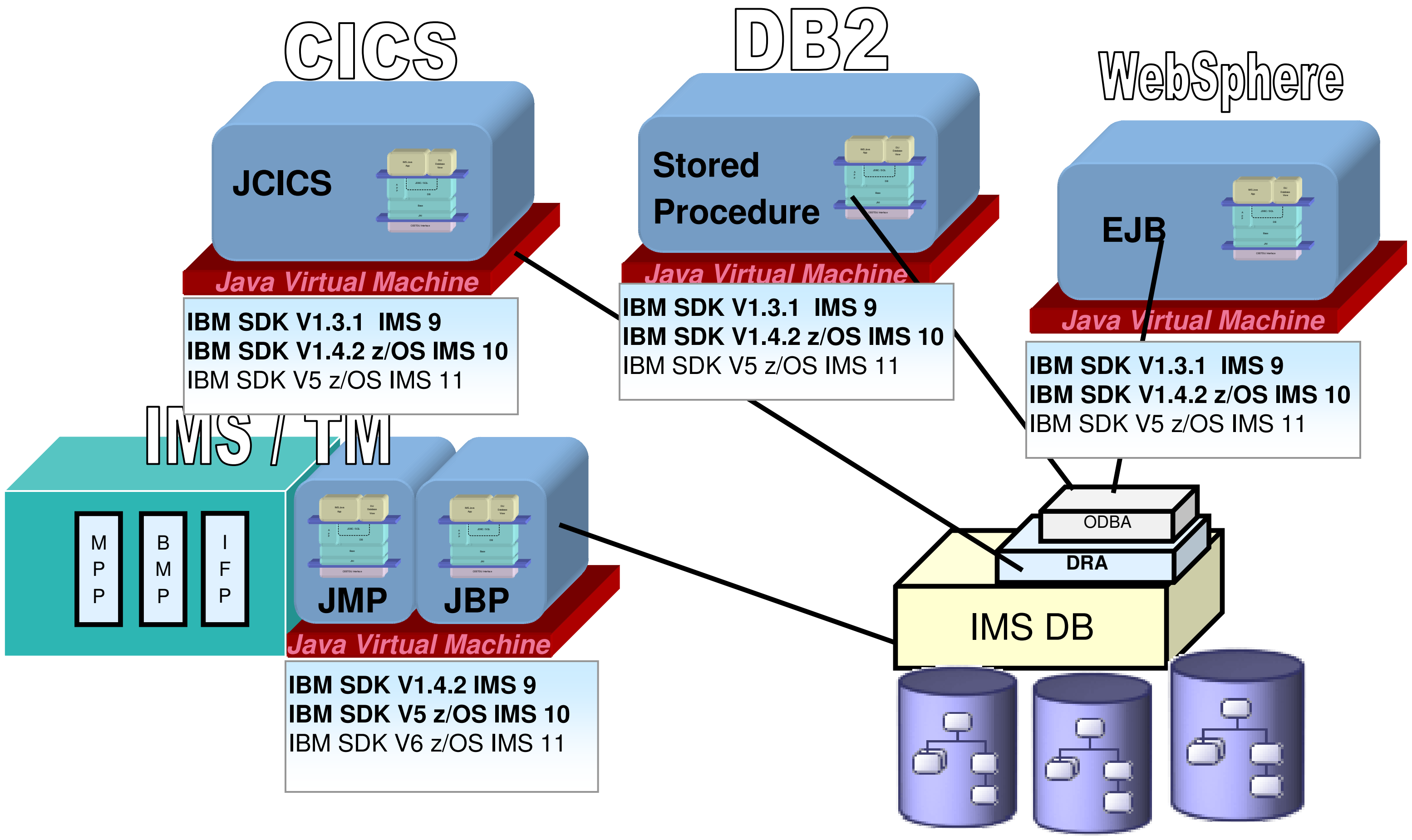
IMS Java Solutions



IMS Java Drivers

- IMS 11 Open Database APIs JDBC 3.0
  - Universal DB resource adapter
  - Universal JDBC driver
    - type-4 and type-2 connections
  - Universal DL/I driver

- IMS 9,10 Java Drivers JDBC 2.1
  - Java dependent region resource adapter
  - DB resource adapter
  - Distributed DB resource adapter



JDBC Explained

- Defines a standard Java API for accessing relational databases

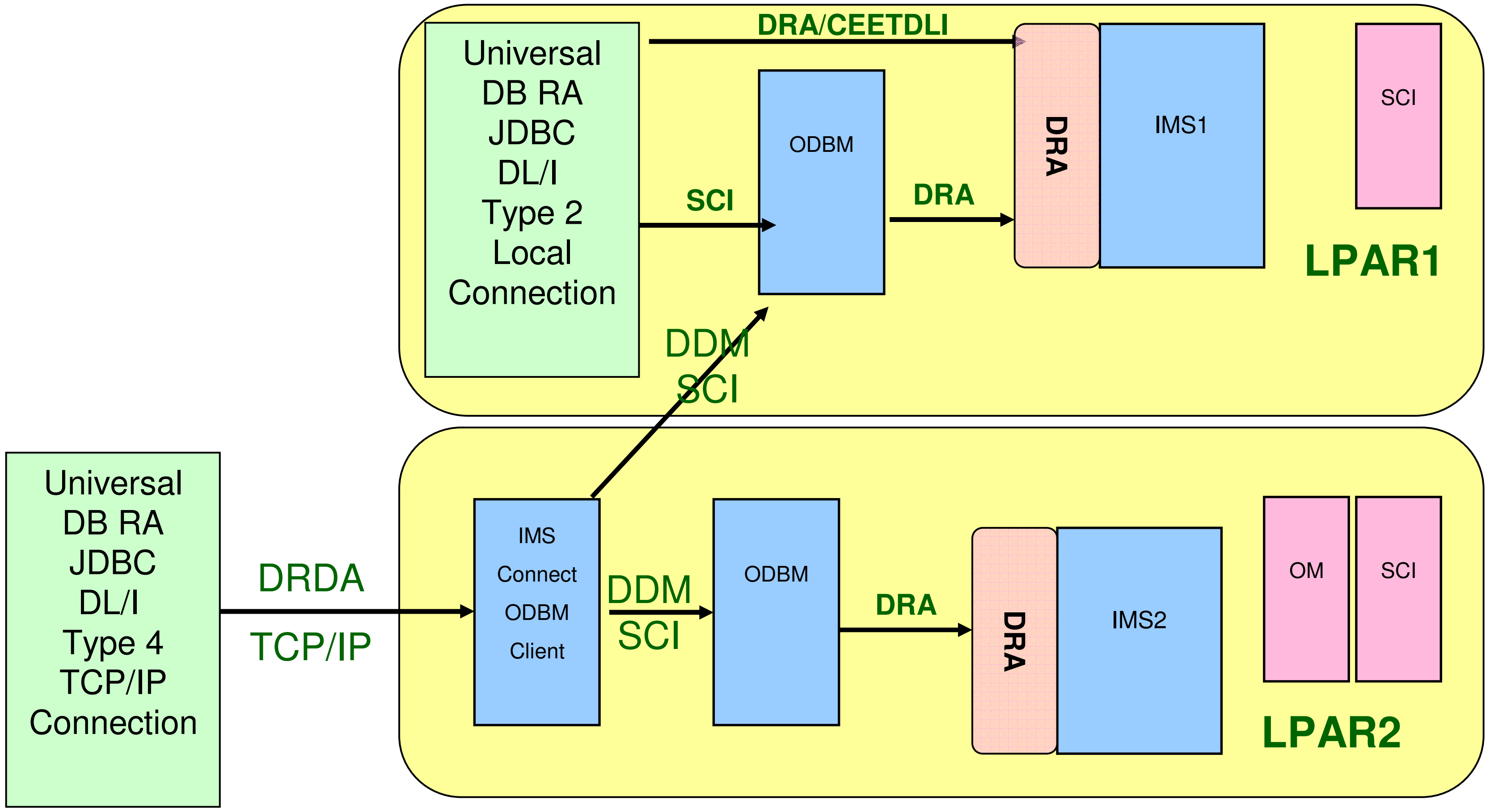
- Provides an API for sending SQL statements to a database and processing the tabular data returned

- Executing JDBC query statements
  - Establish and open connection to database
  - Execute query and obtain results
  - Process results
  - Close connection

JDBC API for IMS DB

- Needed IMS database access from Java
  - Java library provides a database access API
    - Connects to an IMS database
      - Allocates the PSB (APSB) if not under IMS control
    - Provides ability to process all IMS DB access commands
      - GHU, GU, GHN, GN, GNP, ISRT, REPL, DLET
    - Provides Java Objects representative of SSA lists and Segment Search Arguments
      - Full support of SSA functionality
        - All command codes are supported

IMS Universal Drivers Type 2 and 4 Connections

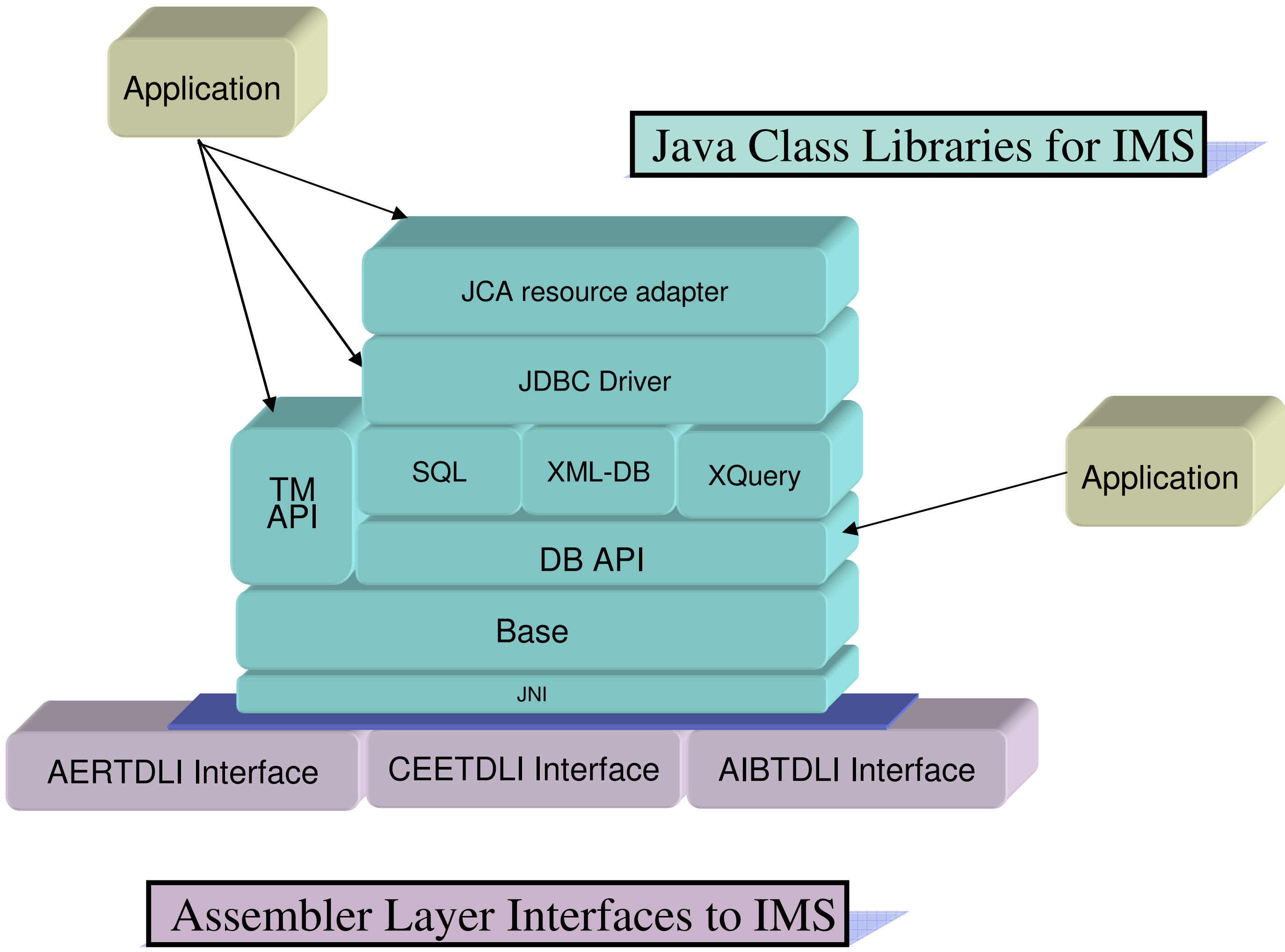


IMS JDBC Drivers

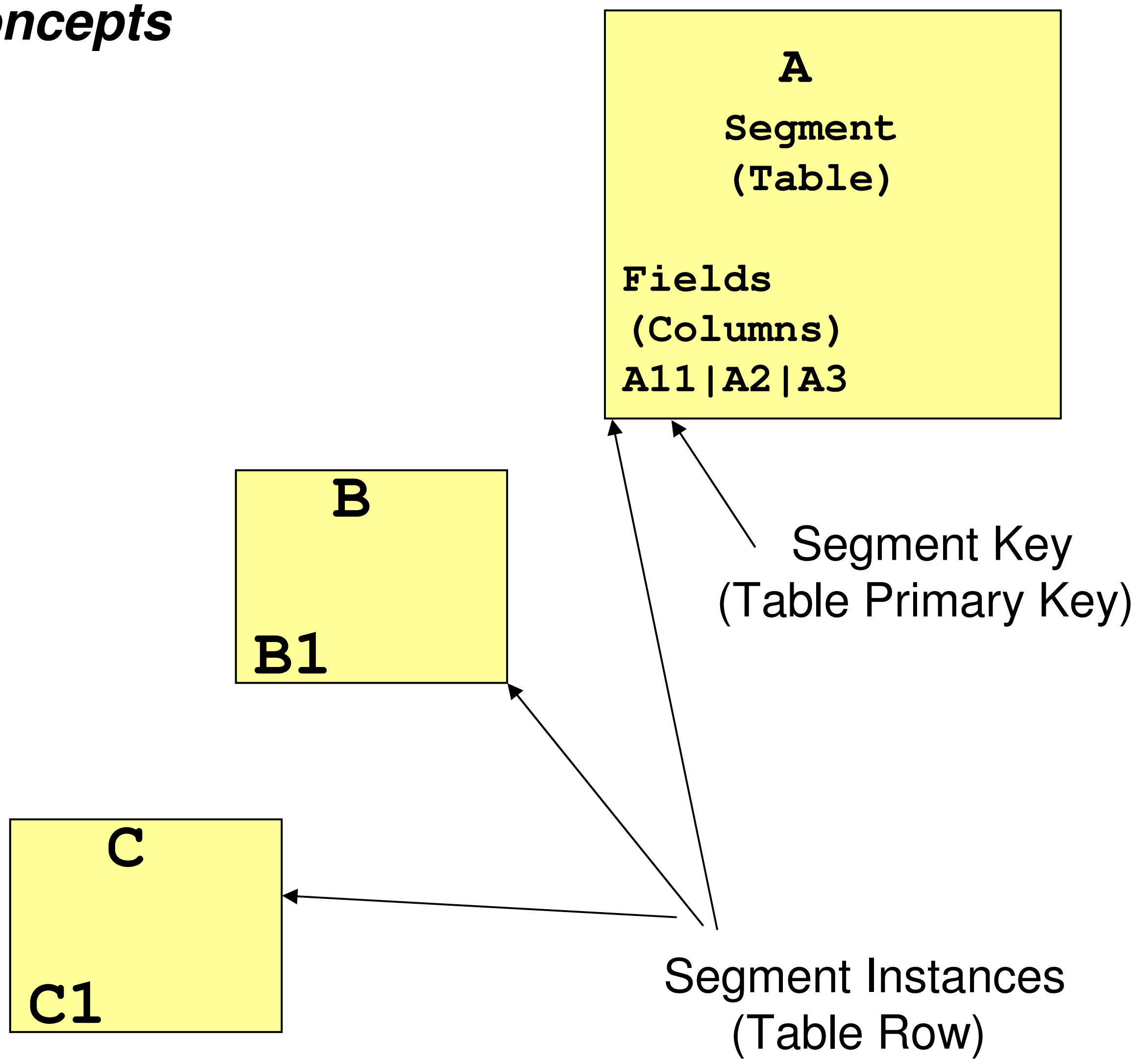
- Comparison of programming approaches for accessing IMS:

Use the J2EE JCA 1.5-compliant IMS 11 IMS Universal DB resource adapter type 4	Access IMS data through TCP/IP from a J2EE application that resides on a distributed platform or a z/OS platform that is on a different LPAR from the IMS subsystem.
Use the IMS 11 Universal JDBC driver type 4 or the Universal DL/I Driver type 4	Accessing IMS data through TCP/IP from a Java application (non-J2EE) that resides on a distributed platform or a z/OS platform that is on a different LPAR from the IMS subsystem.
Use the J2EE JCA 1.5-compliant IMS 11 IMS Universal DB resource adapter type 2	Access IMS data from a J2EE application that resides on a z/OS platform that is on a same LPAR as the IMS subsystem.
Use the IMS 11 Universal JDBC driver type 2 or the Universal DL/I Driver type 2	Accessing IMS data from a Java application (non-J2EE) that resides on a z/OS platform that is on a same LPAR as the IMS subsystem.

Java class libraries for IMS



SQL and IMS Concepts



Application Development Steps

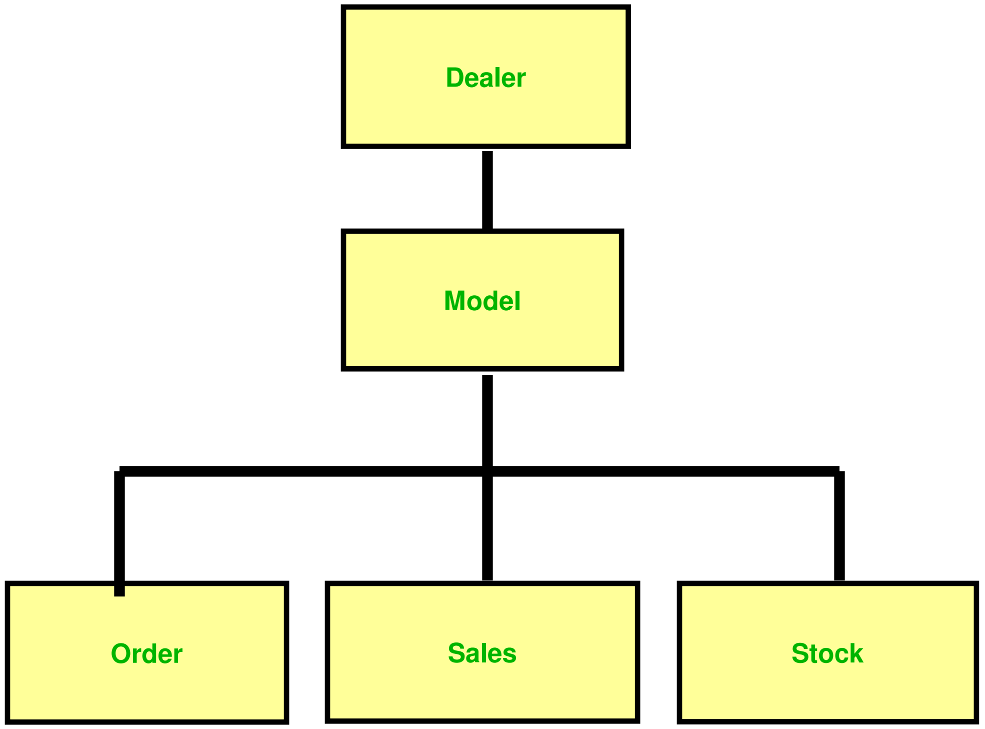
- DBD , PSB , Fields to create IMS Metadata
- Run DL/I Model Utility
  - DLIDatabaseView Metadata
  - IMS Java Report
- Write Application
- Compile
- Execute
- Debug

Building an IMS Java Application

- Define input and output messages
  - ►Subclass IMSFieldMessage
  - ►Repeating fields
- Define database layout
  - ►Subclass DLIDatabaseView
- Define database segments
  - ►Subclass DLISegment
- Write application program
  - ►Subclass IMSApplication

$\left.\begin{array}{l} \\ \\ \\ \\ \\ \\ \end{array}\right\}$ defines metadata required for JDBC

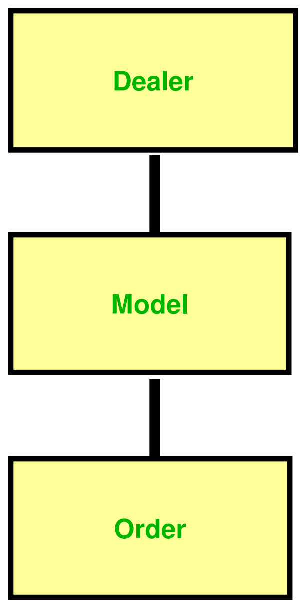
DBD NAME=DEALERDB



DBD NAME=DEALERDB

```
SEGM NAME=DEALER,PARENT=0,BYTES=94,
     FIELD NAME=(DLRNO,SEQ,U),BYTES=4,START=1,TYPE=C

SEGM NAME=MODEL,PARENT=DEALER,BYTES=43
     FIELD NAME=(MODTYPE,SEQ,U),BYTES=2,START=1,TYPE=C

SEGM NAME=ORDER,PARENT=MODEL,BYTES=127
     FIELD NAME=(ORDNBR,SEQ,U),BYTES=6,START=1,TYPE=C

SEGM NAME=SALES,PARENT=MODEL,BYTES=113
     FIELD NAME=(SALDATE,SEQ,U),BYTES=8,START=1,TYPE=C

SEGM NAME=STOCK,PARENT=MODEL,BYTES=62
     FIELD NAME=(STKVIN,SEQ,U),BYTES=20,START=1,TYPE=C
```

PSB NAME=MYDLRPSB
PCB NAME=MYDLRPCB
LANG=JAVA

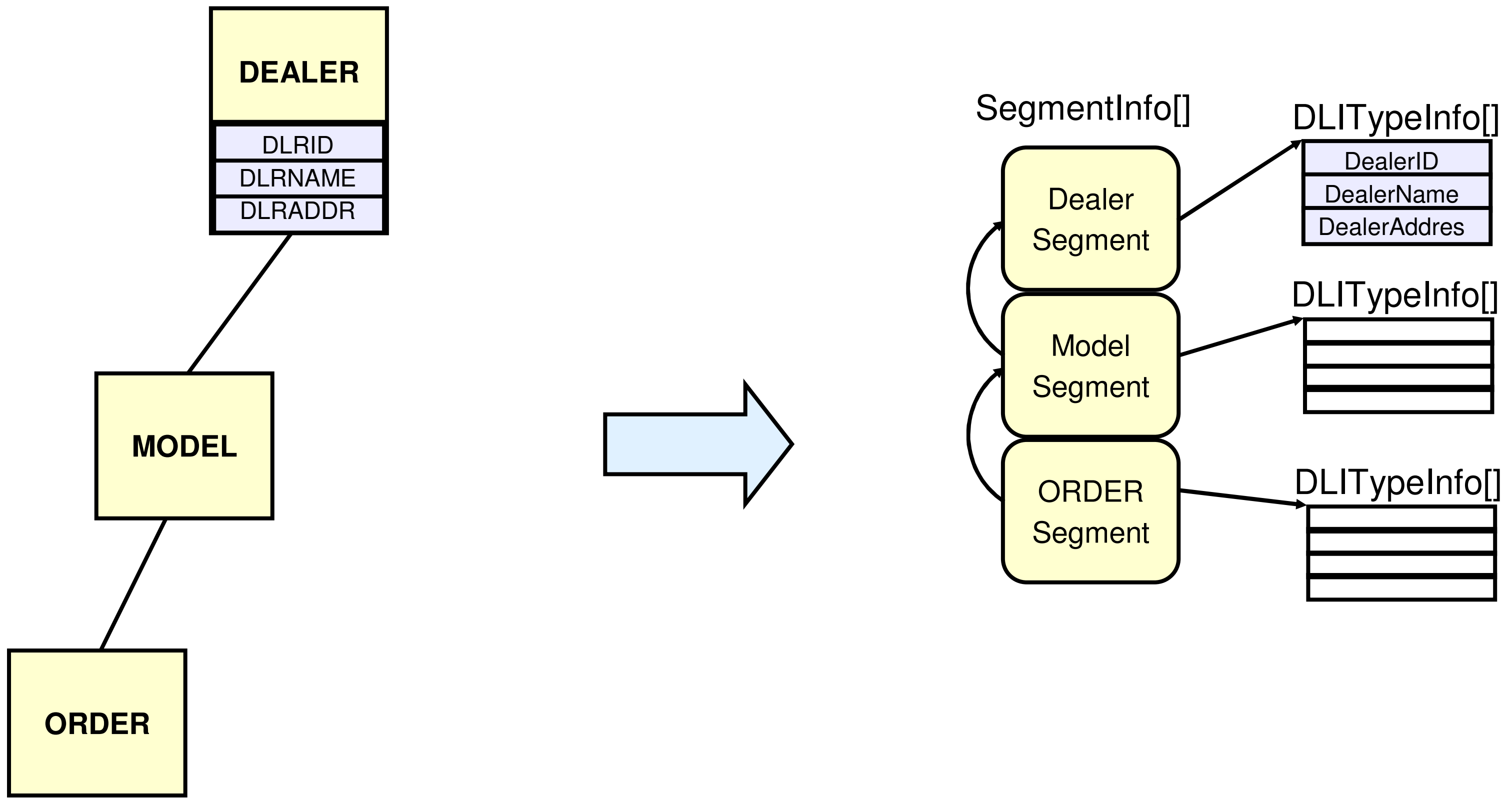


COBOL, SQL, and IMS Java Data Types

Copybook Format	IMS Java Type (SQL Type)	Java Type
PIC X	CHAR	java.lang.String
PIC 9 BINARY	(see next table)	(see next table)
COMP-1	FLOAT	float
COMP-2	DOUBLE	double
PIC 9 COMP-3	PACKEDDECIMAL	java.math.BigDecimal
PIC 9 DISPLAY	ZONEDDECIMAL	java.math.BigDecimal

Digits	Storage Size	IMS Java Type (SQL Type)	Java Type
1 through 4	2 bytes	SMALLINT	short
5 through 9	4 bytes	INTEGER	int
10 through 18	8 bytes	BIGINT	long

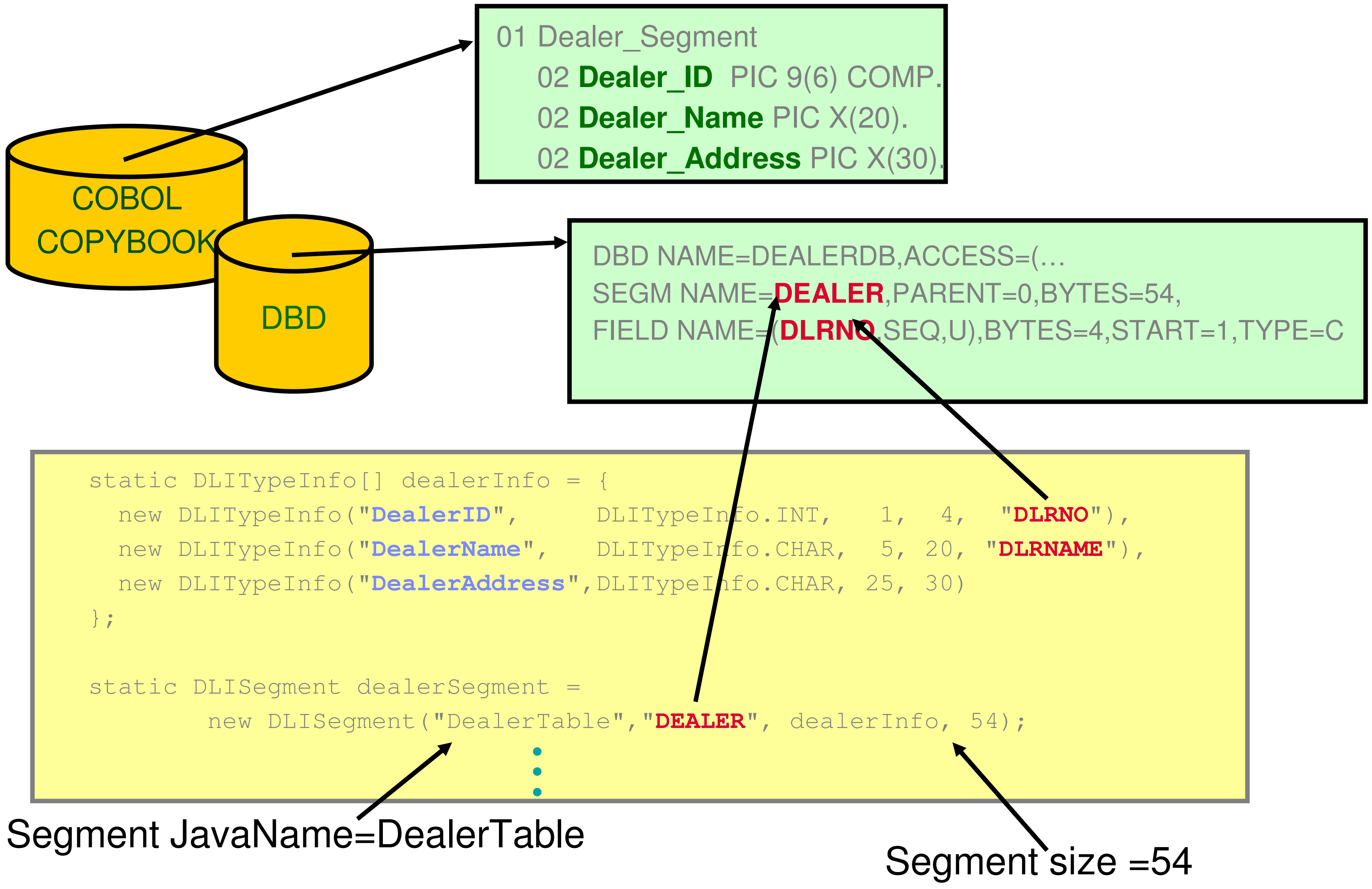
IMS Metadata



DBDLIB, PSBLIB

DLIDatabaseView

Define Database Segments (Tables) and Fields (Columns)

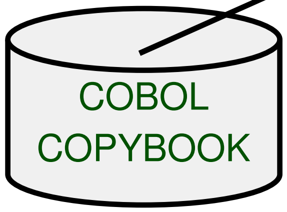


```
01 Dealer_Segment
    02 Dealer_ID  PIC 9(6) COMP.
    02 Dealer_Name PIC X(20).
    02 Dealer_Address PIC X(30).
```

```
DBD NAME=DEALERDB,ACCESS=(…
SEGM NAME=DEALER,PARENT=0,BYTES=54,
FIELD NAME=(DLRNO,SEQ,U),BYTES=4,START=1,TYPE=C
```

```
static DLITypeInfo[] dealerInfo = {
  new DLITypeInfo("DealerID",      DLITypeInfo.INT,    1,  4,  "DLRNO"),
  new DLITypeInfo("DealerName",    DLITypeInfo.CHAR,   5, 20, "DLRNAME"),
  new DLITypeInfo("DealerAddress",DLITypeInfo.CHAR, 25, 30)
};

static DLISegment dealerSegment =
      new DLISegment("DealerTable","DEALER", dealerInfo, 54);
                          :
```

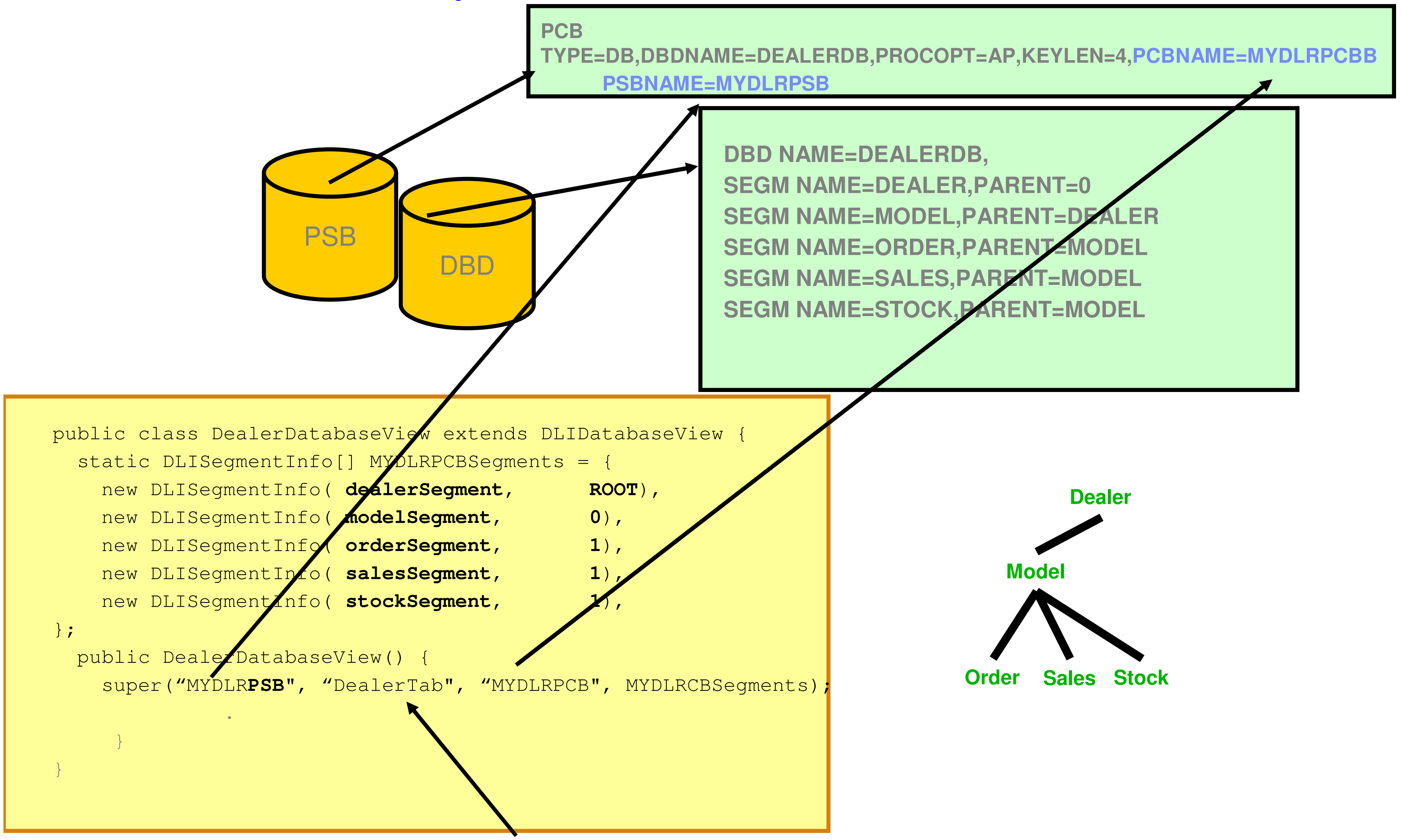
Redefining Fields



```
01 Dealer_Segment
    02 Dealer_ID  PIC X(6) COMP.
    02 Dealer_Name PIC X(20).
    02 Dealer_Address PIC X(30)
        05 Dealer_Street PIC X(14).
        05 Dealer_City PIC X(14).
        05 Dealer_State PIC X(2).
```

```
static DLITypeInfo[] dealerInfo = {
  new DLITypeInfo("DealerNo",      DLITypeInfo.INT,    1,   4, "DLRNO"),
  new DLITypeInfo("DealerName",    DLITypeInfo.CHAR,   5,  20, "DLRNAME"),
  new DLITypeInfo("DealerAddress", DLITypeInfo.CHAR, 25,  30),
    new DLITypeInfo("Street",        DLITypeInfo.CHAR, 25,  14),
    new DLITypeInfo("City",          DLITypeInfo.CHAR, 39,  14),
    new DLITypeInfo("State",         DLITypeInfo.CHAR, 53,   2)
};

static DLISegment dealerSegment =
        new DLISegment("DealerTable","DEALER", dealerInfo, 54);
                        •
                        •
                        •
```

Define Database Layout



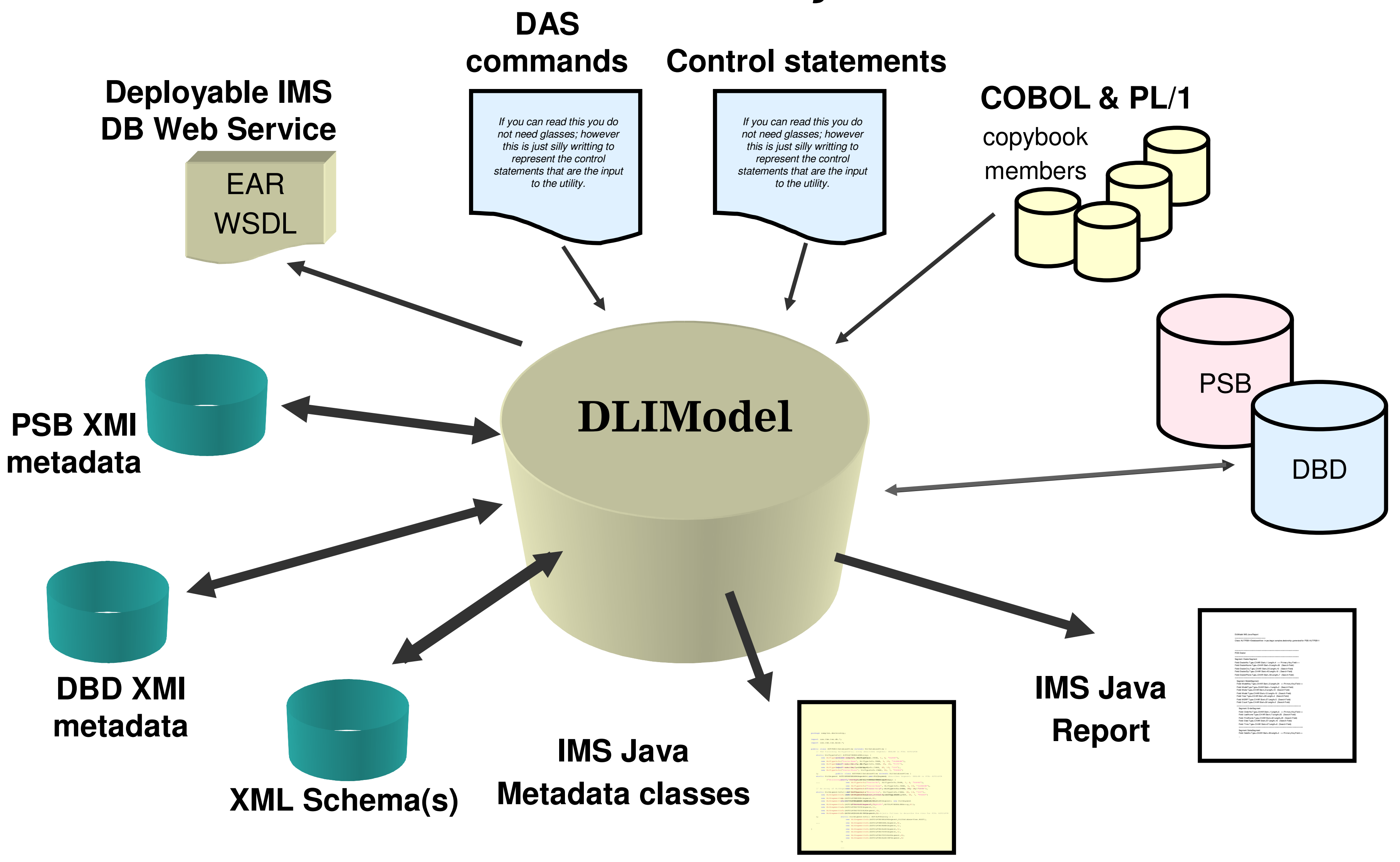
PCB
TYPE=DB,DBDNAME=DEALERDB,PROCOPT=AP,KEYLEN=4,PCBNAME=MYDLRPCBB
PSBNAME=MYDLRPSB

DBD NAME=DEALERDB,
SEGM NAME=DEALER,PARENT=0
SEGM NAME=MODEL,PARENT=DEALER
SEGM NAME=ORDER,PARENT=MODEL
SEGM NAME=SALES,PARENT=MODEL
SEGM NAME=STOCK,PARENT=MODEL

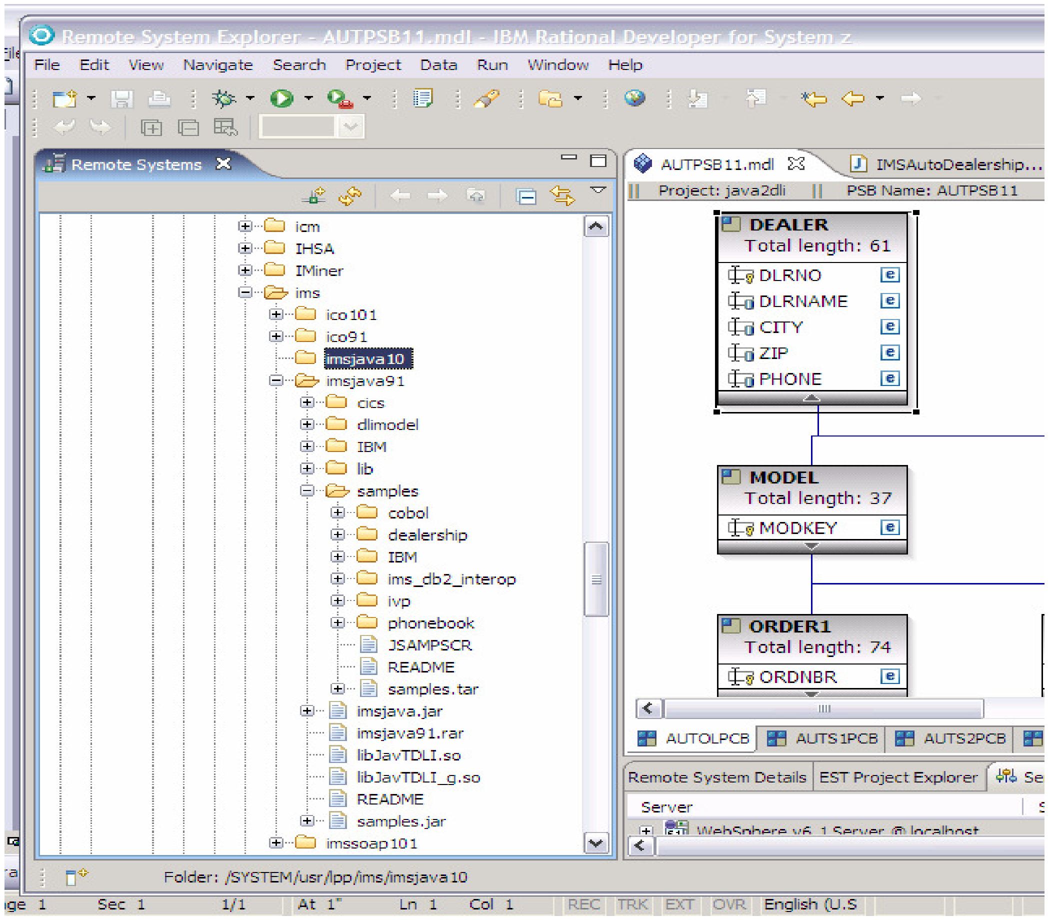
```
public class DealerDatabaseView extends DLIDatabaseView {
  static DLISegmentInfo[] MYDLRPCBSegments = {
    new DLISegmentInfo( dealerSegment,       ROOT),
    new DLISegmentInfo( modelSegment,        0),
    new DLISegmentInfo( orderSegment,        1),
    new DLISegmentInfo( salesSegment,        1),
    new DLISegmentInfo( stockSegment,        1),
};
  public DealerDatabaseView() {
    super(“MYDLRPSB", “DealerTab", “MYDLRPCB", MYDLRCBSegments);
                  .
      }
}
```

PCB JavaName=DealerTab

DLIModel Utility


DAS commands
Control statements
Deployable IMS DB Web Service
EAR
WSDL
If you can read this you do not need glasses; however this is just silly writting to represent the control statements that are the input to the utility.
If you can read this you do not need glasses; however this is just silly writting to represent the control statements that are the input to the utility.
COBOL & PL/1
copybook members
DLIModel
PSB XMI metadata
PSB
DBD
DBD XMI metadata
XML Schema(s)
IMS Java Metadata classes
IMS Java Report

DLIModel GUI Plug-in



Project: java2dli || PSB Name: AUTPSB11

DEALER
Total length: 61
DLRNO
DLRNAME
CITY
ZIP
PHONE

MODEL
Total length: 37
MODKEY

ORDER1
Total length: 74
ORDNBR

Folder: /SYSTEM/usr/lpp/ims/imsjava10

ge 1 Sec 1 1/1 At 1" Ln 1 Col 1

DLIModel Java Report (programming guide)

```
DLIModel IMS Java Report
========================
Class: DealerDatabaseView  in package: ims.dealership  generated for PSB: MYDLRPSB

==================================================
PCB: DealerTab
==================================================
Segment: DealerTable
Field: DealerNo Type=INTEGER Start=1 Length=4   ++ Primary Key Field ++
Field: DealerName Type=CHAR Start=5 Length=30   (Search Field)
Field: DealerCity Type=CHAR Start=35 Length=10   (Search Field)
Field: DealerZip Type=CHAR Start=45 Length=10   (Search Field)
Field: DealerPhone Type=CHAR Start=55 Length=7   (Search Field)
==================================================
  Segment: ModelTable
  Field: ModelKey Type=CHAR Start=3 Length=24   ++ Primary Key Field ++
  Field: ModelType Type=CHAR Start=1 Length=2   (Search Field)
  Field: Make Type=CHAR Start=3 Length=10   (Search Field)
  Field: Model Type=CHAR Start=13 Length=10   (Search Field)
  Field: Year Type=DATE Qualifier=yyyy Start=23 Length=4   (Search Field)
  ==================================================
      Segment: OrderTable
      Field: OrderNo Type=ZONEDDECIMAL Qualifier=999999 Start=1 Length=6   ++ Primary Ke
      ...
      Field: Time Type=CHAR Start=67 Length=8   (Search Field)
      ==================================================
```

Establish and Open Connection

- Load the IMS Java JDBC driver
- Get IMS Java Connection from Driver Manager
  - ►URL must begin with 'jdbc:dli:' followed by fully qualified class name
  - ►Connections are made to a PSB by passing in the PSB Database Metadata (DLIDatabaseView)

```
public DealerDatabaseView() {
    super("MYDLRPSB", "DealerTab", "MYDLRPCB", MYDLRCBSegments);
);
```

```
PCB    TYPE=DB,DBDNAME=DEALERDB,PROCOPT=AP,KEYLEN=4,PCBNAME=MYDLRPCBB
       PSBNAME=MYDLRPSB
```

```
//load driver
Class.forName(com.ibm.ims.DLIDriver);

//create connection
Connection con =
DriverManager.getConnection("jdbc:dli:DealerDatabaseView");
```

Executing a Query

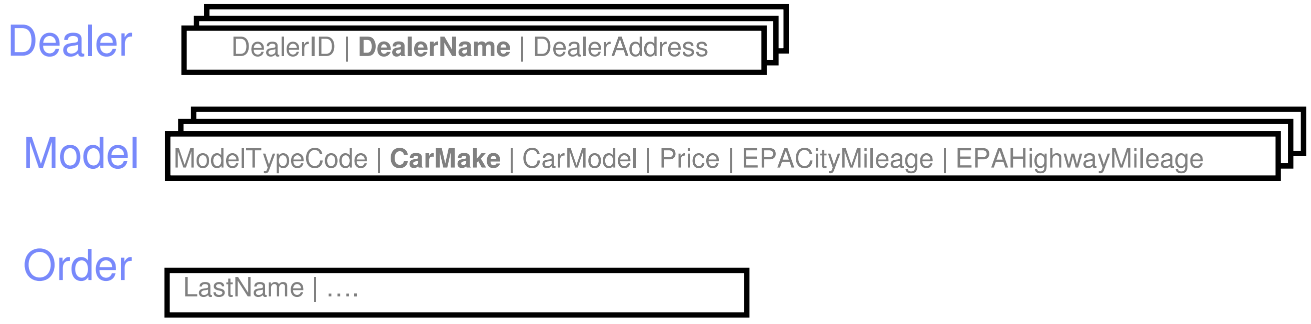
```
Statement stmt = con.createStatement();
ResultSet results = stmt.executeQuery("SELECT Dealer.Table.DealerName    OrderTable.LastName, " +
                                      "FROM DealerTab.ORDER " +
                                      "WHERE ModelTable.MSRP > '50000'" +
                                       "AND OrderTable.Date > = '03/14/2008''"
                                       "AND OrderTable.Date <= ''03/31/2008'"
                                      );
```

```
public DealerDatabaseView() {
  super("MYDLRPSB", "DealerTab", "MYDLRPCB", MYDLRCBSegments);
```

```
PCB   TYPE=DB,DBDNAME=DEALERDB,PROCOPT=AP,KEYLEN=4,PCBNAME=MYDLRPCBB
                                    PSBNAME=MYDLRPSB
```

* **make sure you PCB qualify the segment in the FROM clause**

recall...


Dealer
DealerID | DealerName | DealerAddress
Model
ModelTypeCode | CarMake | CarModel | Price | EPACityMileage | EPAHighwayMileage
Order
LastName | ….

SQL Parsing

SQL

SELECT DealerTable.DealerName, OrderTable.LastName

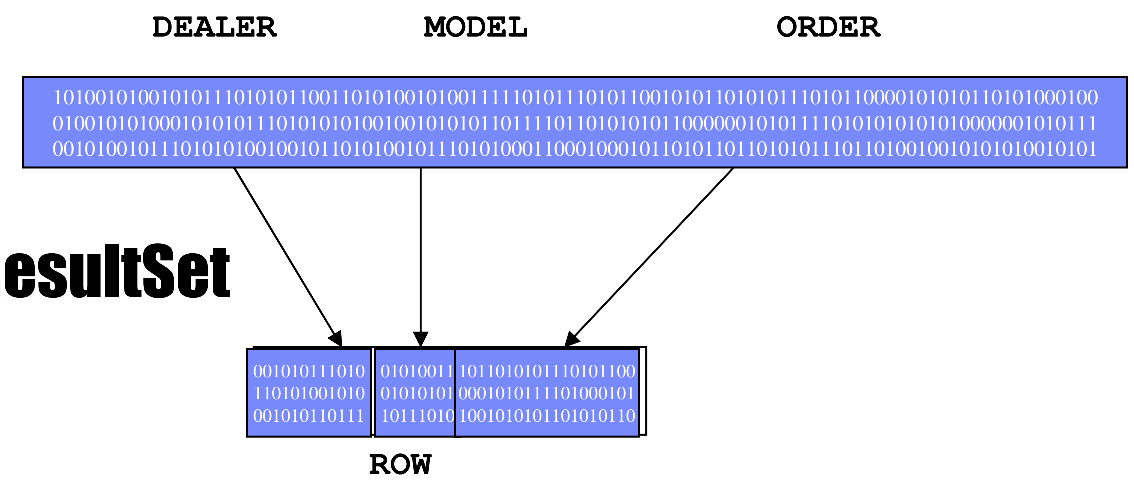
SSA List

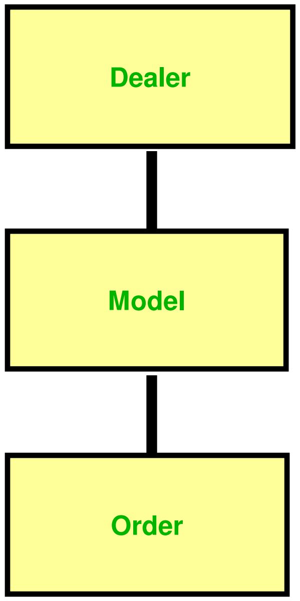
```
DEALER   *D
MODEL    (MSRP      GT50000)
ORDER    (DATE      GE20080314&
          DATE      LE20080331)
```

Note:
D command code
Sets IMS Path call

IOArea

ResultSet





IMS Java Result Set Types

- Forward-Only (default)
  - Each next() call hits the DB
  - TYPE_FORWARD_ONLY
  - Calls:
    - ResultSet.next()

- Scroll-Insensitive
  - executeQuery hits DB, and caches all results
  - TYPE_SCROLL_INSENSITIVE
  - Calls:
    - ResultSet.next()
    - ResultSet.previous()
    - ResultSet.absolute(int)
    - ResultSet.relative(int)

Processing Results

- **Iterate through ResultSet by calling next() method**
  - –Returns false when no more results
- **Call ResultSet.getXXX methods to access individual fields in results**

```
while (results.next()) {
    String DealerNme = results.getString(“DealerName");        //or results.getString(1);
    String OrderNme =  results.getString(“OrderLastName");   //or results.getString(2);

}
```

path call ioarea result 1

53SJ7 | George | 555 Bailey Ave.	FF13 |Toyota |camry|9000|24|28	11346 |06122000|00000000|blue|9000|1998.

path call ioarea result 2

53SJ7 | George | 555 Bailey Ave.	PR27|Dodge|Durango|9800|21|26	12456 |07232000|00000000|blue|9800|1999.

Datatype Conversion

	TINYINT	SMALLINT	INTEGER	BIGINT	FLOAT	DOUBLE	BIT	CHAR	VARCHAR	PACKEDDECIMAL	ZONEDDECIMAL	BINARY	DATE	TIME	TIMESTAMP
getByte	X	O	O	O	O	O	O	O	O	O	O				
getShort	O	X	O	O	O	O	O	O	O	O	O				
getInt	O	O	X	O	O	O	O	O	O	O	O				
getLong	O	O	O	X	O	O	O	O	O	O	O				
getFloat	O	O	O	O	X	O	O	O	O	O	O				
getDouble	O	O	O	O	O	X	O	O	O	O	O				
getBoolean	O	O	O	O	O	O	X	O	O	O	O				
getString	O	O	O	O	O	O	O	X	X	O	O	O	O	O	O
getBigDecimal	O	O	O	O	O	O	O	O	O	X	X				
getBytes												X			
getDate								O	O				X		O
getTime								O	O					X	O
getTimestamp								O	O				O	O	X

An 'X' indicates the getXXX method is recommended to access the given data type
An 'O' indicates the getXXX method may be legally used to access the given data type

IMS Java ResultSet Concurrency

- **Read-Only (default)**
  - CONCUR_READ_ONLY*
  - Does not allow updates using the ResultSet interface

```
Statement stmt = con.createStatement(ResultSet.TYPE_SCROLL_INSENSITIVE,
                                     ResultSet.CONCUR_READ_ONLY)
```

- **Updatable**
  - CONCUR_UPDATABLE*
  - Allows updates using the ResultSet interface

```
st = con.createStatement(ResultSet.TYPE_FORWARD_ONLY,
     ResultSet.CONCUR_UPDATABLE);
```

*Concurrency is hard-coded into the PCB and cannot be modified

Close Connection

```
    try {
        connection.close();
        IMSTransaction.getTransaction().commit();     // IMS 9 JMP only
    } catch (SQLException e) {
        System.err.println("Error while closing connection" + e.toString());
        IMSTransaction.getTransaction().rollback();
    }
}
```

SQL keywords support

- Field Renaming
  - AS

```
SELECT DealerNo AS DealerNumber
FROM DealerTable.Dealer
```

Display all the values of DealerNo in a column labeled DealerNumber.

  - Aggregates
  - AVG, COUNT, MAX, MIN, SUM, and GROUP BY

```
SELECT AVG(age), Dept AS Department
FROM MyPCB.Employees
GROUP BY Department
```

Display the average age per department.

SQL keywords support

- Ordering
  - ORDER BY, ASC, DESC

```
SELECT firstName, lastName, department
FROM MyPCB.Employees
ORDER BY lastName ASC, firstName DESC
```

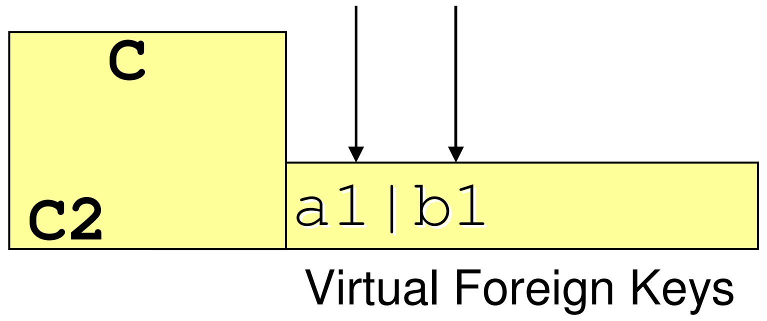
Order by lastName in ascending order, followed by firstName in descending order in the case of a tie.

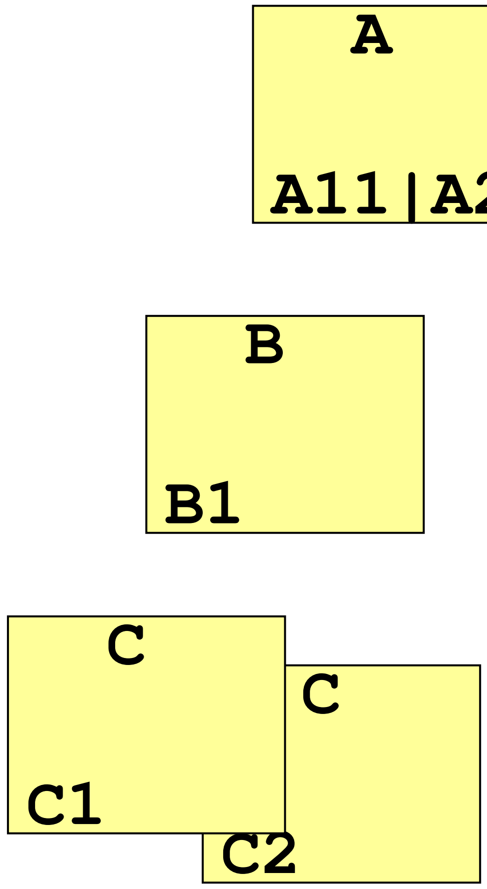
IMS 11 Open Database API JDBC enhancements

- Virtual Foreign Key fields
  - IMS Java maintains the unique keys for segments up to the root
  - SQL SELECT, INSERT, UPDATE, and DELETE queries
    - SQL syntax for IMS appears similar to standard SQL
- Updatable Result Sets
  - Update or delete of current row
- Metadata Discovery
  - Access the IMS Java Metadata classes generated by the DLIModel utility
- autoCommit support
  - updates are committed as they happen
- setFetchSize
  - An application can set the expected or desired number of rows to be returned

Virtual Foreign Keys
INSERT Example

INSERT INTO PCB.C (A, B, C)
VALUES (‘a1', ‘b1', ‘c2‘)



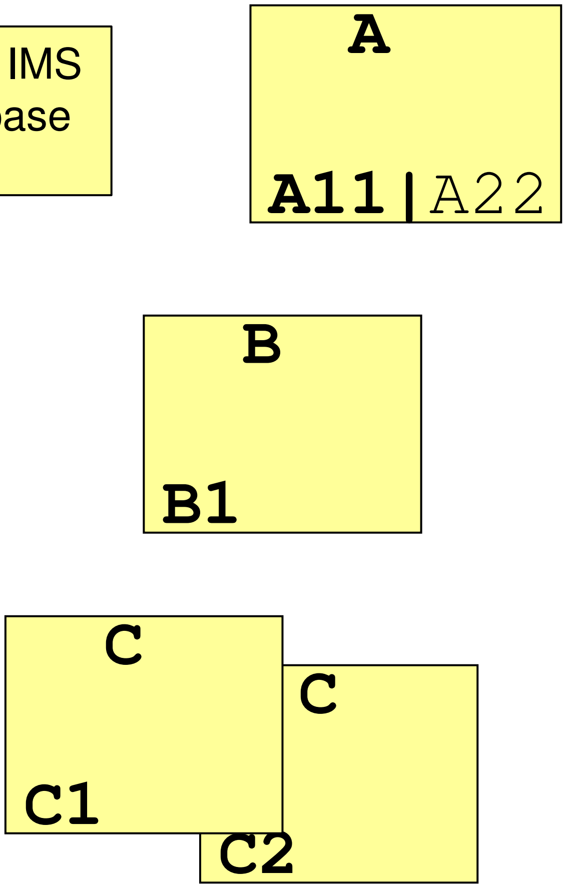


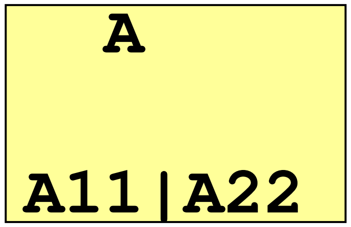
Updatable Result Sets Example

Query IMS Database

```
rs = st.executeQuery("SELECT A2 FROM PCB.A");

while(rs.next()){
                    rs.updateString("A2", "A22");
                    rs.updateRow();
         }
```

While processing result set update IMS Database





autoCommit occurs when the result set is closed or has no more rows

SQL keywords support

- XML Support
  - Retrieval, Storage

```
SELECT firstName, lastName, retrieveXML(Employees)
FROM DealerTable.Employees
WHERE serialNumber = ‘3A0140’
```

Build an XML document out of the Employee Segment and all dependant Segments in this PCB for the employee with serial number 3A0140.

IMS Universal DL/I Driver – DLI API for Java

- IMSConnectionSpec
  - Used by applications to pass connection request-specific properties to the PSBFactory.
- PSBFactory
  - A factory for creating PSB objects
- PSB
  - Used to obtain a reference to any PCB contained in the PSB. Contains one or more PCB objects.
- PCB
  - Provides the function to create an SSAList and issue database calls to retrieve, insert, update, and delete database information.
- SSAList
  - Represents a list of SSA (Segment Search Argument) objects.
- PathSet
  - Represents a collection of Path objects
- Path
  - Represents a collection of segments and their fields along a hierarchic path.

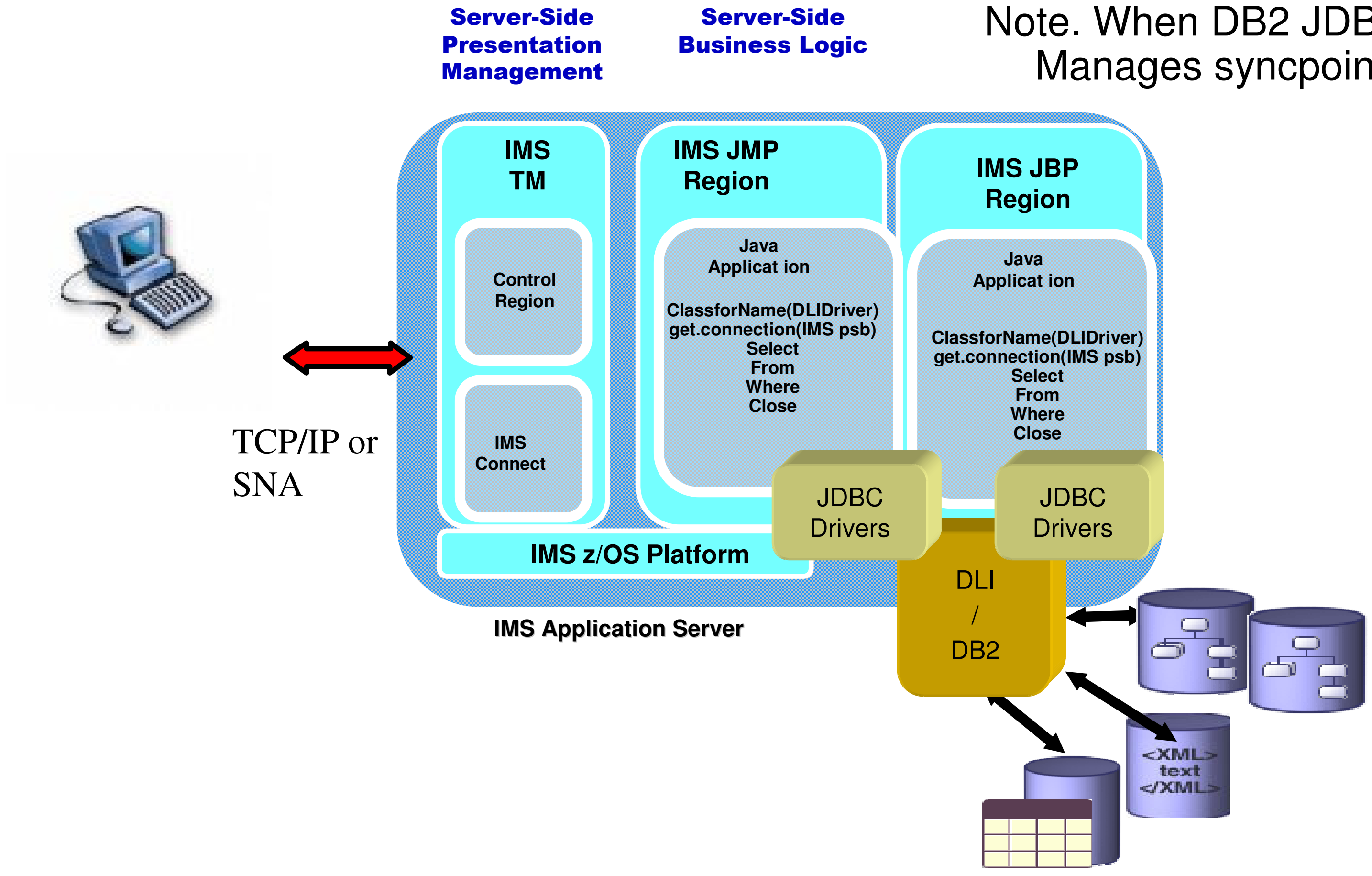
IMS TM Java

Commit/rollback of resources

Must be activated in the Java application by using IMSTransaction methods

Managed by IMS syncpoint processing

Note. When DB2 JDBC is used RRS Manages syncpoint processing



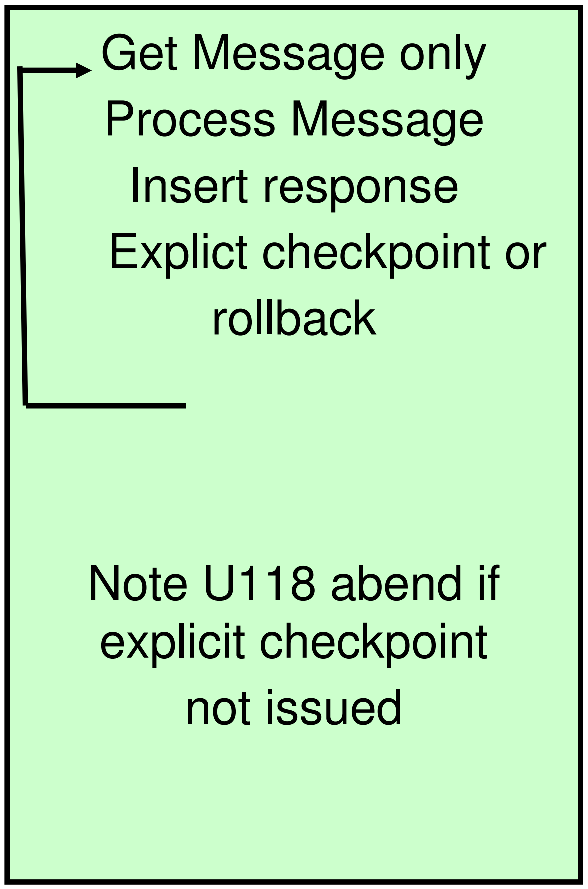
How does IMS know?

- There is a value JAVA that can be supplied for the LANG= parameter in the PSBGEN macro
  - LANG=JAVA can also be supplied in the APPLCTN macro for GPSBs
  - Specifying LANG=JAVA will result in the transaction being scheduled in a Java dependent region
    - When IMS receives the name of the transaction and looks up the PSB associated with the transaction code, if JAVA is specified the transaction will be queued to execute in a Java dependent region

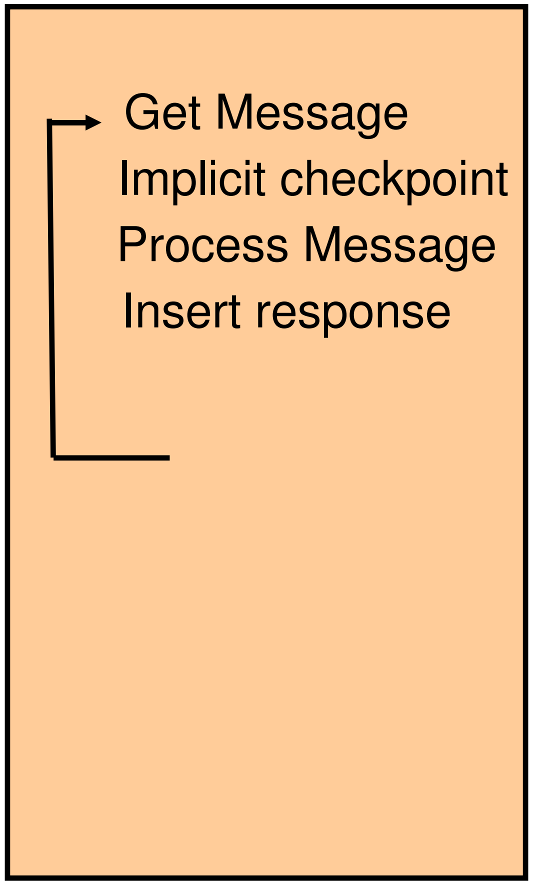
```
APPLCTN PSB=MYDLRPSB,PGMTYPE=TP,SCHDTYP=PARALLEL
TRANSACT CODE=JAVATRAN,PRTY=(7,10,2),INQUIRY=NO,MODE=SNGL,   X
   MSGTYPE=(SNGLSEG,NONRESPONSE,1)
```

```
MYDLRPCB  PCB   TYPE=DB,DBDNAME=DEALERDB,PROCOPT=AP,…
        SENSEG NAME=DEALER,PARENT=0,PROCOPT=AP
PSBGEN LANG=JAVA,PSBNAME=MYDLRPSB,CMPAT=YES,OLIC=YES
END
```

JDR/COBOL Interoperability


Get Message only
Process Message
Insert response
Explict checkpoint or rollback
Note U118 abend if explicit checkpoint not issued

IMS V9 JDR
Explicit commit Model



IMS V10 11 JDR
Standard commit Model

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

Define Input Messages

|LL|ZZ|TRANCODE|RequestCode|DealerName|DealerID

```
public class InputMessage extends IMSFieldMessage {
  final static DLITypeInfo[] messageInfo = {
    new DLITypeInfo("RequestCode", DLITypeInfo.INT,    1,  4),
    new DLITypeInfo("DealerName",  DLITypeInfo.CHAR,   5, 20),
    new DLITypeInfo("DealerID",    DLITypeInfo.INT,   25,  4)
  };

  public InputMessage() {
        super(messageInfo, 28, false);
  }
} // end InputMessage
```

Field type

Starting offset

Length

Message length

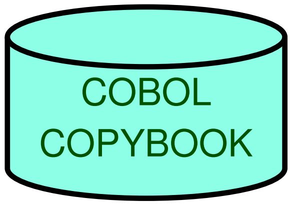
isSpa

NOTE: **Do not define LL, ZZ, and TRANCODE fields. Use getMessageLength and getTransactionCode methods provided by IMSFieldMessage to get length and transaction code.**

Define Output Messages

```
public class CanceledOrder extends IMSFieldMessage {

  final static DLITypeInfo[] cancelInfo = {
    new DLITypeInfo("Message",   DLITypeInfo.CHAR,   1,  30),
    new DLITypeInfo("OrderDate", "MMddYYYY", DLITypeInfo.DATE,  31,  8)
  };

  public Model() {
    super(cancelInfo, 38, false);
  }
}
```

Repeating Fields



```
01  MODEL-OUT.
    05  MODEL-COUNT    PIC 9(6).
    05  MODEL-INFO     OCCURS 100 TIMES.
        10  MAKE       PIC X(20).
        10  MODEL      PIC X(20).
        10  COLOR      PIC X(20).
```

```
public class ModelOutput extends IMSFieldMessage {

  static DLITypeInfo[] modelTypeInfo = {
     new DLITypeInfo("Make",   DLITypeInfo.CHAR,     1,  20)
     new DLITypeInfo("Model",  DLITypeInfo.CHAR,    21,  20)
     new DLITypeInfo("Color",  DLITypeInfo.CHAR,    41,  20)
   };

  static DLITypeInfo[] modelOutputTypeInfo = {
     new DLITypeInfo("ModelCount",  DLITypeInfo.INTEGER, 1,    4),
     new DLITypeInfoList("Models",  modelTypeInfo, 5, 60, 100)
  };

  public ModelOutput() {
     super(modelOutputTypeInfo, 6004, false);
  }
}
```

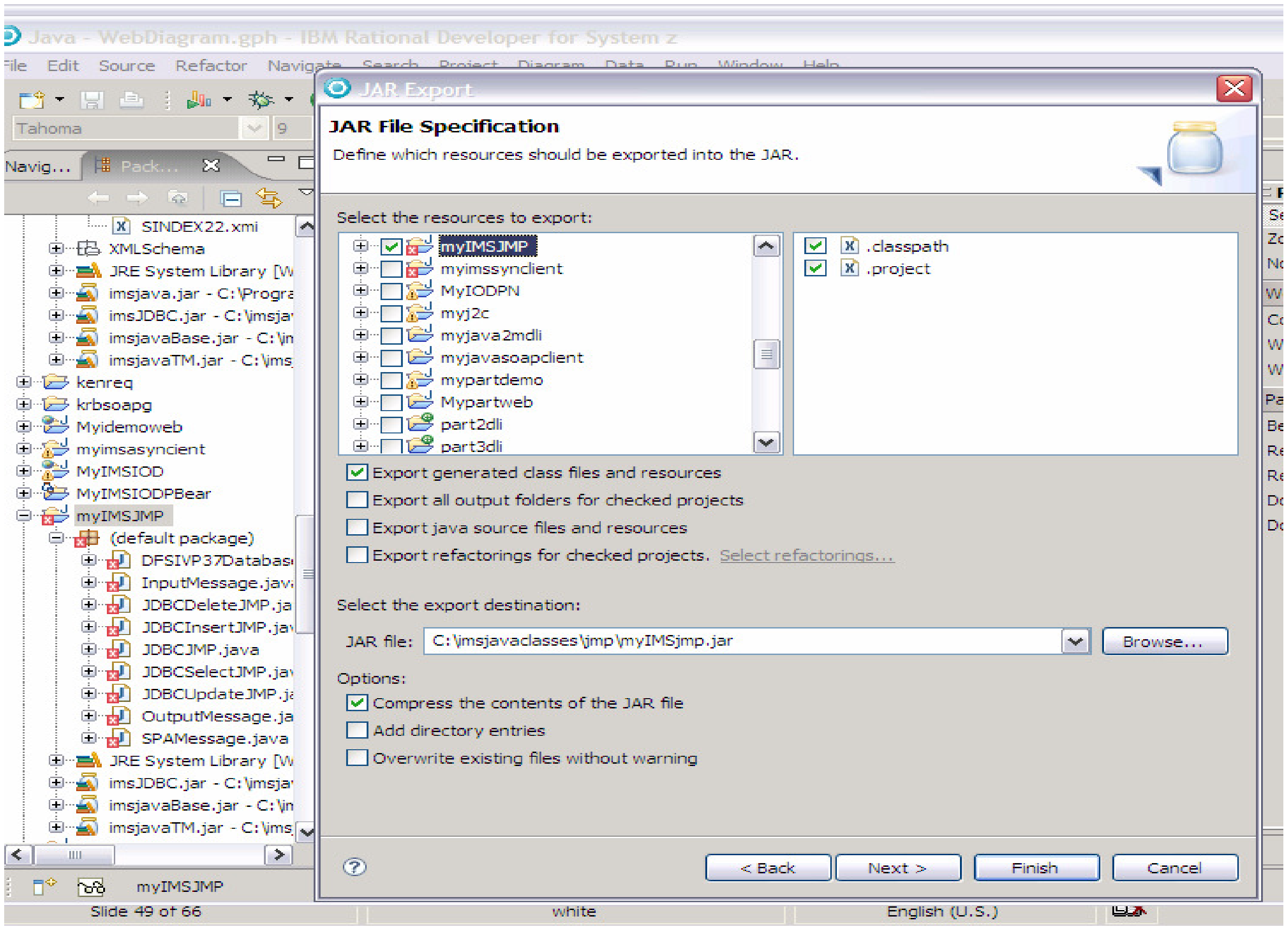
Total Length = 60*100 + 4

Starting Offset

Group Length

Repeat Count

RDz create .jar



JAR Export

JAR File Specification

Define which resources should be exported into the JAR.

Select the resources to export:

- [✓] myIMSJMP
- [ ] myimssynclient
- [ ] MyIODPN
- [ ] myj2c
- [ ] myjava2mdli
- [ ] myjavasoapclient
- [ ] mypartdemo
- [ ] Mypartweb
- [ ] part2dli
- [ ] part3dli

- [✓] .classpath
- [✓] .project

- [✓] Export generated class files and resources
- [ ] Export all output folders for checked projects
- [ ] Export java source files and resources
- [ ] Export refactorings for checked projects. Select refactorings...

Select the export destination:

JAR file: C:\imsjavaclasses\jmp\myIMSjmp.jar Browse...

Options:

- [✓] Compress the contents of the JAR file
- [ ] Add directory entries
- [ ] Overwrite existing files without warning

< Back   Next >   Finish   Cancel

IMS Java Application (JMP)

```
package samples.dealership;

public class IMSAuto {

    public static void main(String args []) {
        IMSAuto imsauto = new IMSAuto();

        IMSMessageQueue messageQueue = new IMSMessageQueue();
        FindCarInput  inputMessage   = new FindCarInput();
        FindCarOutput outputMessage = new FindCarOutput();

        try {
            while (messageQueue.getUniqueMessage(inputMessage)) {
                imsauto.proccessMessage(inputMessage, outputMessage);
                messageQueue.insertMessage(outputMessage.format());
            }
        } catch (IMSException e) {
            e.printStackTrace();
        }
    }
}
```

IMS JVMs

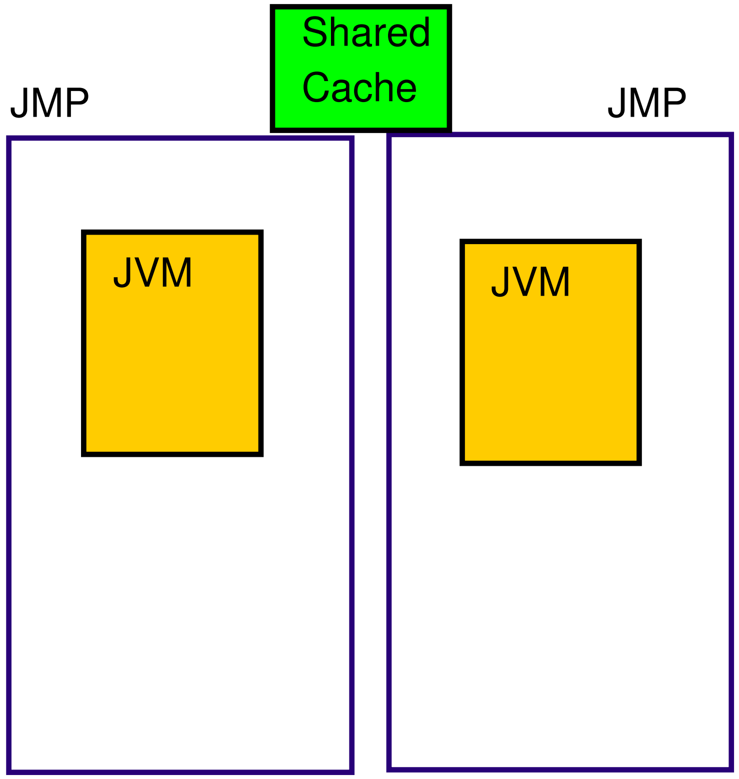
IMS.PROCLIB(DFSJVMAP)

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

DFSJVMMS -Djava.class.path=/ims/java/applications/IMSJavaPgm1

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

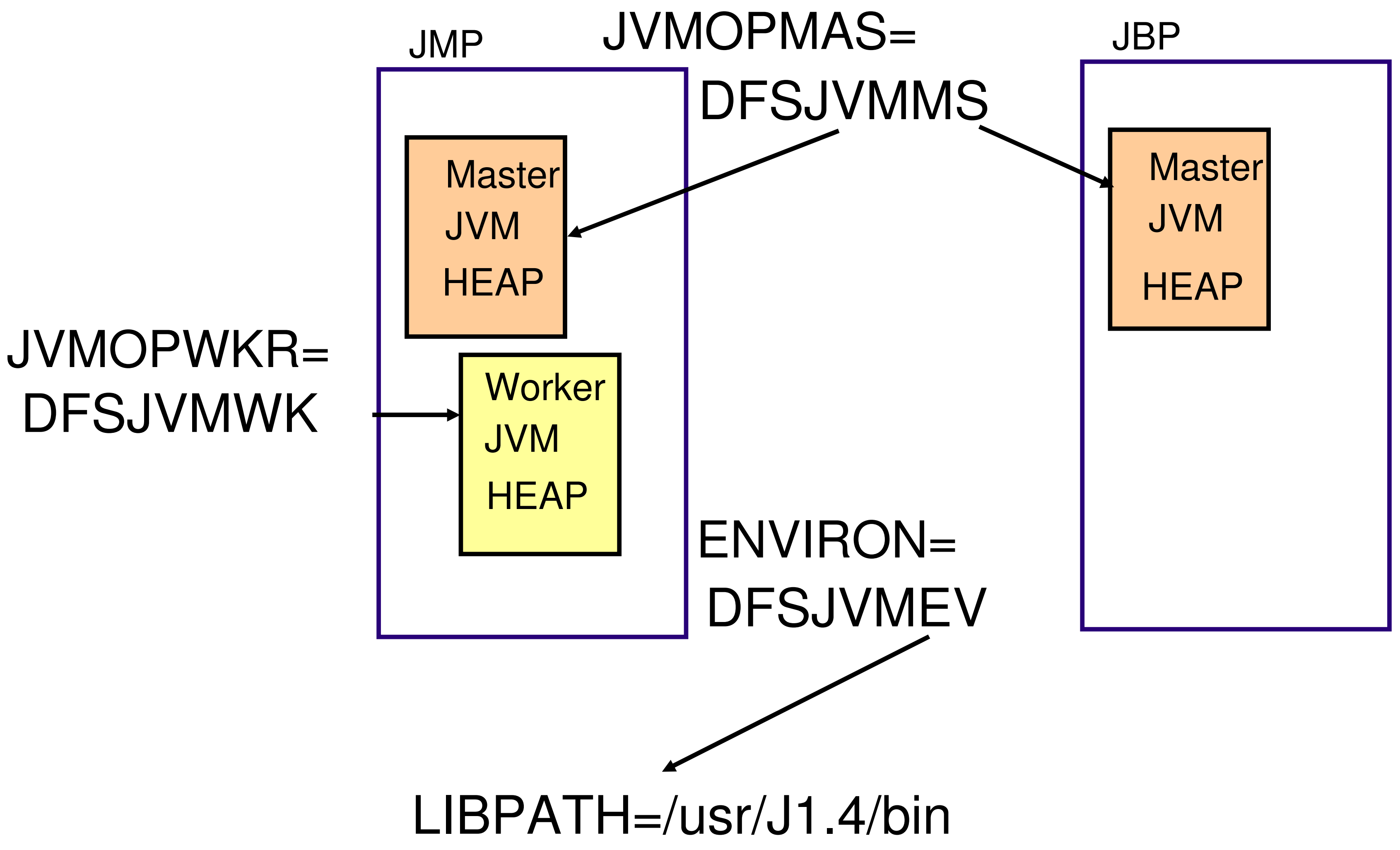
DFSJVMMS -Djava.class.path=/ims/java/applications/IMSJavaPgm2.jar

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

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

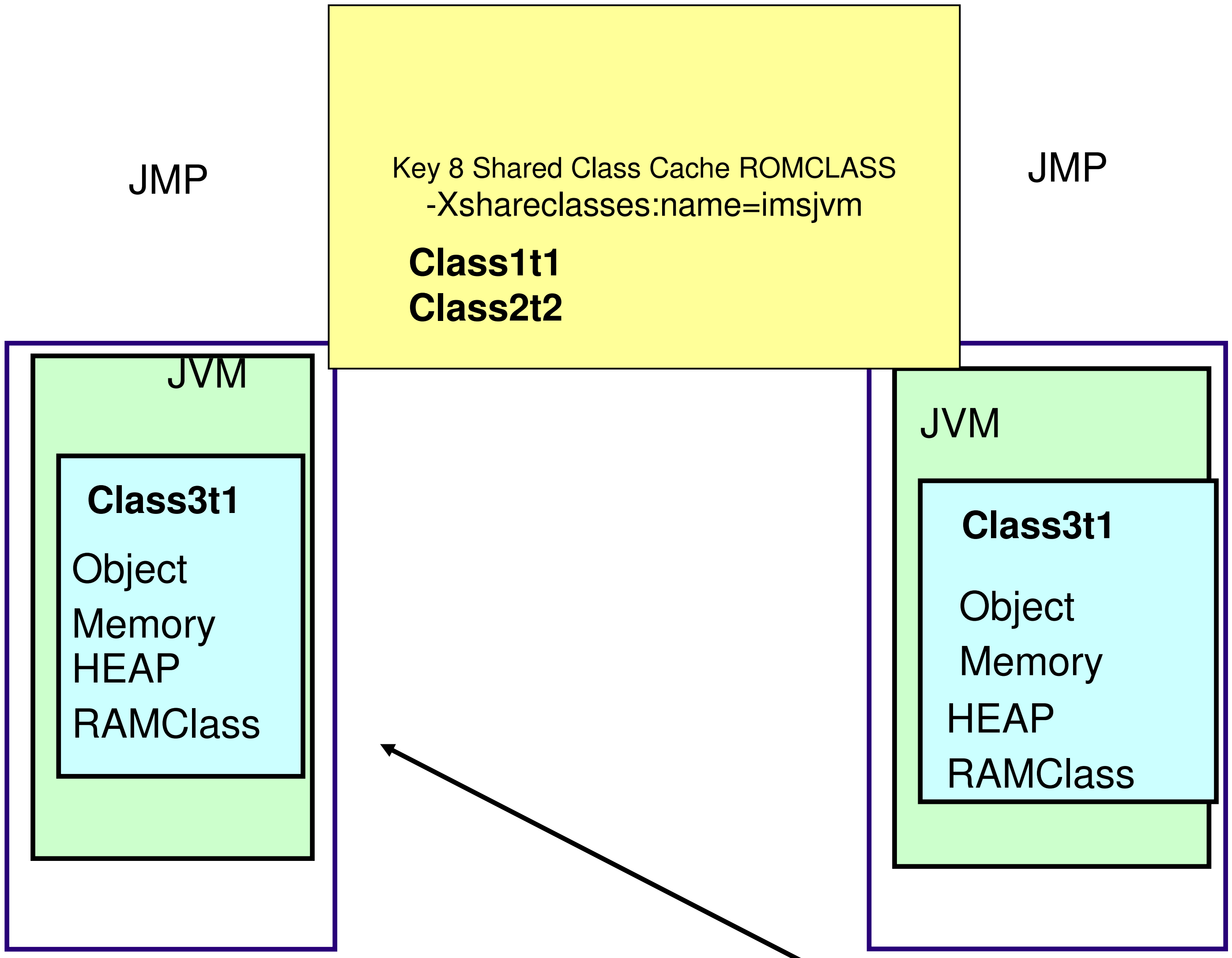
```
***************************************************************************
* Pathname for JAVAPGM2
***************************************************************************
JAVAPGM2=imsjava2/appl/jmp/IMSJavaPgm2
```

IMS V9
Resettable JVM SDK V1.4.2



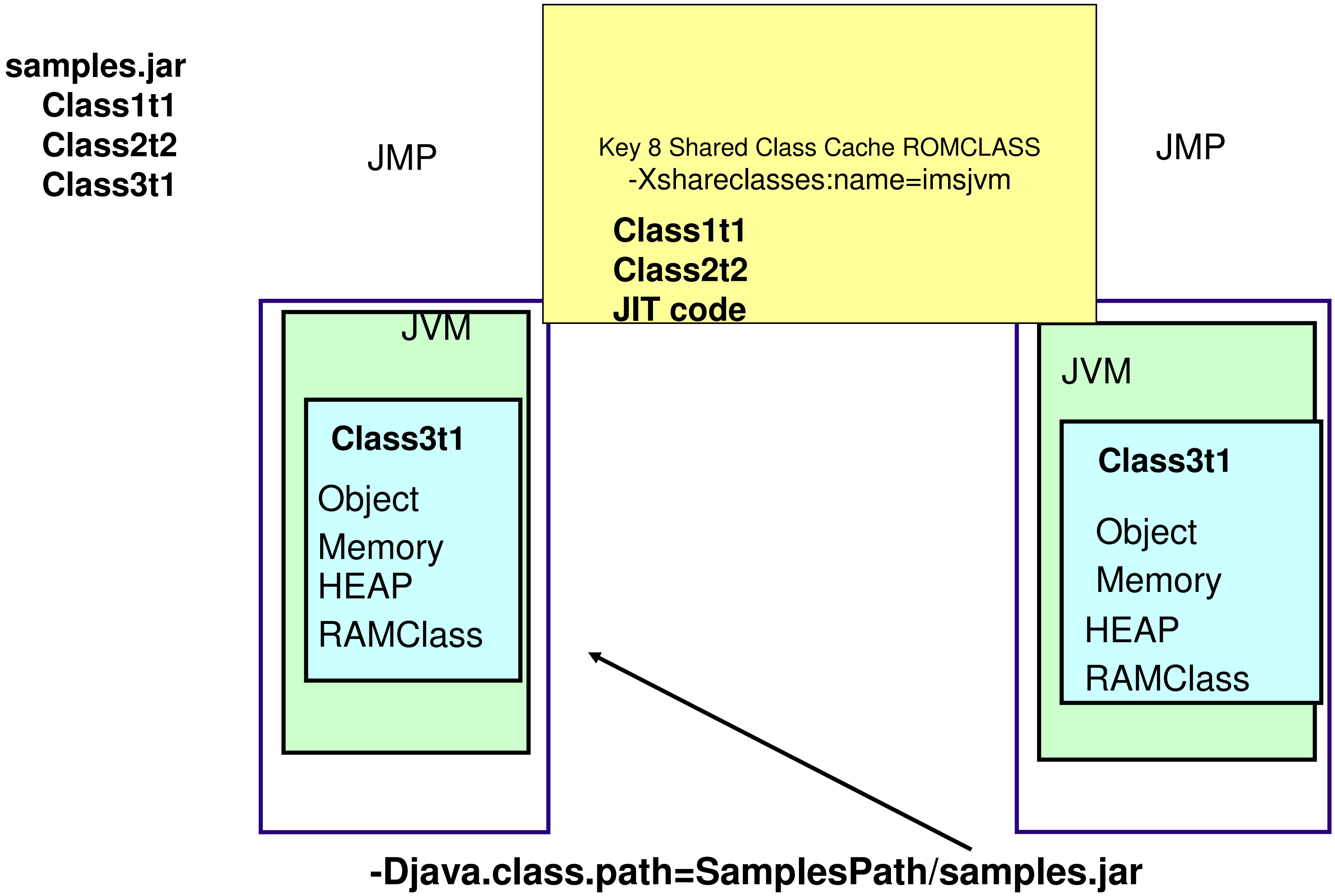
IMS 10 Shared Class Cache - SDK V5

samples.jar
- **Class1t1**
- **Class2t2**
- **Class3t1**

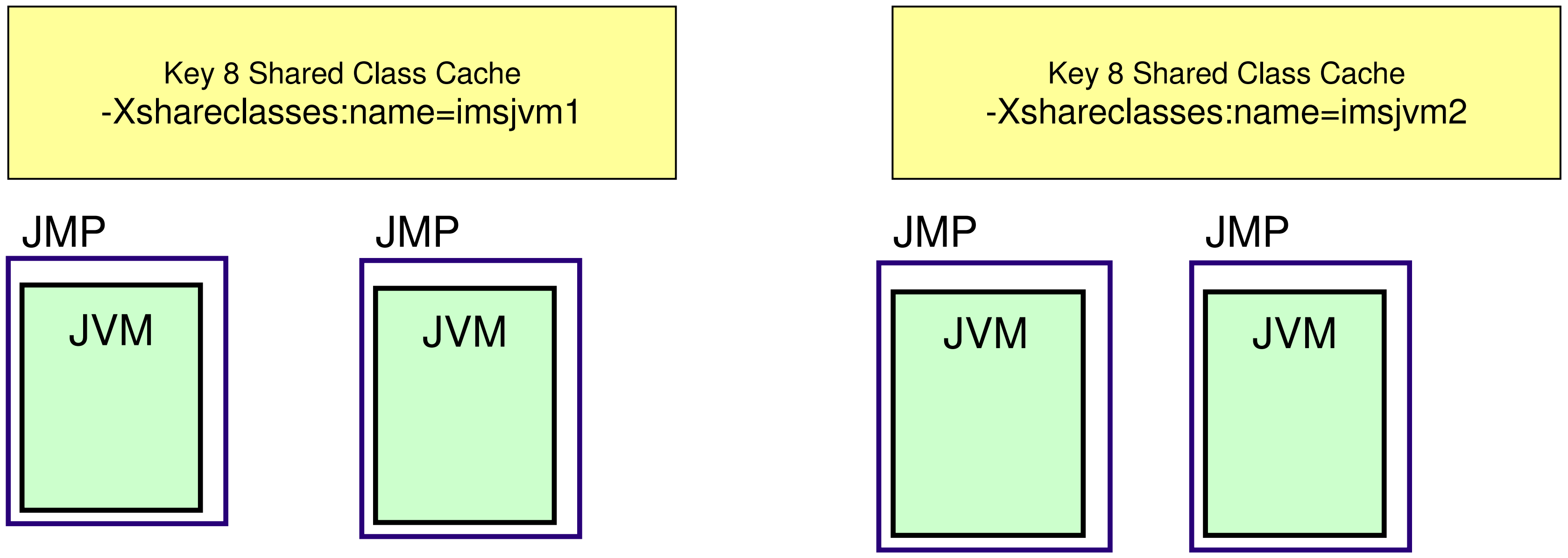


-Djava.class.path=SamplesPath/samples.jar

IMS 11 Shared Class Cache - SDK V6



Multiple Shared Class Cache



IMS 10 Java Class changes

- 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 meta data
      - 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

IBM SDK V5 for z/OS support
- IMS Java Application sample API

Current

```
public class CustomerApplication extends IMSApplication {
 public static void main(String args[]) {
        CustomerApplication myapp = new CustomerApplication();
        myapp.begin();
```

remove

remove

Modified

```
public class CustomerApplication  {
 public static void main(String args[]) {
        CustomerApplication myapp = new CustomerApplication();
```

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

Enable IMS Java Library Tracing

Enable And Set Trace Level

```
XMLTrace.enable("TestRun", XMLTrace.TRACE_DATA3);
```

Establish Output Stream

```
XMLTrace.setOutputStream(System.err);
```

or

```
XMLTrace.createOutputFile("tmp/TestRun.xml");
```

Close Trace

```
XMLTrace.close();
```

XMLTrace.libTraceLevel values

```
0                   none
TRACE_EXCEPTIONS    least
TRACE_CTOR1
TRACE_METHOD1
TRACE_DATA1
TRACE_CTOR2
TRACE_METHOD2
TRACE_DATA2
TRACE_CTOR3
TRACE_METHOD3
TRACE_DATA3         most
```

Sample Trace Output

```
  <?xml version="1.0"?>
- <IMSJavaTrace programName="AggregateTest" version="1.0">
    <data name="Release" type="char">jims81</data>
    <data name="Level" type="char">L2002090501</data>
    <data name="Build Date" type="char">Thu Sep 05 16:43:41 PDT 2002</data>
  + <method name="JavaToDLI.initialize()">
  + <method name="DLIDriver.connect(String, Properties)">
  + <method name="testCountAggregate()">
  + <method name="testSumAggregate()">
  + <method name="testMaxAggregate()">
  - <method name="testMinAggregate()">
    + <method name="DLIStatement(Connection, DLIConnection, int, int)">
    - <method name="DLIStatement.executeQuery(String)">
        <parameter name="sql" type="char">SELECT Min(Year) AS OldestCar
          FROM Dealer.ModelSegmen</parameter>
        <method name="DLIStatement.clearWarnings"/>
        <method name="SSAList(String)"/>
      - <method name="DLISQLException(String, String)">
          <parameter name="reason" type="char">"Dealer.ModelSegmen" is an
            undefined segment (table) name.  SQLSTATE=42704</parameter>
          <parameter name="sQLState" type="char">42704</parameter>
        </method>
      </method>
    </method>
  + <method name="testAvgAggregate()">
  + <method name="testGroupByColumnNameDoesNotExist()">
  + <method name="testAsClauseOverridesDefault()">
  + <method name="DLIConnection.close()">
  + <method name="IMSTransaction.commit()">
  </IMSJavaTrace>
```

IMS Java debug

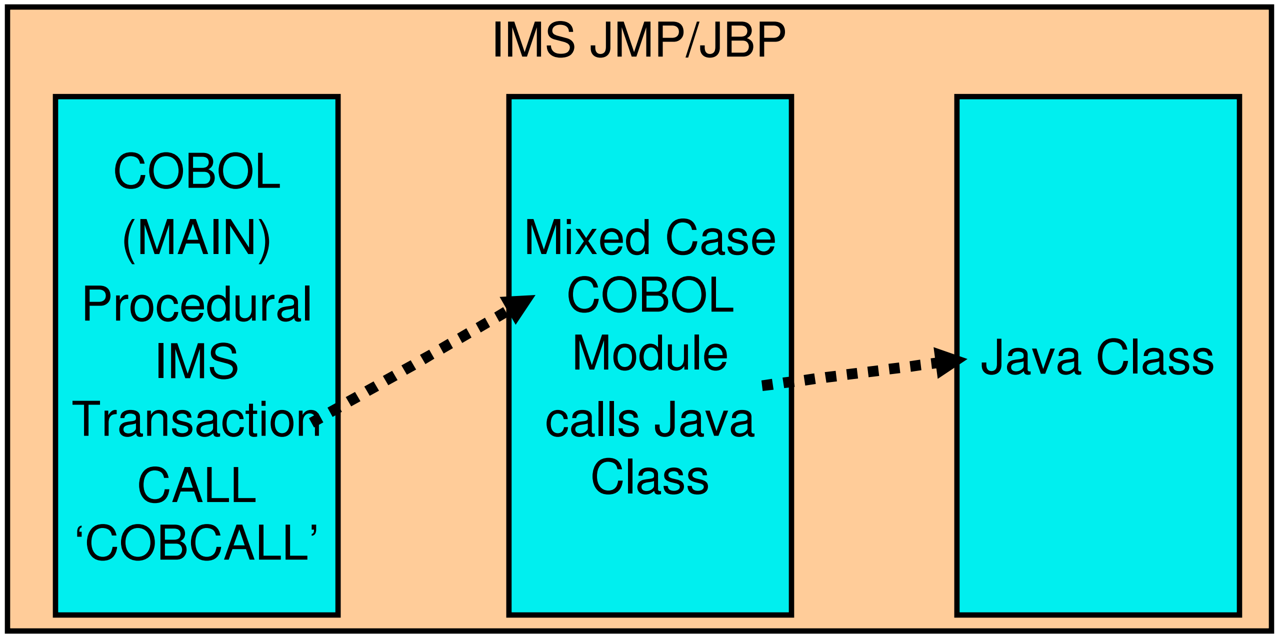
- Remote debug
  - Java Debugger (JDB)
  - TCP/IP socket

```
java -Xdebug -Xrunjdwp:transport=dt_socket,server=y,address=<port> <class>

jdb -attach <host>:<port>
```

- IMS Support
  - V9: APAR = PK66882   PTF = UK40576
  - V10: APAR = PK74919   PTF = ?
  - V11: APAR = ?

COBOL calls Java in JMP/JBP

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


IMS JMP/JBP
COBOL (MAIN) Procedural IMS Transaction CALL ‘COBCALL’
Mixed Case COBOL Module calls Java Class
Java Class

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

How Do I Get It?

- IMS Integration Suites
  - http://www.ibm.com/software/data/ims/toolkit/
    - IMS TM resource adapter
    - IMS DB resource adapter and JDBC driver (information)
    - IMS XML DB (information)
    - IMS DLIModel utility
    - IMS MFS Web support
    - IMS SOAP Gateway
- FMID for DB resource adapter, JDBC driver, and XML-DB support
  - IMS V8 (JMK8806)
  - IMS V9 (JMK9906)
    - Also Includes XML-DB support
  - IMS V10 (JMK1016)
    - Also Includes XQuery support
- AlphaWorks for XQuery beta
  - http://www.alphaworks.ibm.com
    - Search for ‘Virtual XML Garden’