E31

IMS Java Application Development

Christopher Holtz

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

RETURN TO INDEX

Overview

- **IMS Java**
  - **What Is IMS Java**
  - **Why Use IMS Java**
  - **Java Database Standards**
  - **IMS Java Class Library Architecture**
- **DL/I Model Utility**
- **Dealer Database Example**
  - **Generating DL/I Metadata**
  - **JMP Application**
  - **Message Queue**
  - **SQL Query**
- **Compile**
- **JMP / JBP Setup**
- **Run**

What is IMS Java?

IMS

- **A new feature in IMS v7**
- **A set of classes that...**
  - **Offers Java support to access IMS Databases from various environments (IMS, CICS, DB2, WebSphere)**
  - **Enables SQL access through the JDBC interface**
- **Java Virtual Machine (JVM) support in dependent regions**
  - **JDK 1.3 support**
  - **JDBC 2.1 support**
  - **Just-In-Time (JIT) compilation**
  - **Resetable JVM**

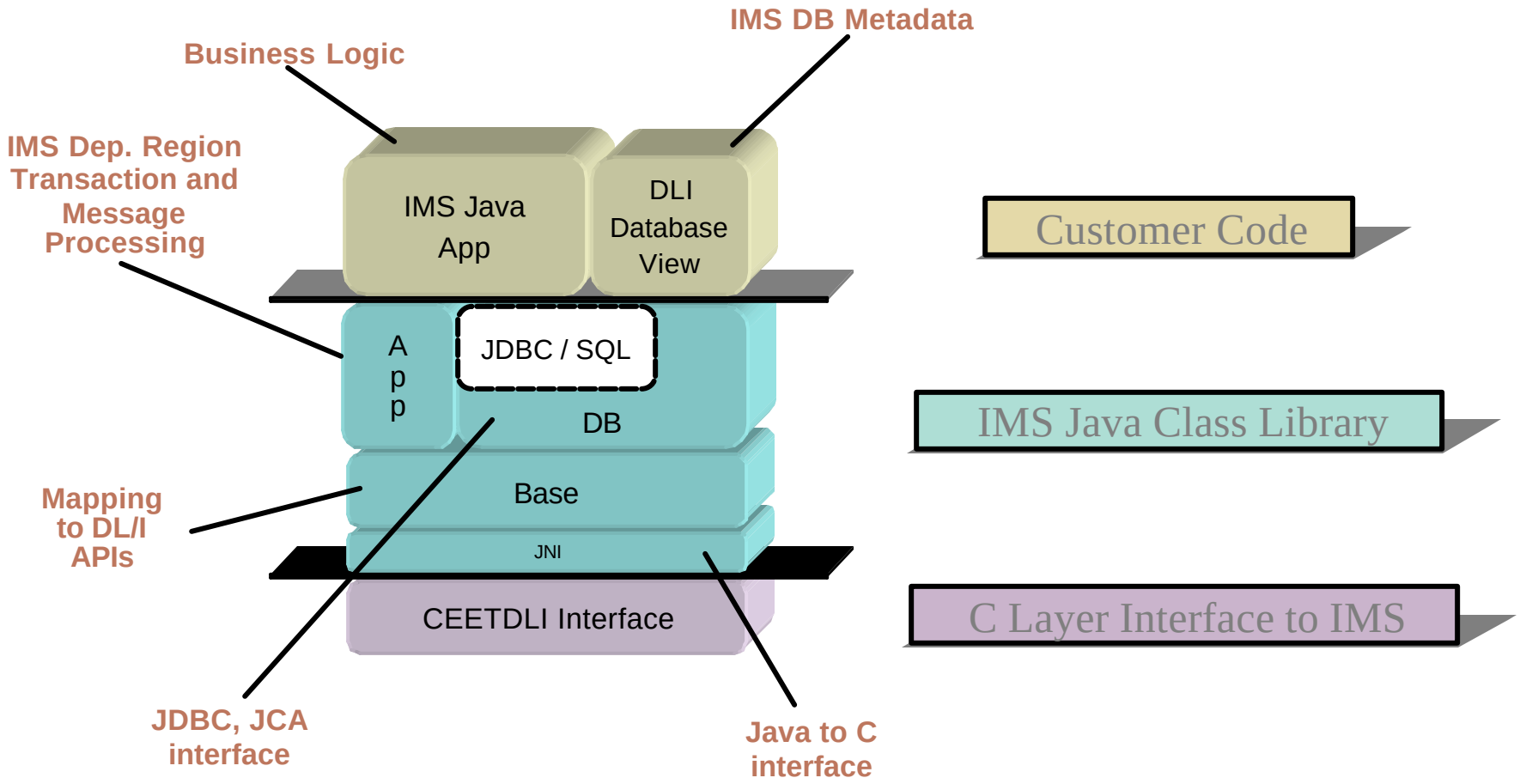
Why IMS Java?

IMS

- **Rapid Application Development**
  - **Reduce the Total Cost of Ownership (TCO) for IT and Data Management needs and Total Time to Value (TTV)**
- **Extend the life and scope of IMS applications**
  - **Minimum amount of impact on core applications and effort for developers, system programmers, and DBAs**
- **Leverage existing marketplace, industry-sanctioned standards - they are the slowest changing and most persistent**
  - **JDBC and J2EE are standards – help to minimize specific back end knowledge of IMS**
- **Leverage new and abundant skills in the marketplace and mitigate the loss of 390 skills for customers**
- **Integrate with other products**
  - ☆ **Our response is IMS Java, Web Services, WebSphere support, CICS support, DB2 SP support**

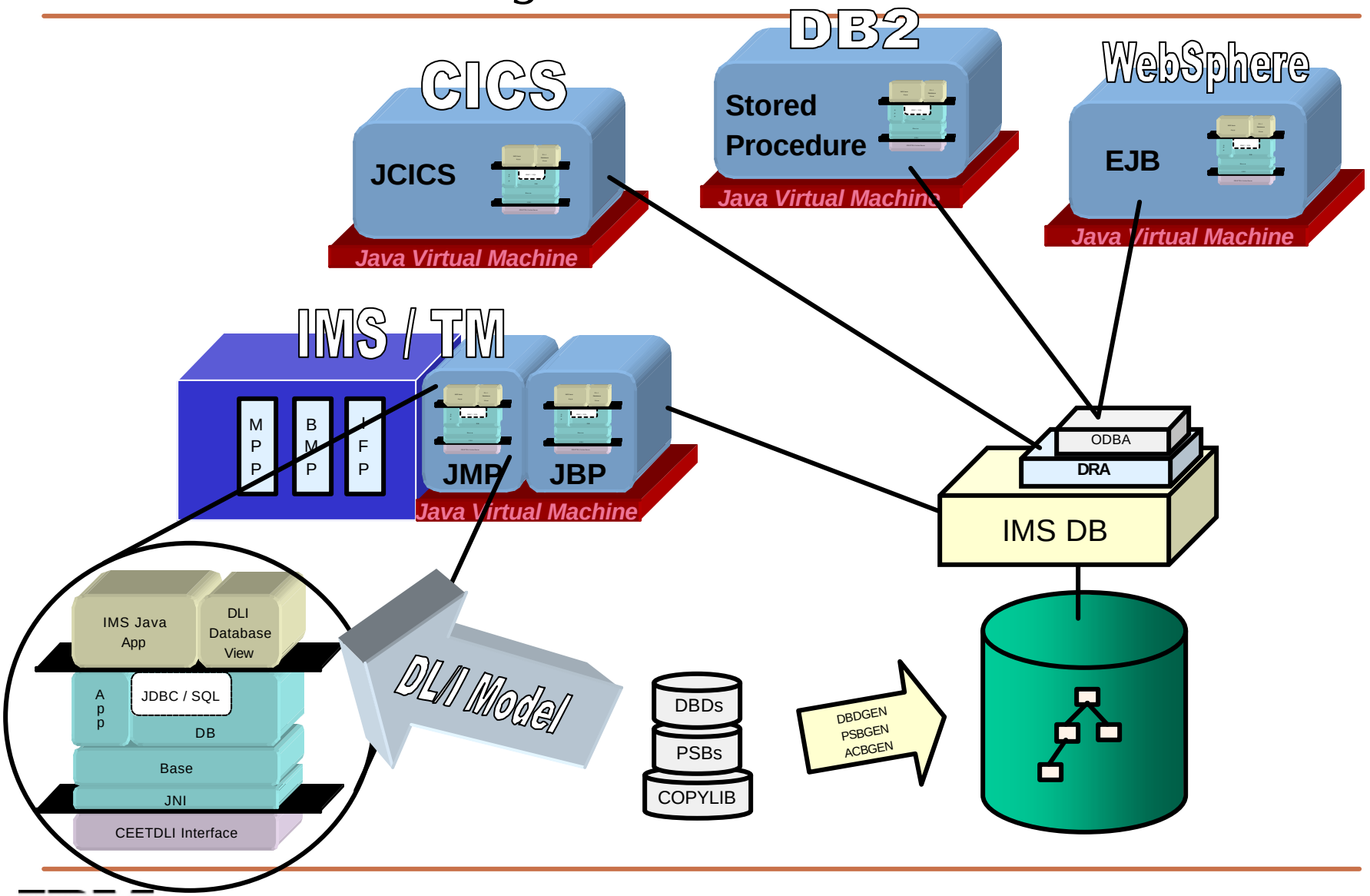
Java Class Library

IMS



IMS Java – The Big Picture

IMS



JDBC and J2EE Standards Evolution

IMS

- **Standard way to Query Database (relational)**
  - **Structured Query Language (SQL)**
- **Communicating Query to Database**
  - **Open Database Connectivity (ODBC) – C based**
- **Standard API to Query Database**
  - **“Java Database Connectivity” (JDBC) – Platform/DB Independent**
- **Standard API to Establish Connection**
  - **J2EE Connection Architecture (JCA, J2C)**
- **Standard API to Build Enterprise Applications**
  - **Java 2 Enterprise Edition (J2EE)**

- **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**

IMS JDBC Obstacles

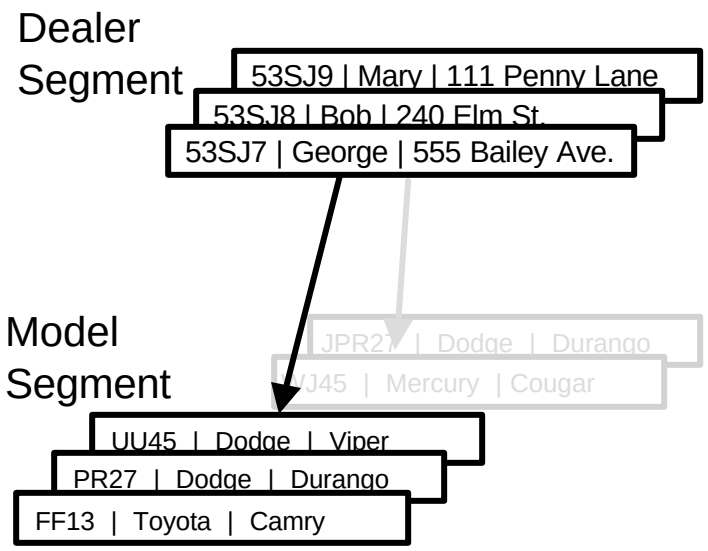
IMS

- **IMS uses Segment Search Arguments (SSA) not SQL**
  - **Internal SQL-to-SSA Parser (with modified SQL syntax)**
- **No Runtime Metadata Catalog**
  - **DLIDatabaseView Class**
- **No Access to DLI Data from Java**
  - **JNI-to-CEETDLI Interface**
- **No Persistent (JVM) in IMS Dependent Regions**
  - **JMP (analogous to MPP)**
  - **JBP (analogous to non-message driven BMP)**

Hierarchical to Relational Terminology Mapping

IMS

Hierarchical Design



Relational Design

Dealer Table

	DealerID	DealerName	DealerAddress
0	53SJ7	George	555 Bailey Ave.
1	53SJ8	Bob	240 Elm St.
2	53SJ9	Mary	111 Penny Ln.
...	...	...	...

Relational JOIN

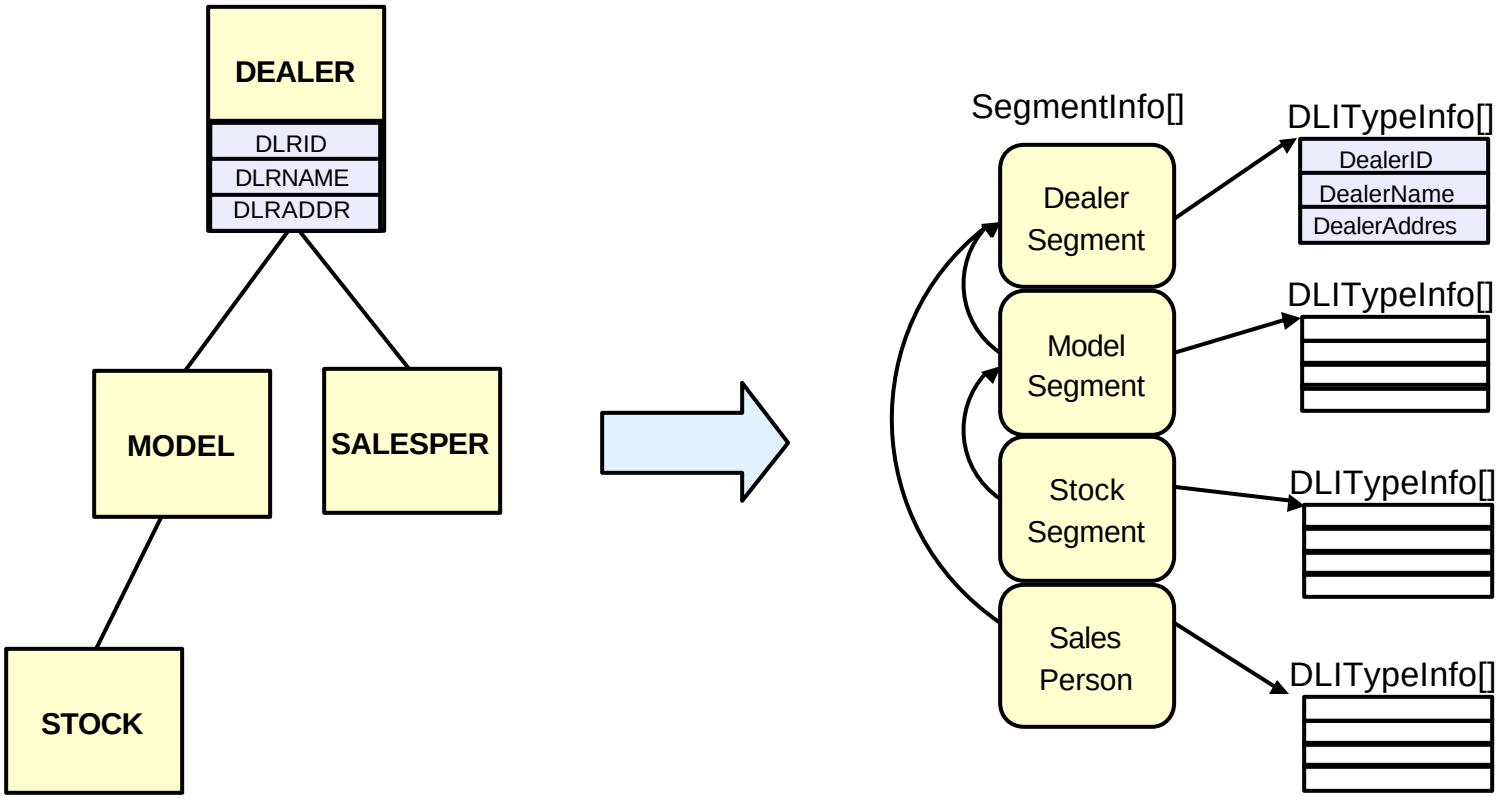
Model Table

ID	Make	Model	Dealer	
UU45	Dodge	Viper	53SJ7	**0**
PR27	Dodge	Durango	53SJ7	**0**
FF13	Toyota	Camry	53SJ7	**0**
JR27	Dodge	Durango	53SJ8	1
WJ45	Mercury	Cougar	53SJ8	1
...	...	...	...	...

Note: **Segment Names ~ Table Names**
Segment Instances ~ Table Rows
Field Names ~ Column Names

IMS Metadata

IMS



DBDLIB, PSBLIB

DLIDatabaseView

COBOL, SQL, and IMS Java Data Types

IMS

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

Datatype Conversion

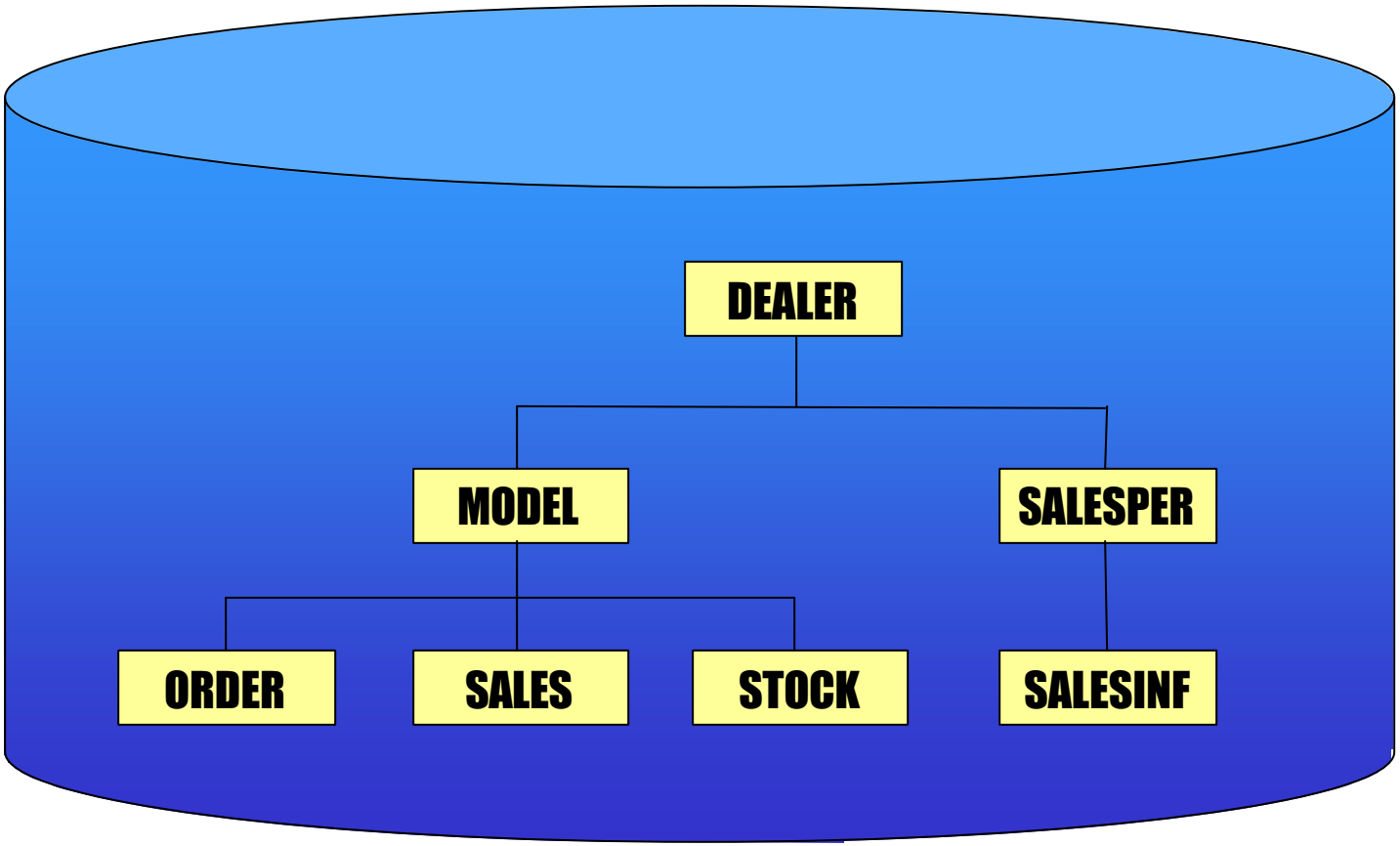
IMS

	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

Dealership Sample Database

IMS



SQL

```
SELECT Dealer.Name, Dealer.Phone, Order.LastName
FROM SomePCB.Order
WHERE  Model.MSRP > ‘50000’
        AND  Order.Date >= ‘5/1/2003’
        AND  Order.Date <= ’5/31/2003’
```

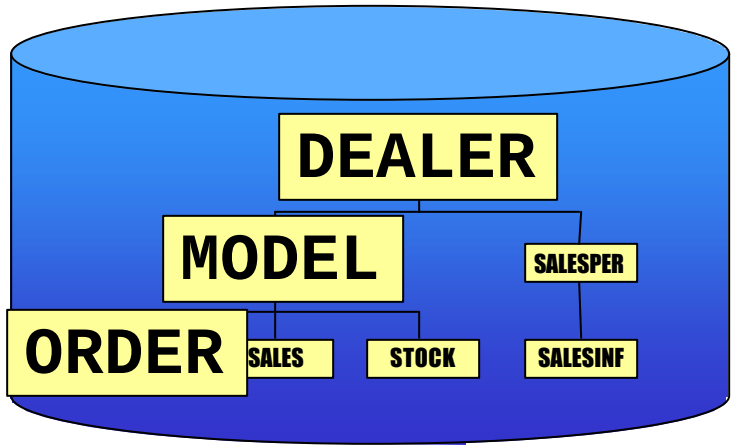
SSA List

SQL

```
SELECT Dealer.Name, Dealer.Phone, Order.LastName
FROM SomePCB.Order
WHERE  Model.MSRP > '50000'
       AND  Order.Date >= '5/1/2003'
       AND  Order.Date <= '5/31/2003'
```

SSA List

DEALER
MODEL
ORDER



SQL

```
SELECT Dealer.Name, Dealer.Phone, Order.LastName
FROM SomePCB.Order
WHERE   Model.MSRP > '50000'
        AND  Order.Date >= '5/1/2003'
        AND  Order.Date <= '5/31/2003'
```

SSA List

```
DEALER
MODEL    (MSRP     GT50000)
ORDER    (DATE     GE20030501&
           DATE    LE20030531
```

SQL

```
SELECT Dealer.Name, Dealer.Phone, Order.LastName
FROM SomePCB.Order
WHERE  Model.MSRP > ‘50000’
       AND  Order.Date >= ‘5/1/2003’
       AND  Order.Date <= ’5/31/2003’
```

SSA List

```
DEALER   *D
MODEL    (MSRP     GT50000)
ORDER    (DATE     GE20030501&
           DATE    LE20030531)
```

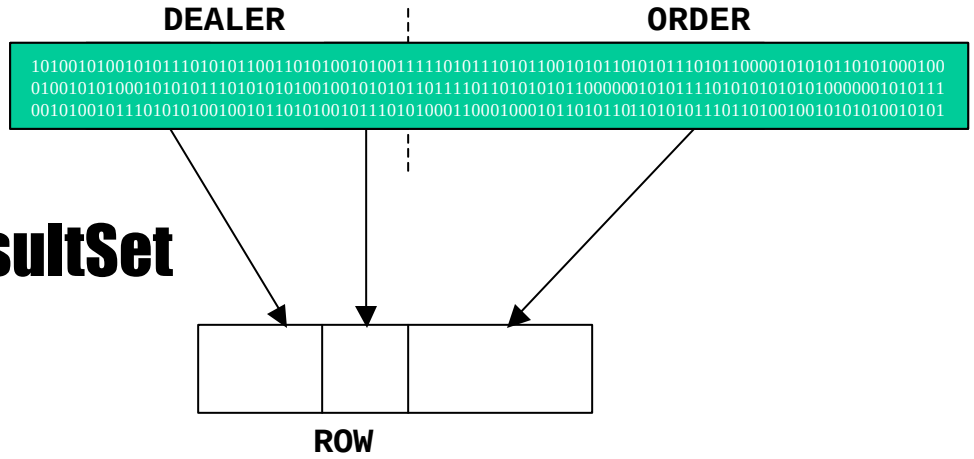
SQL

SELECT Dealer.Name, Dealer.Phone, Order.LastName

SSA List

```
DEALER  *D
MODEL   (MSRP    GT50000)
ORDER   (DATE    GE20030501&
         DATE    LE20030531)
```

IOArea



ResultSet

Overview

- IMS Java
  - What Is IMS Java
  - Why Use IMS Java
  - Java Database Standards
  - IMS Java Class Library Architecture
- DL/I Model Utility
- Dealer Database Example
  - Generating DL/I Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- Compile
- JMP / JBP Setup
- Run

DLIDatabaseView (online catalog)

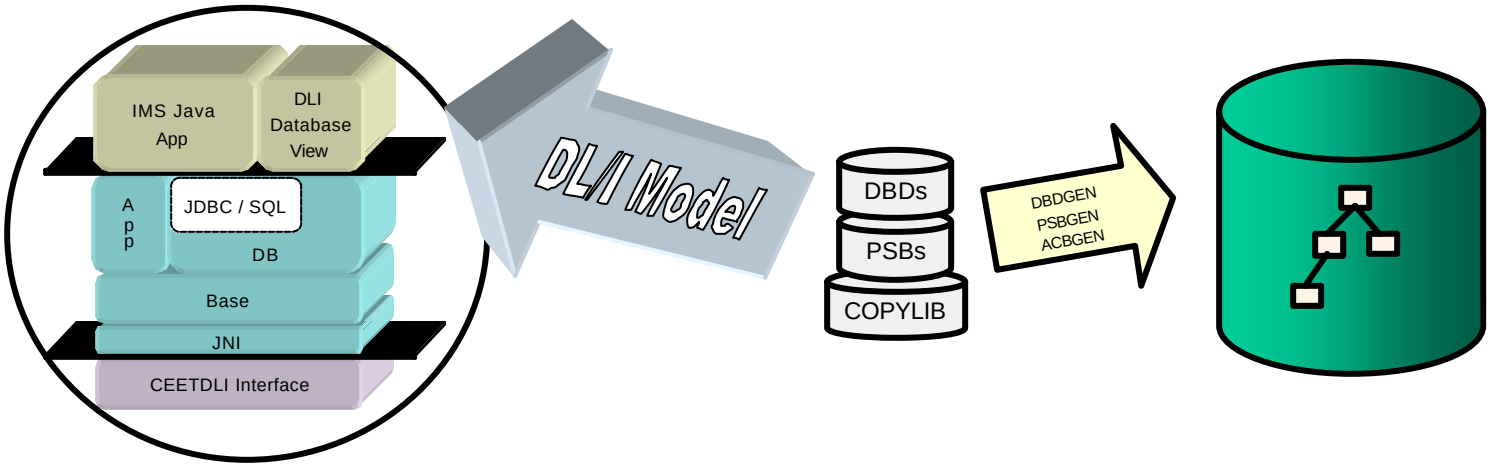
IMS

```
package samples.dealership;

import com.ibm.ims.db.*;
import com.ibm.ims.base.*;

public class AUTPSB11DatabaseView extends DLIDatabaseView {
  // The following DLITypeInfo[] array describes Segment: DEALER in PCB: AUTOLPCB
  static DLITypeInfo[] AUTOLPCBDEALERArray= {
      new DLITypeInfo("DealerNo",    DLITypeInfo.CHAR,  1,  4, "DLRNO"),
      new DLITypeInfo("DealerName",  DLITypeInfo.CHAR,  5, 30, "DLRNAME"),
      new DLITypeInfo("DealerCity",  DLITypeInfo.CHAR, 35, 10, "CITY"),
      new DLITypeInfo("DealerZip",   DLITypeInfo.CHAR, 45, 10, "ZIP"),
      new DLITypeInfo("DealerPhone", DLITypeInfo.CHAR, 55,  7, "PHONE")
  };
  static DLISegment AUTOLPCBDEALERSegment= new DLISegment
          ("DealerSegment","DEALER",AUTOLPCBDEALERArray,61);
  ...

  // An array of DLISegmentInfo objects follows to describe the view for PCB: AUTOLPCB
  static DLISegmentInfo[] AUTOLPCBarray = {
      new DLISegmentInfo(AUTOLPCBDEALERSegment,DLIDatabaseView.ROOT),
      new DLISegmentInfo(AUTOLPCBMODELSegment,0),
      new DLISegmentInfo(AUTOLPCBORDERSegment,1),
      new DLISegmentInfo(AUTOLPCBSALESSegment,1),
      new DLISegmentInfo(AUTOLPCBSTOCKSegment,1),
      new DLISegmentInfo(AUTOLPCBSTOCSALESegment,4),
      new DLISegmentInfo(AUTOLPCBSALESINFSegment,5)
  };
  ...
}
```

DL/I Model Utility

IMS

- **Parse DBD, PSB and Control Statements (COBOL Copylib)**
- **Produce XMI to act as a standard form of IMS Metadata**
- **Generate the IMS Java metadata (DLIDatabaseView) from the XMI**



DL/I Model Utility

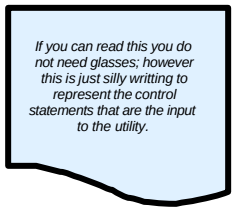
IMS

Control statements:

1) Choose PSBs/DBDs

2) Choose copybook members

3) Aliases, data types, new fields.



COBOL
copybook members

PSB

DBD

DLIModel Utility

XMI 1.2

IMS Java classes

IMS Java report

Overview

- IMS Java
  - What Is IMS Java
  - Why Use IMS Java
  - Java Database Standards
  - IMS Java Class Library Architecture
- DL/I Model Utility
- Dealer Database Example
  - Generating DL/I Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- Compile
- JMP / JBP Setup
- Run

IMS Java Dealership Sample

IMS

- **JMP**
- **Search for a Car currently in stock at a dealership**
- **Input**
  - **Car Make**
  - **Car Year**
- **Output**
  - **Dealer Name**
  - **Number of cars in stock**
  - **Car Model**
  - **Lot**

Application Development Steps

IMS

- **Create Control Statements**
- **Run DL/I Model Utility**
  - **DLIDatabaseView Metadata**
  - **IMS Java Report**
- **Write Application**
- **Compile**
- **Execute**
- **Debug**
  - **IMS Java XML Tracing**

Control Statements

IMS

```
//*******************************
//      Options
//*******************************
OPTIONS PSBds=SAMPLE.PDS.AUTO       DBDds=SAMPLE.PDS.AUTO
        GenJavaSource=YES           OutPath=samples/dealership
        GenTrace=YES
        Package=samples.dealership
```

Control Statements

IMS

```
//*****************************
//   PSB + PCB Definitions
//*****************************
PSB psbName=AUTPSB11
  PCB pcbName=AUTOLPCB JavaName=Dealer
  PCB pcbName=AUTS1PCB JavaName=Order
  PCB pcbName=AUTS2PCB JavaName=DealerStock
  PCB pcbName=AUSI2PCB JavaName=SecIndx2
  PCB pcbName=EMPLPCB  JavaName=EmployeePCB
```

Control Statements

IMS

```
//*****************************
// Physical Segment Definitions
//*****************************
SEGM DBDName=AUTODB SegmentName=DEALER JavaName=DealerSegment
  FIELD Name=DLRNO   JavaType=INTEGER JavaName=DealerNo
  FIELD Name=DLRNAME                  JavaName=DealerName
  FIELD Name=CITY                     JavaName=DealerCity
  FIELD Name=ZIP                      JavaName=DealerZip
  FIELD Name=PHONE                    JavaName=DealerPhone
  XDFLD Name=XFLD2                    JavaName=SecIndxFldB

SEGM DBDName=AUTODB SegmentName=MODEL JavaName=ModelSegment
  FIELD Name=MODKEY                          JavaName=ModelKey
  FIELD Name=YEAR                            JavaName=Year
  FIELD Name=MSRP    JavaType=PACKEDDECIMAL
                     TypeQualifier=999999.99 JavaName=MSRP
  FIELD Name=COUNT   JavaType=INTEGER        JavaName=Count
 ...
```

Running the DL/I Model Utility

IMS

```
//DLIMODEL PROC DSNAME=,SOUT='*'                                        00010000
//*************************************************************        00020000
//* THIS PROC RUNS THE IMS JAVA UTILITY IN BATCH MODE                   00030000
//*************************************************************        00040000
//STEP1 EXEC PGM=BPXBATCH,                                              00050000
//   PARM='SH "/usr/lpp/ims/imsjava71/dlimodel/go" "&DSNAME"'           00060001
//STDENV DD DUMMY                                                       00070000
//STDOUT DD PATH='/tmp/&SYSUID..out',                                   00080000
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),                                    00090000
// PATHMODE=SIRWXU                                                      00100000
//STDERR DD PATH='/tmp/&SYSUID..err',                                   00110000
// PATHOPTS=(OWRONLY,OCREAT,OTRUNC),                                    00120000
// PATHMODE=SIRWXU                                                      00130000
//*--------------------------------------------                         00140000
//* Redirect stdout and stderr output to SYSOUT:                        00150000
//STEP2 EXEC PGM=IKJEFT01 ,DYNAMNBR=300,COND=EVEN                       00160000
//SYSTSPRT DD SYSOUT=&SOUT                                              00170000
//HFSOUT DD PATH='/tmp/&SYSUID..out'                                    00180000
//HFSERR DD PATH='/tmp/&SYSUID..err'                                    00190000
//STDOUTL DD SYSOUT=&SOUT,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)          00200000
//STDERRL DD SYSOUT=&SOUT,DCB=(RECFM=VB,LRECL=133,BLKSIZE=137)          00210000
//SYSPRINT DD SYSOUT=&SOUT                                              00220000
// PEND                                                                 00230000
```

DLIDatabaseView (online catalog)

IMS

```
package samples.dealership;

import com.ibm.ims.db.*;
import com.ibm.ims.base.*;

public class AUTPSB11DatabaseView extends DLIDatabaseView {
  // The following DLITypeInfo[] array describes Segment: DEALER in PCB: AUTOLPCB
  static DLITypeInfo[] AUTOLPCBDEALERArray= {
      new DLITypeInfo("DealerNo",    DLITypeInfo.INTEGER,  1,  4, "DLRNO"),
      new DLITypeInfo("DealerName",  DLITypeInfo.CHAR,     5, 30, "DLRNAME"),
      new DLITypeInfo("DealerCity",  DLITypeInfo.CHAR,    35, 10, "CITY"),
      new DLITypeInfo("DealerZip",   DLITypeInfo.CHAR,    45, 10, "ZIP"),
      new DLITypeInfo("DealerPhone", DLITypeInfo.CHAR,    55,  7, "PHONE")
  };
  static DLISegment AUTOLPCBDEALERSegment= new DLISegment
          ("DealerSegment","DEALER",AUTOLPCBDEALERArray,61);
  ...

  // An array of DLISegmentInfo objects follows to describe the view for PCB: AUTOLPCB
  static DLISegmentInfo[] AUTOLPCBarray = {
      new DLISegmentInfo(AUTOLPCBDEALERSegment,DLIDatabaseView.ROOT),
      new DLISegmentInfo(AUTOLPCBMODELSegment,0),
      new DLISegmentInfo(AUTOLPCBORDERSegment,1),
      new DLISegmentInfo(AUTOLPCBSALESSegment,1),
      new DLISegmentInfo(AUTOLPCBSTOCKSegment,1),
      new DLISegmentInfo(AUTOLPCBSTOCSALESegment,4),
      new DLISegmentInfo(AUTOLPCBSALESINFSegment,5)
  };
  ...
}
```

Java Report (programming guide)

IMS

```
DLIModel IMS Java Report
========================
Class: AUTPSB11DatabaseView  in package: samples.dealership  generated for PSB: AUTPSB11

================================================
PCB: Dealer
================================================
Segment: DealerSegment
Field: DealerNo    Type=INTEGER ++ Primary Key Field ++
Field: DealerName  Type=CHAR         (Search Field)
Field: DealerCity  Type=CHAR         (Search Field)
Field: DealerZip   Type=CHAR         (Search Field)
Field: DealerPhone Type=CHAR         (Search Field)
================================================
  Segment: ModelSegment
  Field: ModelKey  Type=CHAR                              ++ Primary Key Field ++
  Field: Year      Type=CHAR                                   (Search Field)
  Field: MSRP      Type=PACKEDDECIMAL TypeQualifier=999999.99 (Search Field)
  Field: Count     Type=INTEGER                               (Search Field)
  ================================================
      Segment: OrderSegment
      Field: OrderNo Type=CHAR ++ Primary Key Field ++
      ...
      Field: Time    Type=TIME      (Search Field)
      ================================================
      Segment: SalesSegment
      Field: SaleNo  Type=CHAR ++ Primary Key Field ++
      ...
```

Define Input Message

IMS

```
package samples.dealership;

public class FindCarInput extends IMSFieldMessage {
   final static DLITypeInfo[] fieldInfo = {
      new DLITypeInfo("InputMake",     DLITypeInfo.CHAR,       1, 5),
      new DLITypeInfo("InputYear",     DLITypeInfo.CHAR,       6, 4),
   };

   public FindCarInput() {
      super(fieldInfo, 9, false);
   }
}
```

IMS Java Application (JMP)

IMS

```
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();
        }
    }
}
```

Obtain a Connection

IMS

```
public void processMessage(FindCarInput inputMessage, FindCarOutput outputMessage) {
    Connection connection = null;
    try {
        Class.forName("com.ibm.ims.db.DLIDriver");
        String url = "jdbc:dli:samples.dealership.AUTPSB11DatabaseView";
        connection = DriverManager.getConnection(url);
    } catch (Exception e) {
        e.printStackTrace();
    }

    execute query...
    process results...
    close connection...
}
```

recall:

Class: AUTPSB11DatabaseView in package: samples.dealership generated for PSB: AUTPSB11

Execute Query

IMS

```
public void processMessage(FindCarInput inputMessage, FindCarOutput outputMessage) {
    obtain connection...

    String inputMake = inputMessage.getString("InputMake").trim();
    String inputYear = inputMessage.getString("InputYear").trim();

    String query =
        "SELECT StockSegment.Color, StockSegment.Lot, DealerSegment.DealerName, " +
        "ModelSegment.Make, ModelSegment.Model, ModelSegment.Year " +
        "FROM Dealer.StockSegment " +
        "WHERE ModelSegment.Make = '" + inputMake + "' " +
        "AND ModelSegment.Year = '" + inputYear + "'";

    Statement statement = connection.createStatement();
    ResultSet results = statement.executeQuery(query);

    process results...
    close connection...
}
```

Process Results

IMS

```
public void processMessage(FindCarInput inputMessage, FindCarOutput outputMessage) {
    obtain connection...
    execute query...

    while (results.next()) {

        CarDetails car = new CarDetails();
        car.dealerName = results.getString("DealerName");
        car.carMake    = results.getString("Make");
        car.carModel   = results.getString("Model");
        car.carYear    = results.getString("Year");
        car.lot        = results.getString("Lot");

        outputMessage.add(car);
    }

    close connection...
}
```

Close Connection

IMS

```
public void processMessage(FindCarInput inputMessage, FindCarOutput outputMessage) {
    obtain connection...
    execute query...
    process results...

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

Compile

- **Classpath must contain:**
  - **imsjava.jar (shipped with product)**
  - **Generated DLIDatabaseView (.java) (DLIModel utility)**
  - **Application Source Code (.java)**

```
export CLASSPATH= . : /usr/lpp/ims/imsjava81/imsjava.jar
```

- **Compile**

```
javac samples/dealership/*.java
```

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**
    - **Similar to the DFSMPR procedure for MPPs**
    - **Couple of new parameters**
    - **Several DFSMPR parameters not supported**

- **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**
    - **Similar to the IMSBATCH procedure for BMPs**
    - **Couple of new parameters**
    - **Several IMSBATCH parameters not supported**

JMP Setup

IMS

- **JCL PROC**
  - **Add location of DFSCLIB to STEPLIB**
  - **e.g. DQEIVP.ECDVL01.DLL**
- **DFSJVMAP**
  - **Alias PSB Name to map to Java Application Name**
    - **DFSIVP37=samples/ivp/ims/IMSIVP**
- **DFSJVMEV**
  - **Set location of libJavTDLI.so**
    - **LIBPATH=/usr/lpp/ims/imsjava81**
- **DFSJVMMS (Master JVM)**
  - **Set middleware classpath to IMS Java Jar (*Java ARchive*)**
    - **-Dibm.jvm.trusted.middleware.class.path=>**
    - **/usr/lpp/ims/imsjava81/imsjava.jar**
  - **Set application classpath to location of app code**
    - **-Dibm.jvm.sharable.application.class.path=>**
    - **/usr/lpp/ims/imsjava81/samples/samples.jar**
- **DFSJVMWK (Worker JVM)**

- **Bring up JMP Region (JCL)**
- **Schedule Transaction**

Enable Library Tracing

IMS

Enable And Set Trace Level

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

Establish Output Stream

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

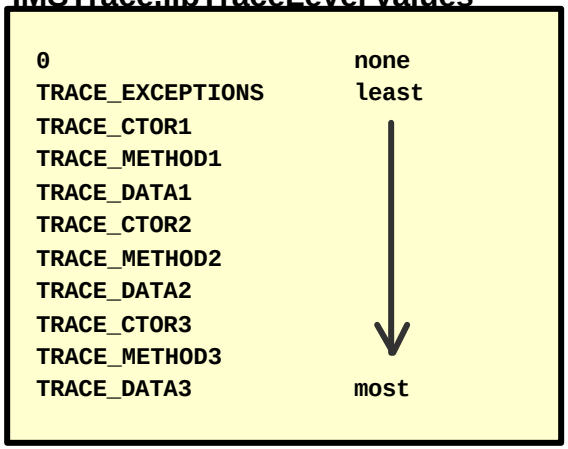
or

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

Close Trace

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

IMSTrace.libTraceLevel values



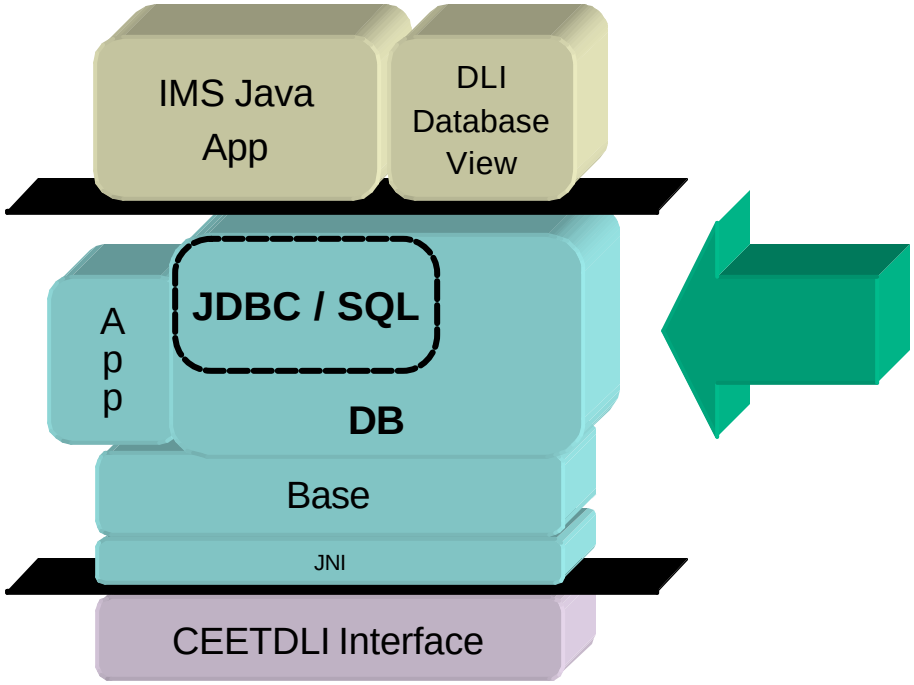
Sample Trace Output

IMS

```
  <?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>
```

JDBC 2.0 Enhancements

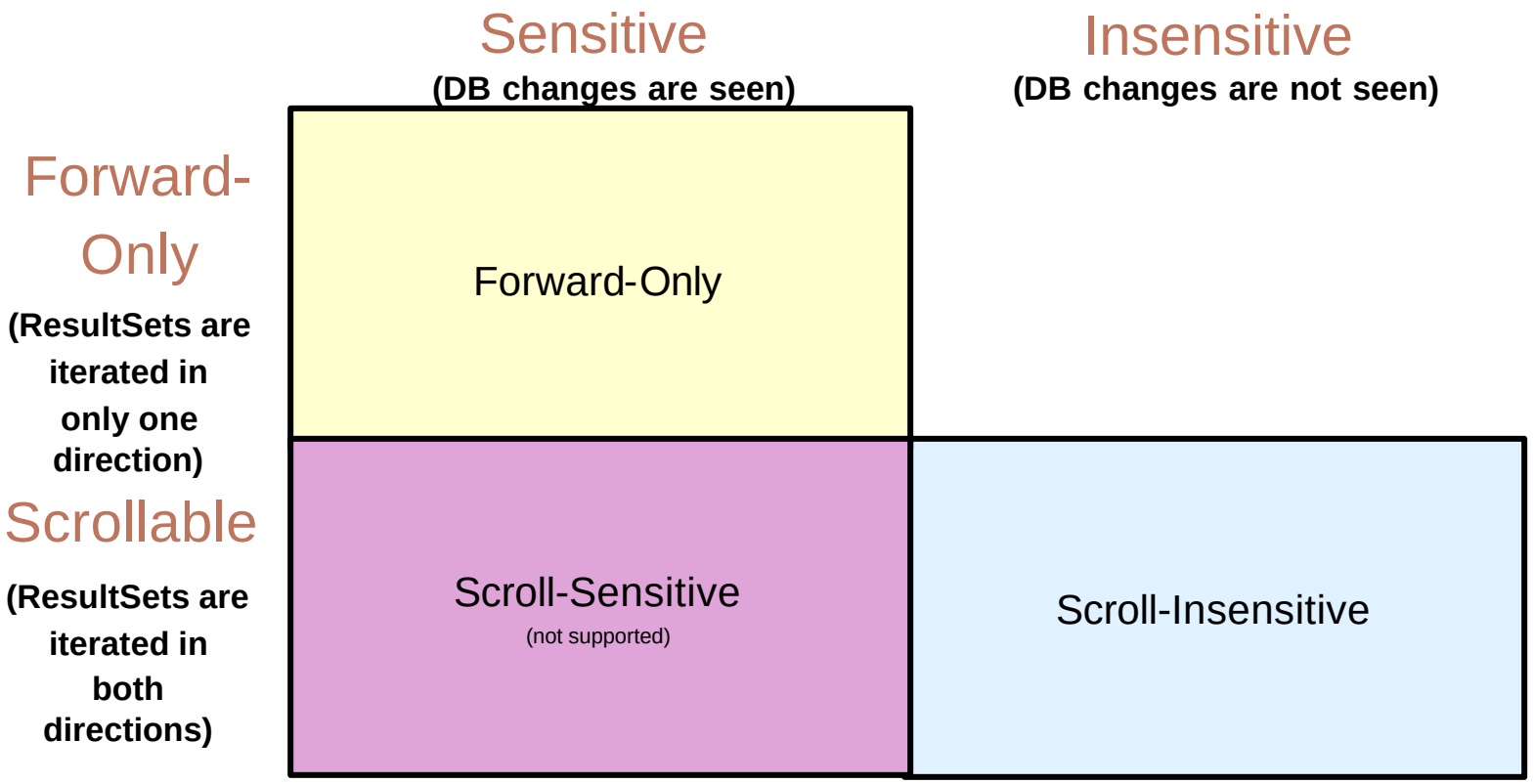
IMS



- **DataSource**
- **ResultSet features**
- **New SQL Keywords**

IMS Java Result Set Types

IMS



*IMS has no means to traverse a Query backwards

IMS Java Result Set Types

IMS

- **Forward-Only (currently supported) (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)**

IMS Java Result Set Concurrency

IMS

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

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

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

New SQL keywords support

IMS

- **Field Renaming**
  - **AS**

```
SELECT EMPNO AS EmployeeNumber
FROM Employees
```

Display all the values of EMPNO in a column labeled EmployeeNumber.

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

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

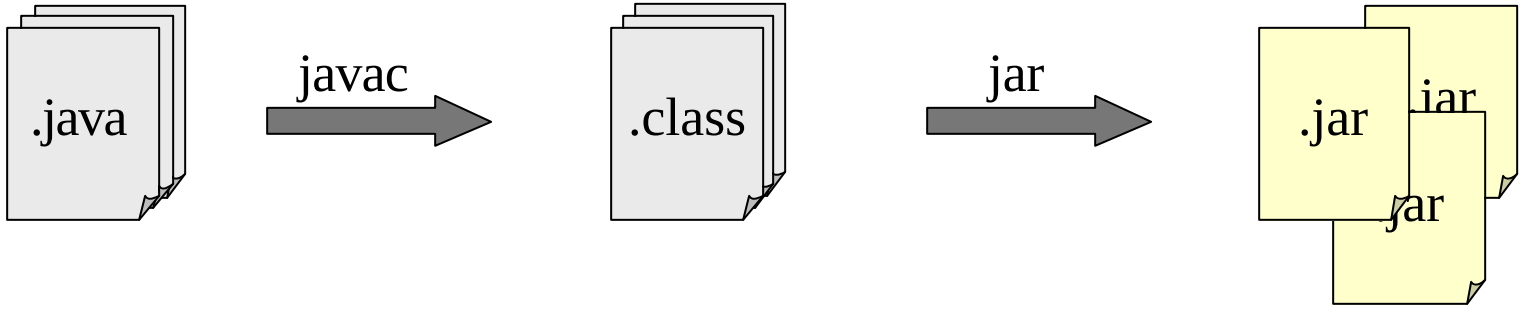
Display the average age per department.

- **Ordering**
  - **ORDER BY, ASC, DESC**

```
SELECT firstName, lastName, department
FROM 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.

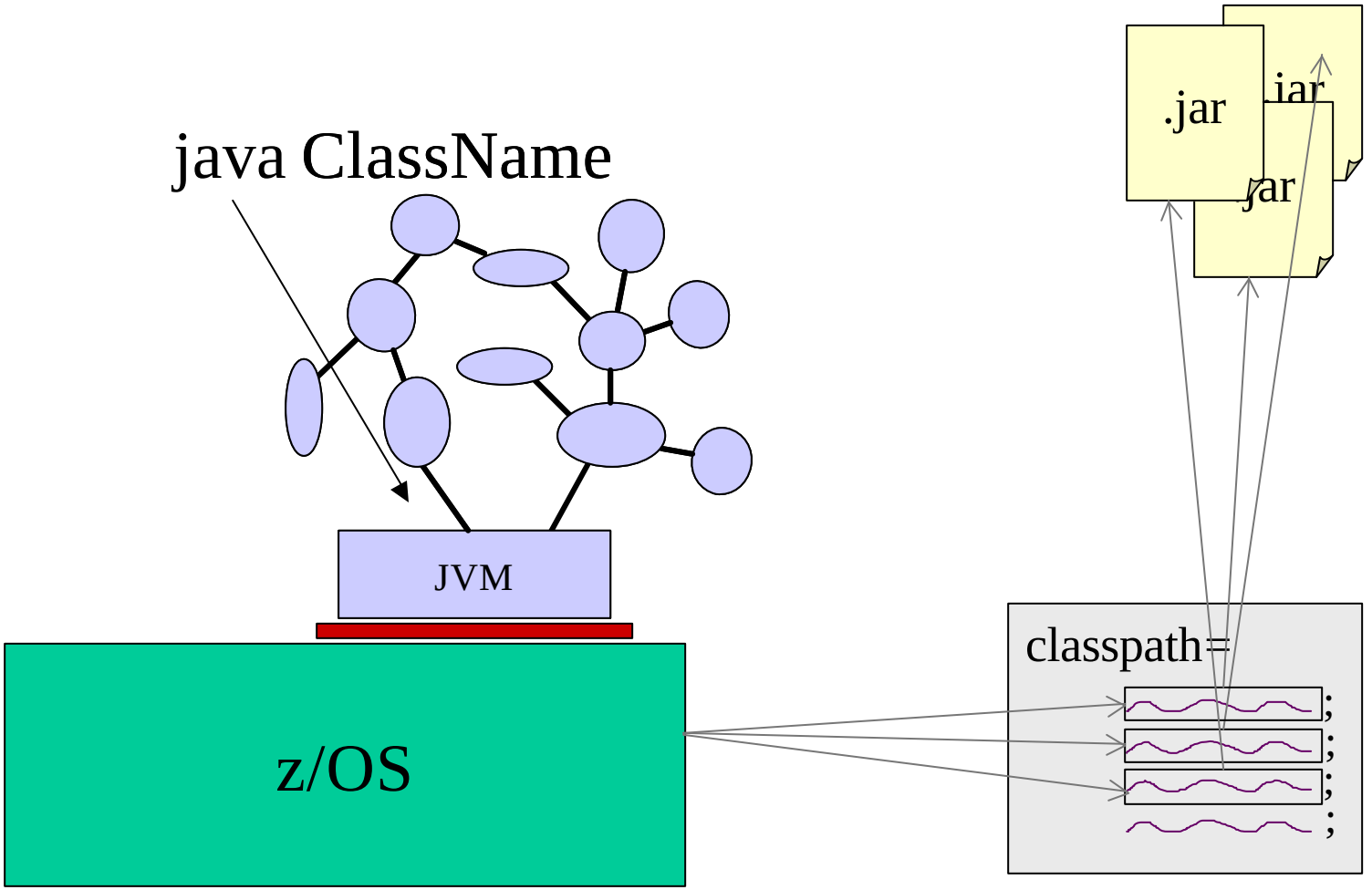
Java Compilation



z/OS

classpath=

;
;
;
;



Persistent Reusable JVM Runtime

IMS