E13

IMS Java Application Development

Kyle Charlet
charletk@us.ibm.com

St. Louis, MO **Sept. 30 - Oct. 3, 2002**

RETURN TO INDEX

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **DLIModel Utility**
- **Dealer Database Example**
  - Generating DLI Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- **Compile**
- **JMP / JBP Setup**
- **Running**

IBM Software

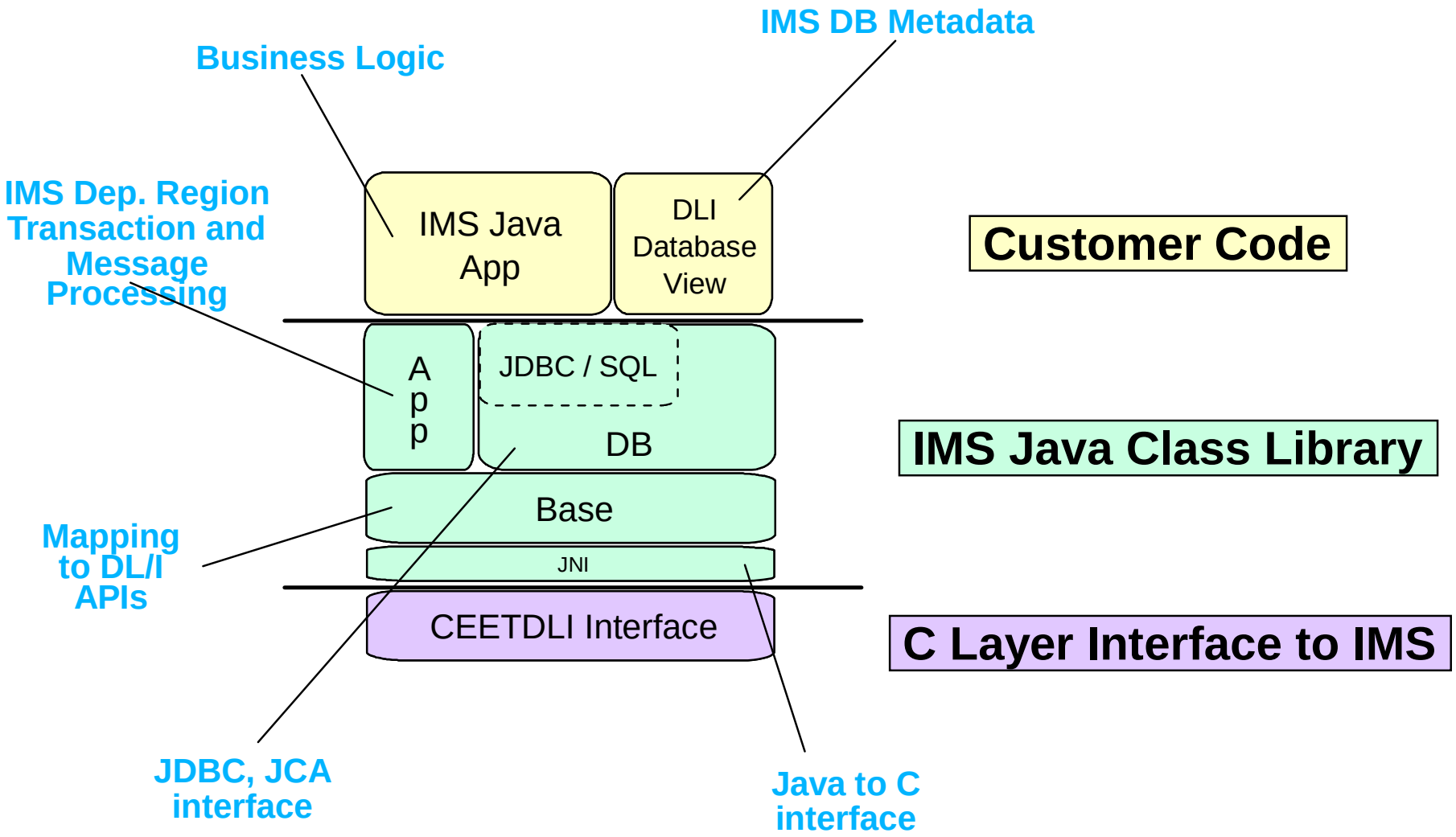
What is IMS Java?

- **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?

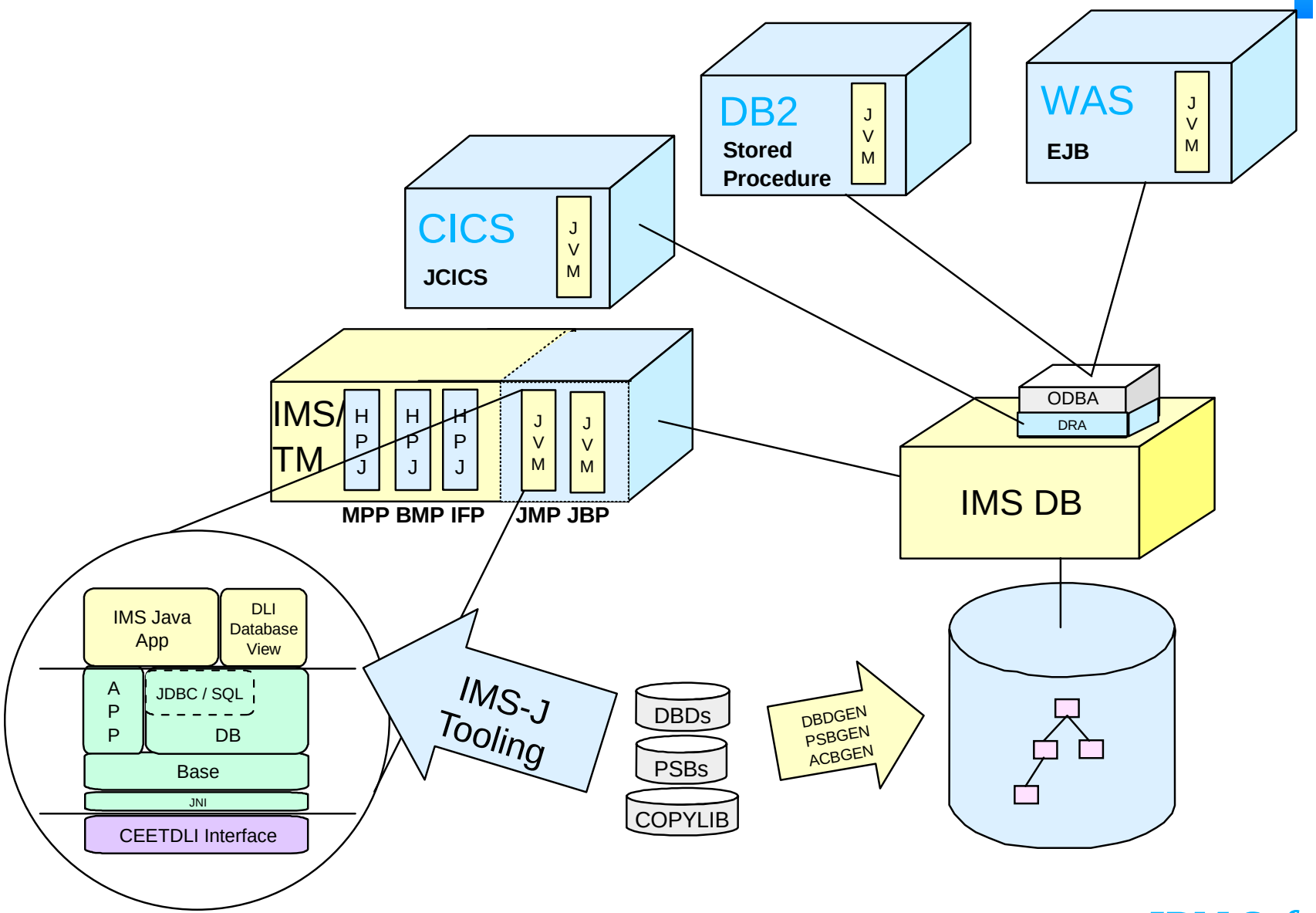
- **Colleges teach Java, very few still teach COBOL**
- **Colleges teach relational DBs with SQL access, very few teach hierarchical with SSA access**
- **JDBC is an industry standard**
  - Minimizes specific backend DB knowledge of IMS
- **Customer requests for Java support**
- **Rapid Application Development**

Java Class Library



IBM Software

IMS Java - The Big Picture



IBM Software

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **DLIModel Utility**
- **Dealer Database Example**
  - Generating DLI Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- **Compile**
- **JMP / JBP Setup**
- **Running**

IBM Software

JDBC and J2EE Evolution

- **Standard way to Query Database (relational)**
  - Structured Query Language (SQL) - 1992
- **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)
- **Standard API to Build Enterprise Applications**
  - Java 2 Enterprise Edition (J2EE)

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

IMS JDBC Obstacles

- **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 Java Virtual Machine (JVM) in IMS Dependent Region**
  - JMP (analogous to MPP)
  - JBP (analogous to non-message driven BMP)

Hierarchical to Relational Mapping

Hierarchical Design

Dealer Segment

53SJ9 | Mary | 111 Penny Lane
53SJ8 | Bob | 240 Elm St.
53SJ7 | George | 555 Bailey Ave.

JPR27 | Dodge | Durango
WJ45 | Mercury | Cougar

Model Segment

UU45 | Dodge | Viper
PR27 | Dodge | Durango
FF13 | Toyota | Camry

Equivalent 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

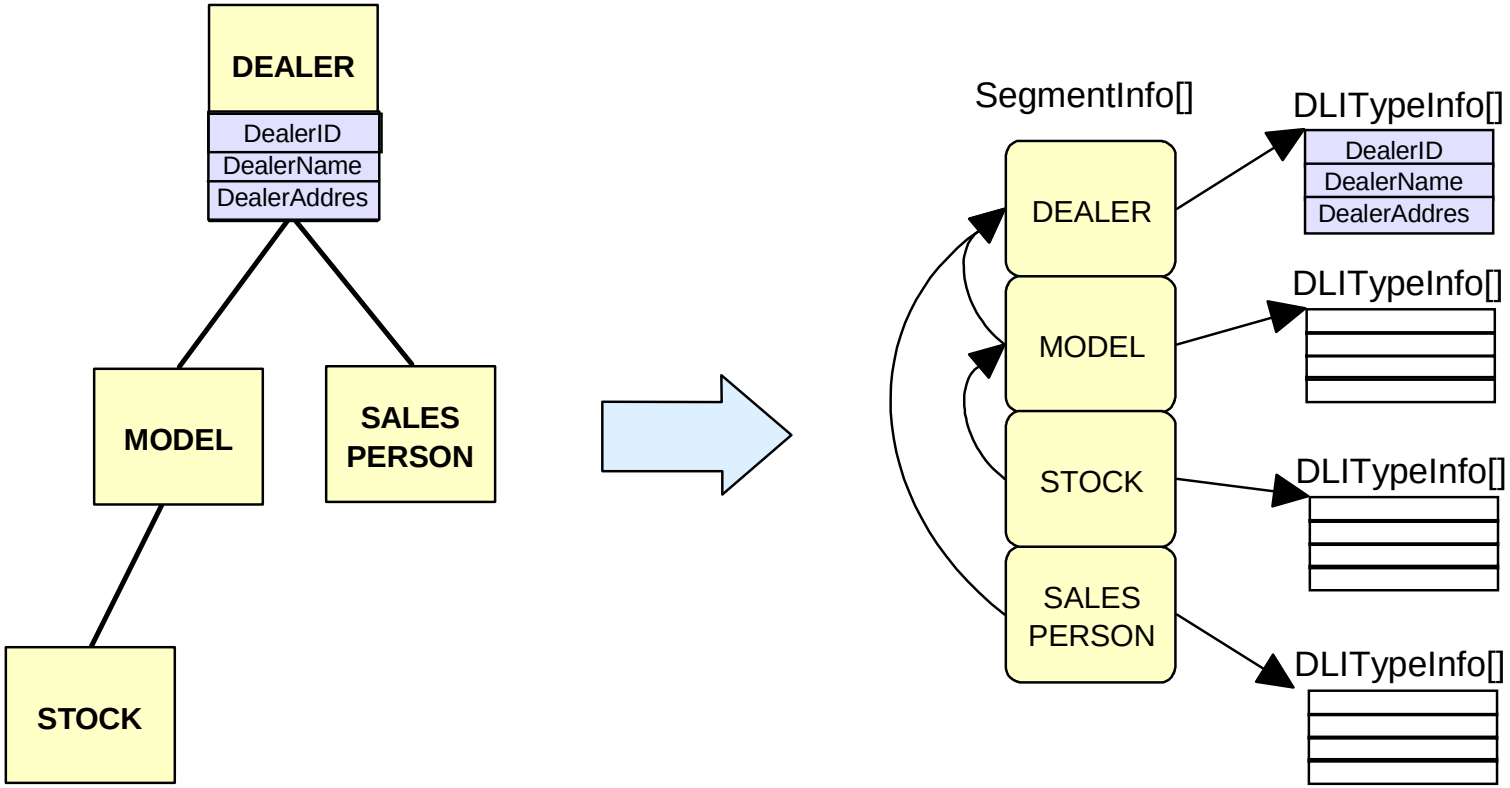
Model Table

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

IBM Software

IMS Metadata



DBDLIB, PSBLIB, COPYLIB

DLIDatabaseView

IBM Software

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

IBM Software

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

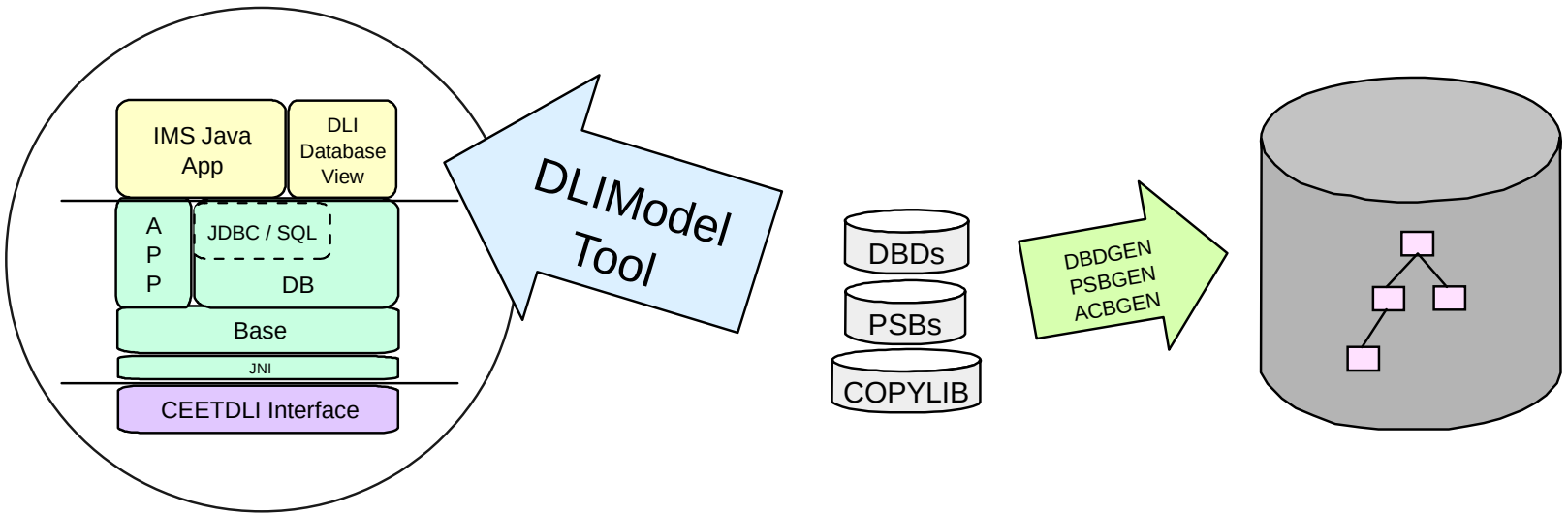
Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **DLIModel Utility**
- **Dealer Database Example**
  - Generating DLI Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- **Compile**
- **JMP / JBP Setup**
- **Running**

IBM Software

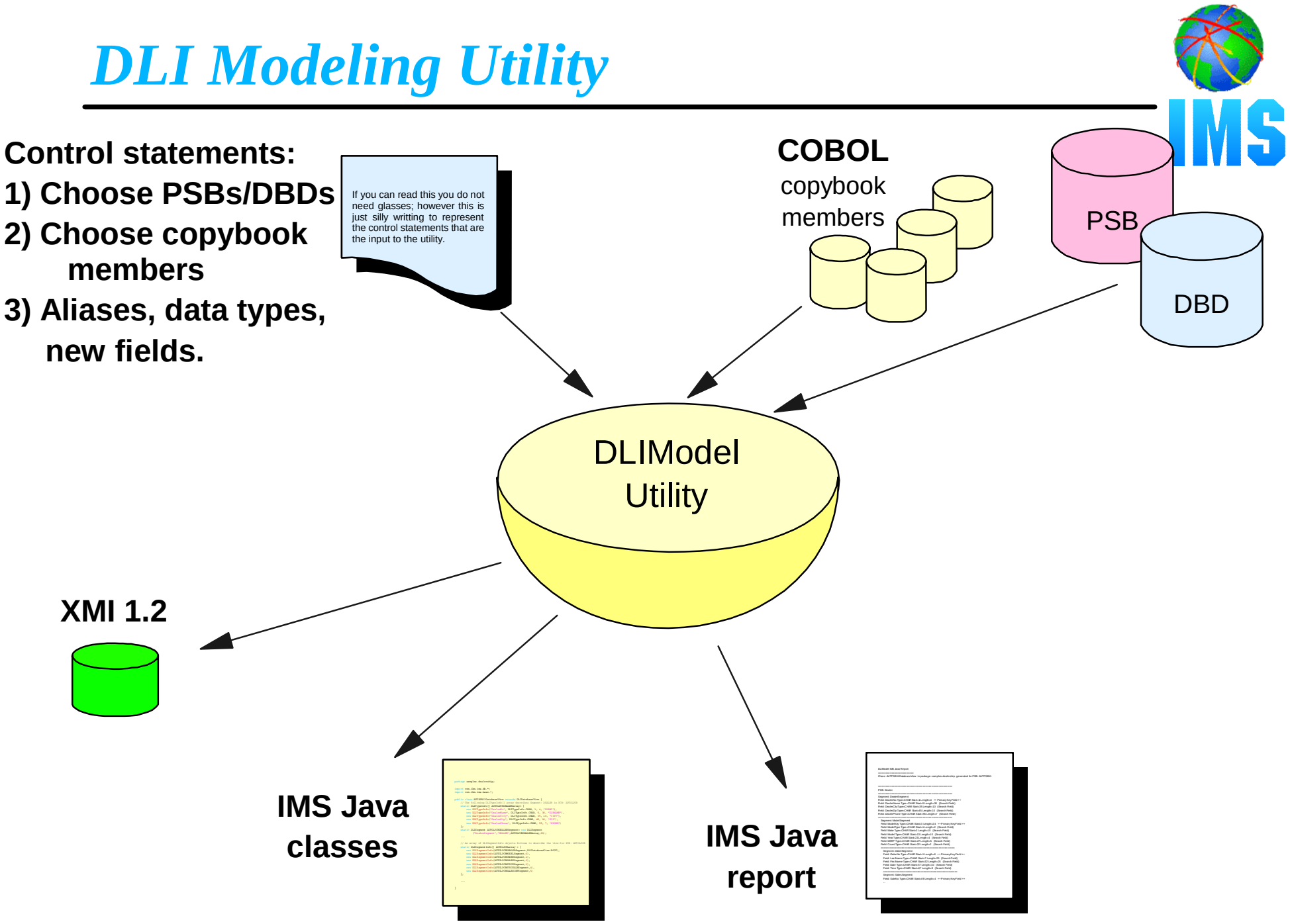
DLI Modeling Utility

- 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



IBM Software

DLI Modeling Utility

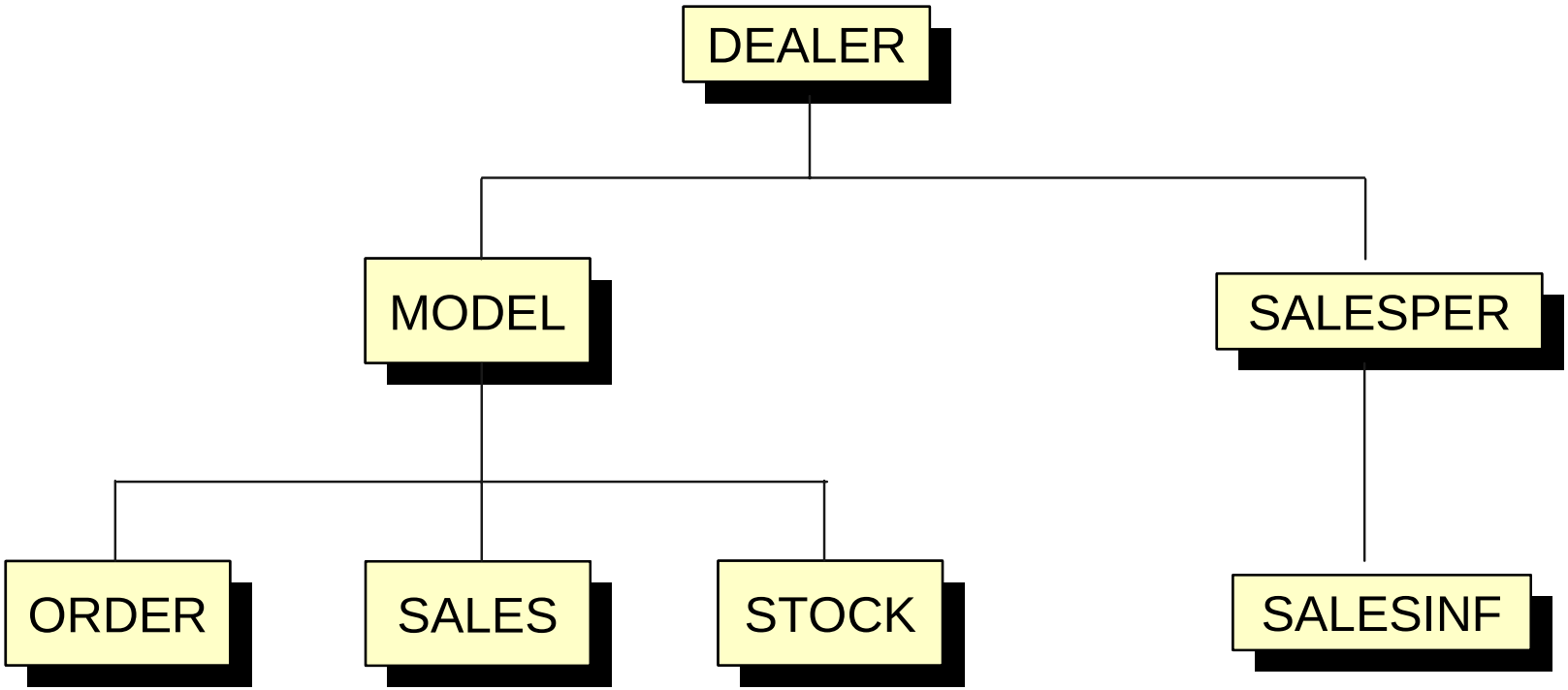


IBM Software

Overview

- **IMS Java**
  - What Is IMS Java
  - Why Use IMS Java
  - The IMS Java Class Library Architecture
- **JDBC and J2EE**
- **DLIModel Utility**
- **Dealer Database Example**
  - Generating DLI Metadata
  - JMP Application
  - Message Queue
  - SQL Query
- **Compile**
- **JMP / JBP Setup**
- **Running**

Dealership Sample System



IMS Java Dealership Sample

- **JMP**
- **Search for a Car currently in stock at a dealership**
- **Input**
  - CarMake
  - CarYear
- **Output**
  - Number of cars in stock
  - DealerName
  - CarMake
  - CarModel
  - CarYear
  - Lot

IBM Software

Application Development Steps

- **Create Control Statements**
- **Run DLIModel Utility**
  - DLIDatabaseView Metadata
  - IMS Java Report
- **Write Application**
- **Compile**
- **Execute**
- **Debug**
  - IMS Java XML Tracing

Control Statements


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

Control Statements


```
//*******************************
//   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


```
//*******************************
// Physical Segment Definitions
//*******************************
SEGM DBDName=AUTODB SegmentName=DEALER
  FIELD Name=DLRNO   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
  FIELD Name=MODKEY  JavaName=ModelKey
  FIELD Name=MODTYPE JavaName=ModelType
  FIELD Name=MAKE    JavaName=Make
  FIELD Name=MODEL   JavaName=Model
  FIELD Name=YEAR    JavaName=Year
  FIELD Name=MSRP    JavaName=MSRP
  FIELD Name=COUNT   JavaName=Count

  ...
```

Running the DLI Model Utility


```
//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
```

IBM Software

DLIDatabaseView (online catalog)


```
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)
  };
  ...
}
```

IBM Software

Java Report (programming guide)


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

  ================================================
  PCB: Dealer
  ================================================
  Segment: DealerSegment
  Field: DealerNo Type=CHAR 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: ModelSegment
    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=CHAR Start=23 Length=4   (Search Field)
    ================================================
        Segment: OrderSegment
        Field: OrderNo Type=CHAR Start=1 Length=6   ++ Primary Key Field ++
        ...
        Field: Time Type=CHAR Start=67 Length=8   (Search Field)
        ================================================
        Segment: SalesSegment
        Field: SaleNo Type=CHAR Start=49 Length=4   ++ Primary Key Field ++
        ...
```

IBM Software

IMS Java Application (JMP)


```
package samples.dealership;

public class IMSAuto extends IMSApplication {

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

    public void doBegin() {

      application logic here...

    }

}
```

IBM Software

Message Queue


```
public void doBegin() {
    IMSMessageQueue messageQueue = new IMSMessageQueue();

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

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

Define Input Message


```
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);
   }
}
```

Obtain a Connection


```
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

IBM Software

Execute Query


```
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...
}
```

recall: **Segment: DealerSegment**
Field: DealerName Type=CHAR Start=5 Length=30 (Search Field)

IBM Software

Process Results


```
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


```
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)

```
CLASSPATH= . : /usr/lpp/ims/imsjava71/imsjava.jar
```

- **Installed (in Path)**
  - JDK 1.3
  - PQ57329 (DLIModel Utility)
  - PQ60794 (Latest Level of IMS Java)

- **Compile**

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

IBM Software

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

- **STEPLIB**
  - location of DFSCLIB
    e.g. DQEIVP.ECDVL01.DLL
- **LIBPATH**
  - location of libJavTDLI.so
    e.g. /usr/lpp/ims/imajava71
- **Middleware Classpath**
  - path to IMS Java Java ARchive
    e.g. /usr/lpp/ims/imsjava71/imsjava.jar
- **Sharable Application Classpath**
  - location of application code
    e.g. /application/location

Execute

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

Enable Library Tracing

Establish Output Stream

```
IMSTrace.setOutputStream(System.err);
                or
FileWriter fileWriter =   newFileWriter("/tmp/PrizeDrawing.trace");
IMSTrace.setOutputWriter(fileWriter);
```

Set Trace Level

```
IMSTrace.libTraceLevel = IMSTrace.TRACE_DATA3;
```

Turn tracing on

```
IMSTrace.traceOn = true;
```

IMSTrace.libTraceLevel values

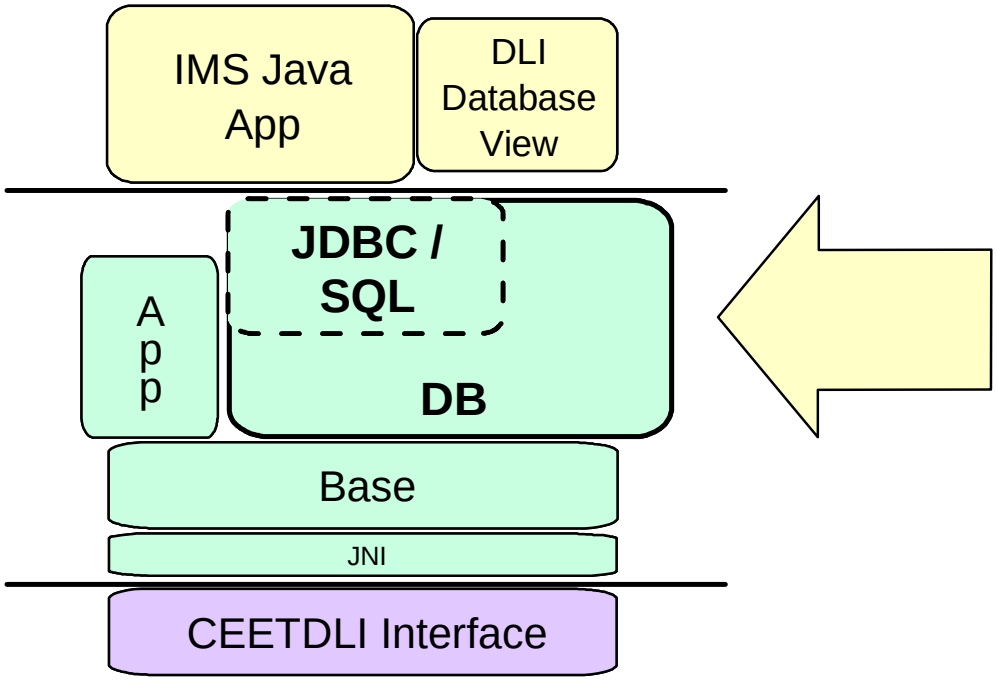
0	none
TRACE_EXCEPTIONS	least
TRACE_CTOR1	
TRACE_METHOD1	
TRACE_DATA1	
TRACE_CTOR2	
TRACE_METHOD2	
TRACE_DATA2	
TRACE_CTOR3	
TRACE_METHOD3	
TRACE_DATA3	most

Note: To ensure maximum tracing, add the trace enabling code to a static block.

Sample Trace Output


```
<?xml version='1.0'?>
<entry>main</entry>
  <entry>OrderStatusJDBC</entry>
  <exit>OrderStatusJDBC</exit>
  <entry>IMSApplication.begin()</entry>
    <entry>IMSApplication.initialize()</entry>
    <exit>IMSApplication.initialize()</exit>
    <entry>doBegin</entry>
      <entry>setup</entry>
        <entry>IMSMessageQueue()</entry>
        <exit>IMSMessageQueue()</exit>
       <entry>IMSFieldMessage(DLITypeInfo, int, boolean)</entry>
          <parm>
            <parmName>length</parmName>
            <parmChar>100</parmChar></parm>
          <parm>
            <parmName>isSPA</parmName>
            <parmChar>false</parmChar></parm>
        <exit>IMSFieldMessage(DLITypeInfo, int, boolean)</exit>
        <entry>IMSFieldMessage(DLITypeInfo, int, boolean)</entry>
          <parm>
            <parmName>length</parmName>
            <parmChar>580</parmChar></parm>
          <parm>
            <parmName>isSPA</parmName>
            <parmChar>false</parmChar></parm>
        <exit>IMSFieldMessage(DLITypeInfo, int, boolean)</exit>
...
```

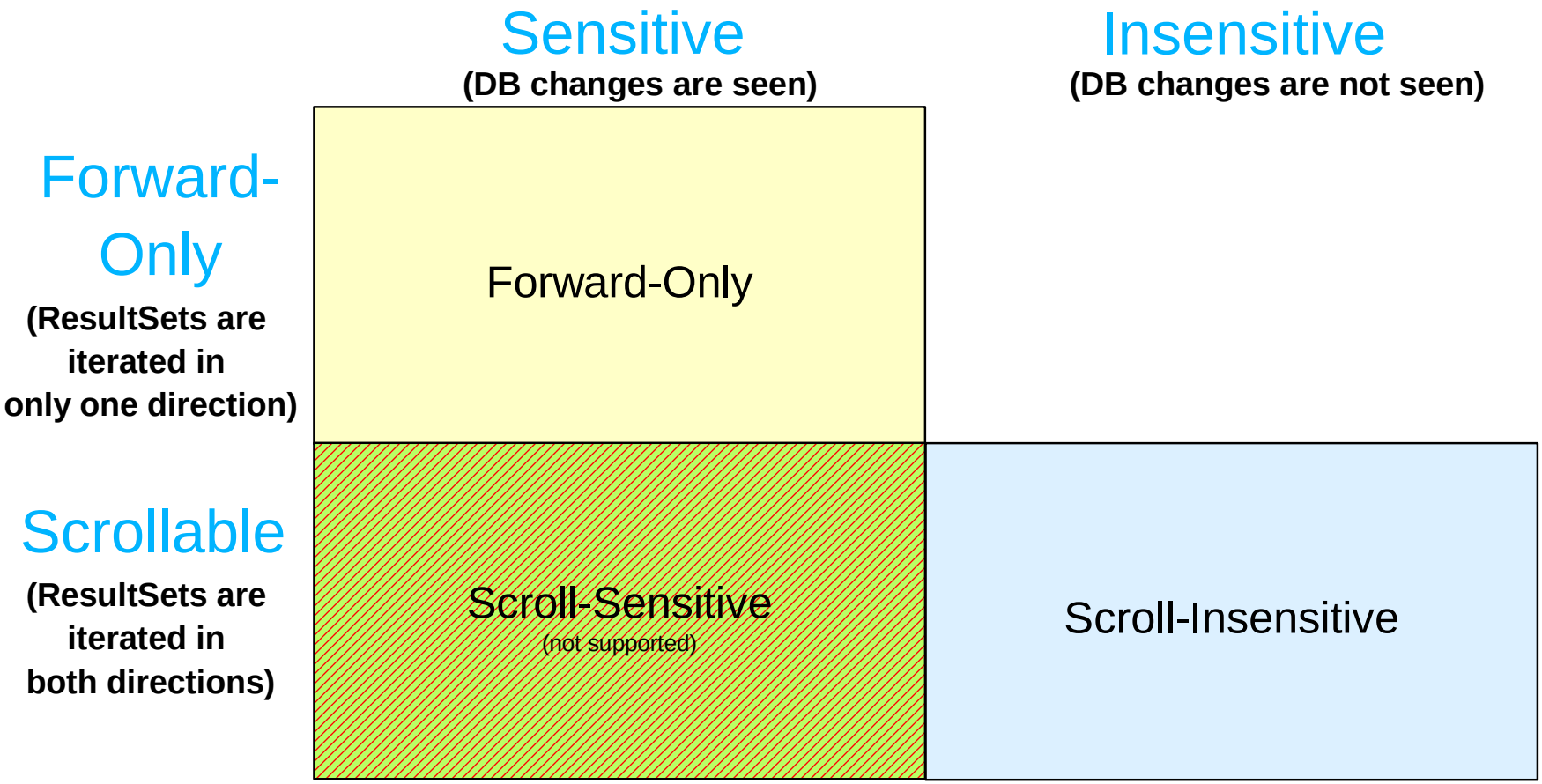
JDBC 2.0 Enhancements



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

IBM Software

IMS Java ResultSet Types



*IMS has no means to traverse a Query backwards

IMS Java ResultSet Types

- **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 ResultSet Concurrency

- **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

- **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.

IBM Software

New SQL keywords support (cont...)

- **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.