Using Java to Access Your CA IDMS™ Databases and Applications

Dave Ross
CA Technologies

IUA/CA IDMS™ Technical Conference   May 7-11, 2018

Abstract

- Many users are focused on developing new applications in Java, but still have a large investment in CA IDMS databases and applications. This session shows how you can use JDBC and SQL to leverage your CA IDMS databases and business logic from your Java applications. It includes an overview of Java programming concepts and JDBC for those new to Java, as well as more advanced topics for more experienced users.

Agenda

1	OOPS CONCEPTS
2	JAVA PROGRAMMING ELEMENTS
3	JDBC OVERVIEW
4	SAMPLE JDBC APPLICATION
5	ADVANCED JDBC FEATURES

OOPS Concepts

Object Oriented Programming
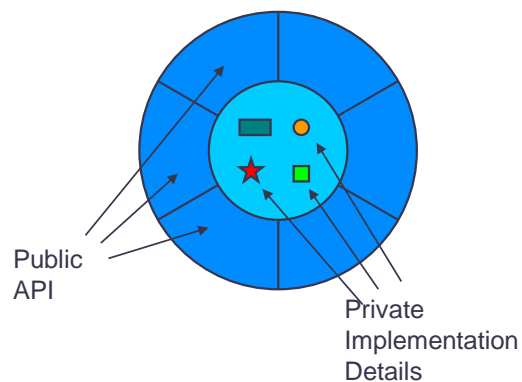
- Objects
- Messages
- Classes
- Inheritance

Object

- Software bundle
  - Variables
  - Methods
- State
- Behavior

Public API

Private Implementation Details

Message

- How objects interact
- Components
  - Object address
  - Method name
  - Parameters

Message
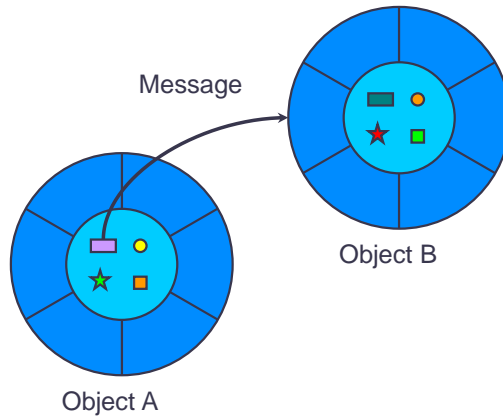
Object B

Object A

Class

- Object "blueprint"
- Members
  - Class
  - Instance

Public
API

Private
Implementation
Details
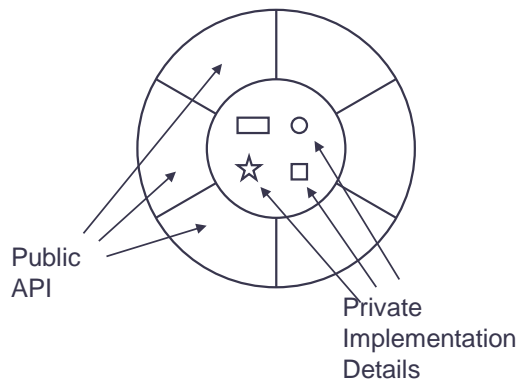
Inheritance

- Superclass
  - Common behavior
  - Reusable
- Subclass
  - Specialized behavior
  - Override methods
  - Hide variables

Superclass

Subclasses
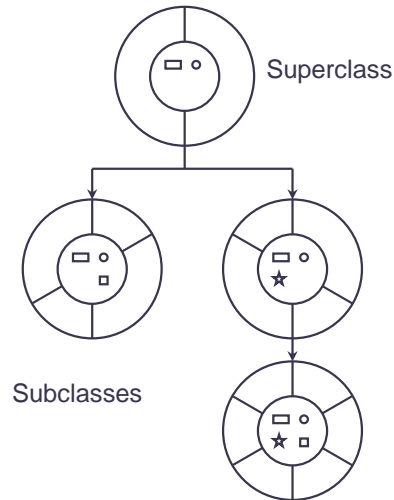
Java Programming Elements

Java

- Object oriented programming language
- Software platform
  - Java Virtual Machine
  - Java Application Programming Interface

Java Language Elements

- class
- interface
- package
- exception
- import

Class

- Basic programming unit
- Members
  - Variables
  - Methods

Interface

- More than a header

- Named "protocol"
  - Defines abstract methods
  - Declares constants

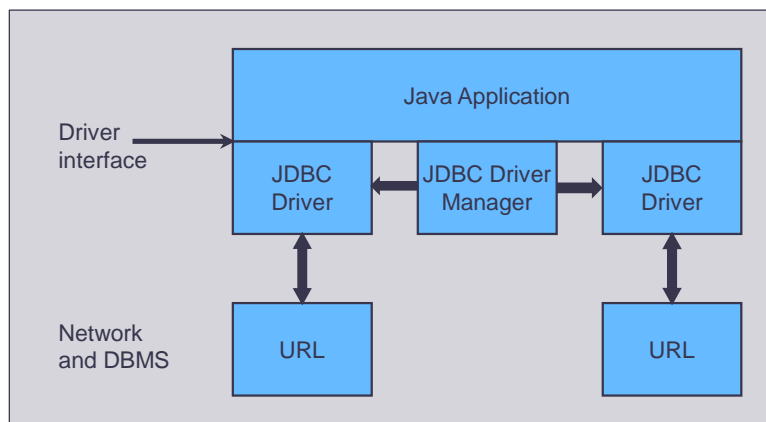- Classes implement interfaces

- Reference type

Package

- Like a library
- Related
  - Classes
  - Interfaces
  - Exceptions
- #import into Java program

JDBC Overview

JDBC Concepts

- "Java Data Base Connectivity"
- Call Level Interface
- Object Oriented
- Interoperability
- Any Java platform

JDBC Architecture

Java Application

Driver interface

JDBC Driver ↔ JDBC Driver Manager ↔ JDBC Driver

Network and DBMS

URL   URL

JDBC Objects

- Packages
  - java.sql
  - Javax.sql
- Interfaces
- Classes
- Exceptions

Commonly Used JDBC Interfaces

- Driver
- Connection
- Statement
- ResultSet
- PreparedStatment
- CallableStatement
- DatabaseMetaData
- ResultSetMetaData

JDBC Classes

- DriverManager
- DriverPropertyInfo
- Types
- Date
- Time
- Timestamp

JDBC Exceptions

- SQLException
- SQLWarning
- DataTruncation

JDBC Object Relationships

```
                    DriverManager

        Connection              Connection

   Statement    Statement       Statement

   ResultSet                    ResultSet
```
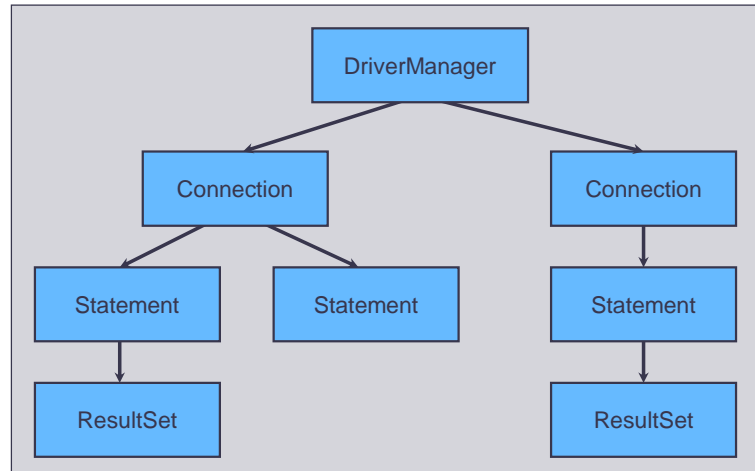
Sample JDBC Application

JDBC Query Application

- Simplified version of IdmsExample
  - Included with CA IDMS Server
  - Can execute any SQL statement
  - Java version of "BCF"
- Connect to a database
- Execute a query
- Retrieve results
- Handle errors

Housekeeping

```
import java.io.*;     // standard output
import java.util.*;   // properties
import java.sql.*;        // basic JDBC classes

class IuaExample
{
  public static void main (String[] args)
  {
    // register driver
    try
    {
      Class.forName("ca.idms.jdbc.IdmsJdbcDriver");
```

Get Connected

```java
// set database name and location
String url = "jdbc:idms://host:3709/dictname";

// set sign on information
Properties info = new Properties();
info.put("userid", "password");

// get a connection to the database
Connection conn =
  DriverManager.getConnection(url, info);
```

Here's the Meat

```java
// get a statement object to…
Statement stmt = conn.createStatement();

// execute the query and get a result set
ResultSet rs = stmt.executeQuery(
  "SELECT EMPFNAME, EMPLNAME FROM DEMO.EMPL");

// get and display the result set columns
while (rs.next())
  System.out.println(
    rs.getString(1) + " " + rs.getString(2));
```

The Punting Game

```java
      // disconnect
      conn.close();
    }
    catch (ClassNotFoundException e)
    {
      System.out.println("No driver: " + e);
    }
    catch (SQLException e)
    {
      System.out.println("SQL Error: " + e);
    }
  }
}
```

Advanced
JDBC Features

CA IDMS JDBC Supported Features

- Type 4 JDBC Driver
- Data Sources
- Procedures and Returned Result Sets
- Batch and Positioned Updates
- Enhanced Result Sets

Type 4 JDBC Driver

- Direct connection from JDBC Driver to CV
  - Uses TCP/IP line driver
  - CAICCI not needed
- JDBC Server in CV
  - Generic listener task
- Transparent to JDBC driver
  - Just identified by URL

Type 4 JDBC Driver

HTML	←	Apache	Advantage CA-IDMS
		JSP	
		JDBC Driver ↔ JDBC Server	
			SQL Client

Browser	Any Java	Any MF*
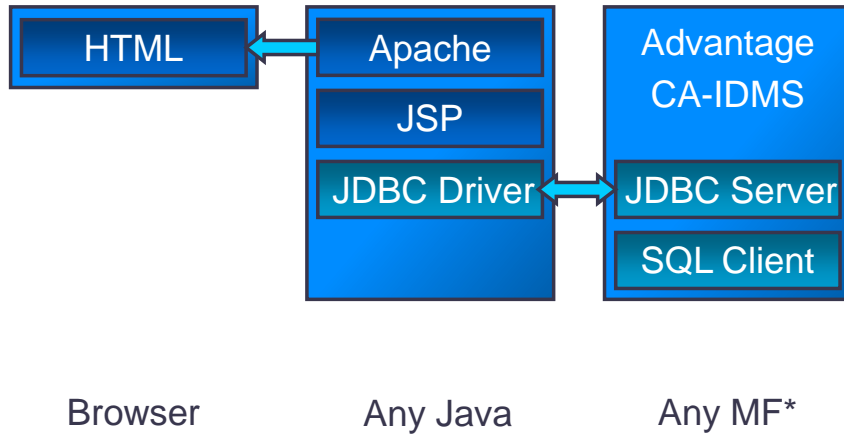
Data Sources

- Encapsulate all connection information

- Use with JNDI

- Java Bean conventions
  - properties
  - get/set methods
  - Serializable

IDMS DataSource Classes

- IdmsDataSource

- IdmsConnectionPoolDataSource

- IdmsXADataSource

- Typically implemented by application server
  - Except when using distributed transactions

Connection Pooling

Datasource API

logical Connection object

physical PooledConnection object

ConnectionPoolDatasource

XA Transactions

JDBC App

Datasource

A B

logical
Connection

Application Server

XAResource

pool A   pool B

A

XA
Tran
Mgr

B

XADatasource   A   B

physical
XAConnection

JDBC Driver   JDBC Driver
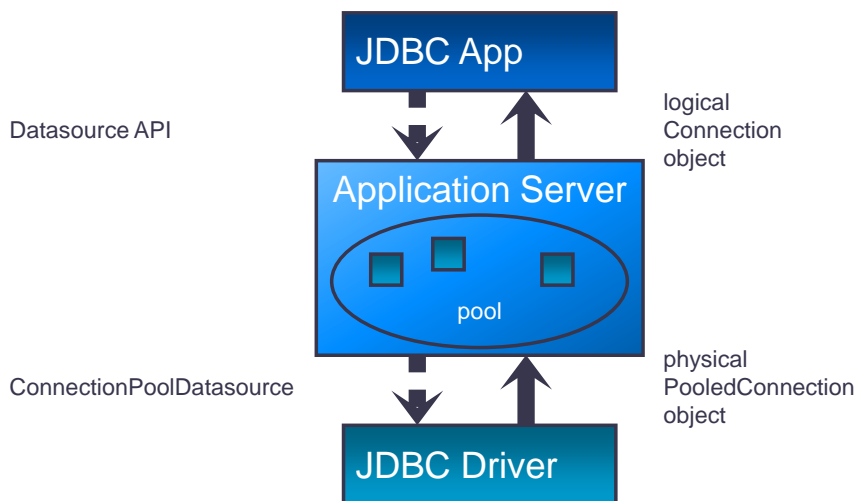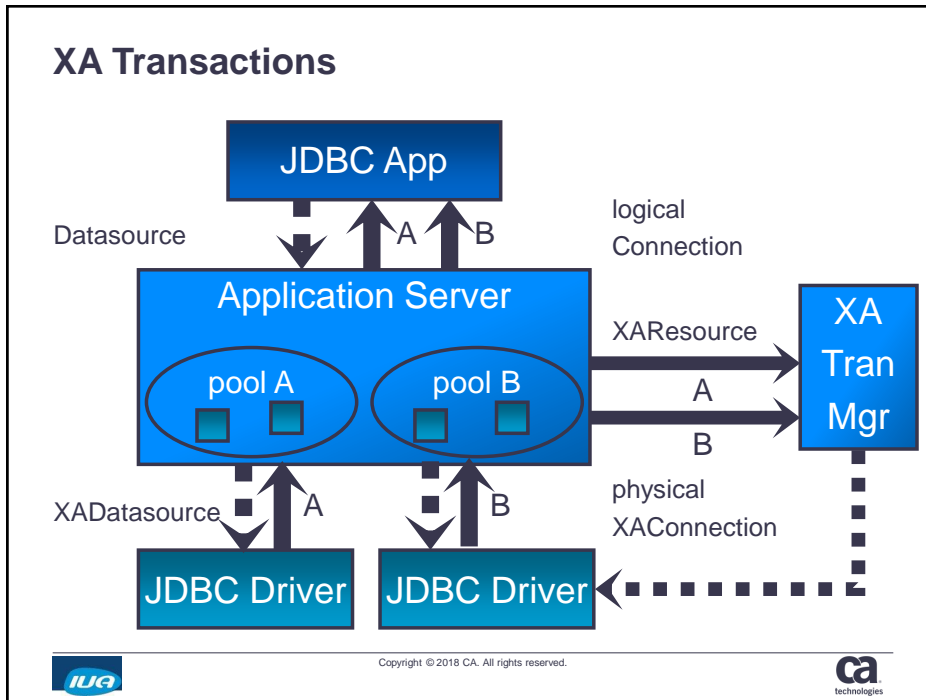
Called Procedures

- IdmsCallableStatement class
- Escape syntax
  - `{call procedurename(?)}`
- DatabaseMetaData methods
  - `getProcedures()`
  - `getProcedureColumns()`
- CallableStatement methods
  - Output parameters

JDBC named parameters

- Bind parameters to parameter markers using names instead of ordinal indexes:
  - Use with SQL CALL statement
  - Names as specified by CREATE PROCEDURE
- Simplifies use of procedures that have many parameters, particularly when defaults are acceptable
- Named parameters can be specified in any order
- Unused parameters can be omitted
- Supports both input and output parameters

JDBC Named Parameter Binding

```
CallableStatement cstmt =
 conn.prepareCall("CALL GETEMPLOYEE(?,?,?)");

cstmt.setString("EmpID", "B503-8907-15");
cstmt.registerOutParameter("Name",
Types.VARCHAR);
cstmt.registerOutParameter("Age",
Types.INTEGER);

cstmt.execute();

System.out.println("Employee: " +
 cstmt.getString("Name"))  + " is " +
 cstmt.getInt("Age") + " old.";
```

CA IDMS Returned Result Sets in Procedure

- Procedures can return result sets

```
CREATE PROCEDURE RSPROC
    (CHAR SOMEPARM, …)
DYNAMIC RESULT SETS 5
```

- Cursors left open after procedure exits are RETURNED

```
ALLOCATE CURSNAME CURSOR WITH RETURN
```

CA IDMS Returned Result Sets in Calling Program

- Calling program gets result sets
- Allocate RECEIVED cursor for result sets

```
CALL RSPROC(…)
ALLOCATE RCURNAME FOR PROCEDURE
    SPECIFIC PROCEDURE RSPROC
```

- Step through RETURNED result sets

```
Loop until SQLSTATE = '0100D'
    Loop until SQLSTATE = '02000'
        FETCH RCURNAME
    CLOSE RCURNAME
```

JDBC Multiple Result Sets

- JDBC driver can have multiple RECEIVED cursors
  - `KEEP_CURRENT_RESULT`
  - `CLOSE_CURRENT_RESULT`
  - `CLOSE_ALL_RESULTS`
- Allocate cursor and step through result sets

```
rc = statement.executeQuery("CALL RSPROC");
while (rc != false) {
    resultSet = statement.getResultSet();
    while (rc != false) {
        resultSet.fetch();
    }
    rc = statement.getMoreResults(CLOSE_CURRENT…);
}
```

Positioned Updates

- Statement methods:
  - `setCursorName("CURSOR1")`
  - `getCursorName()`
- SQL statements
  - `SELECT … FOR UPDATE`
  - `UPDATE WHERE CURRENT OF CURSOR1`
- Disables BULK FETCH

Batched Updates

- Statement methods
  - **addBatch()**
  - **executeBatch()**
  - **cancelBatch()**
- Required for J2EE compliance
- Driver caches most SQL commands
- Direct support for INSERT BULK

Enhanced Result Sets

- Scrollable
- Updateable
- Typical ResultSet methods:
  - **getRow()**
  - **updateRow()**
- Driver caches fetched rows
- Uses "optimistic concurrency" for updating

JDBC Result Sets

- ResultSet type attribute
  - **TYPE_FORWARD_ONLY**
  - **TYPE_SCROLL_INSENSITIVE**
  - **TYPE_SCROLL_SENSITIVE**
- ResultSet concurrency attribute
  - **CONCUR_READ_ONLY**
  - **CONCUR_UPDATABLE**
- ResultSet methods
  - Position cursor
  - Update row

CA IDMS Scrollable Result Sets

- Caches rows as fetched from CA IDMS
- ResutSet type attribute
  - **TYPE_FORWARD_ONLY**
  - **TYPE_SCROLL_INSENSITIVE**
- ResultSet concurrency attribute
  - **CONCUR_READ_ONLY**
- ResultSet methods
  - Position cursor

JDBC Row Sets

- Extend ResultSet interface
- Reference Implementation
  - CachedRowSet
  - Included in JRE
- Extends CA IDMS result set implementation
  - **TYPE_SCROLL_SENSITIVE**
  - **CONCUR_UPDATABLE**
  - Update methods

Summary

- OOPS Concepts
- Java Programming Elements
- JDBC Overview
- Sample JDBC Application
- Advanced JDBC Features

Questions & Answers

Please Complete a Session Evaluation Form

- The number for this session is **A03**

- After completing your session evaluation form, place it in the envelope at the front of the room