CSC584 Enterprise Programming

Chapter 5 – Java Database connectivity

MARSHIMA MOHD ROSLI

COMPUTER SCIENCE

Chapter 5 Outline

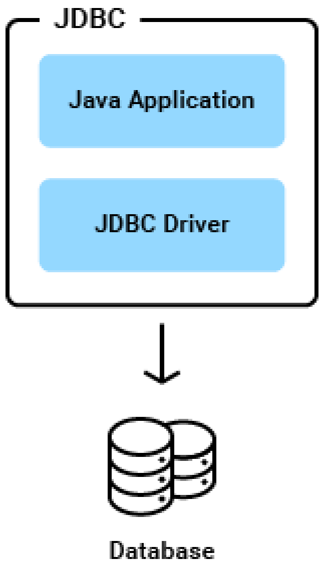
Java Database connectivity

- ❑ Overview of java database programming
- ❑Define JDBC API
- ❑Describe various types of JDBC
- ❑Identify JDBC product
- ❑Describe the 2 tier server client model
- ❑Setup JDBC connection to a database with JSP and Servlet
- ❑Create and Execute SQL statement
- ❑Describe ResultSet Object

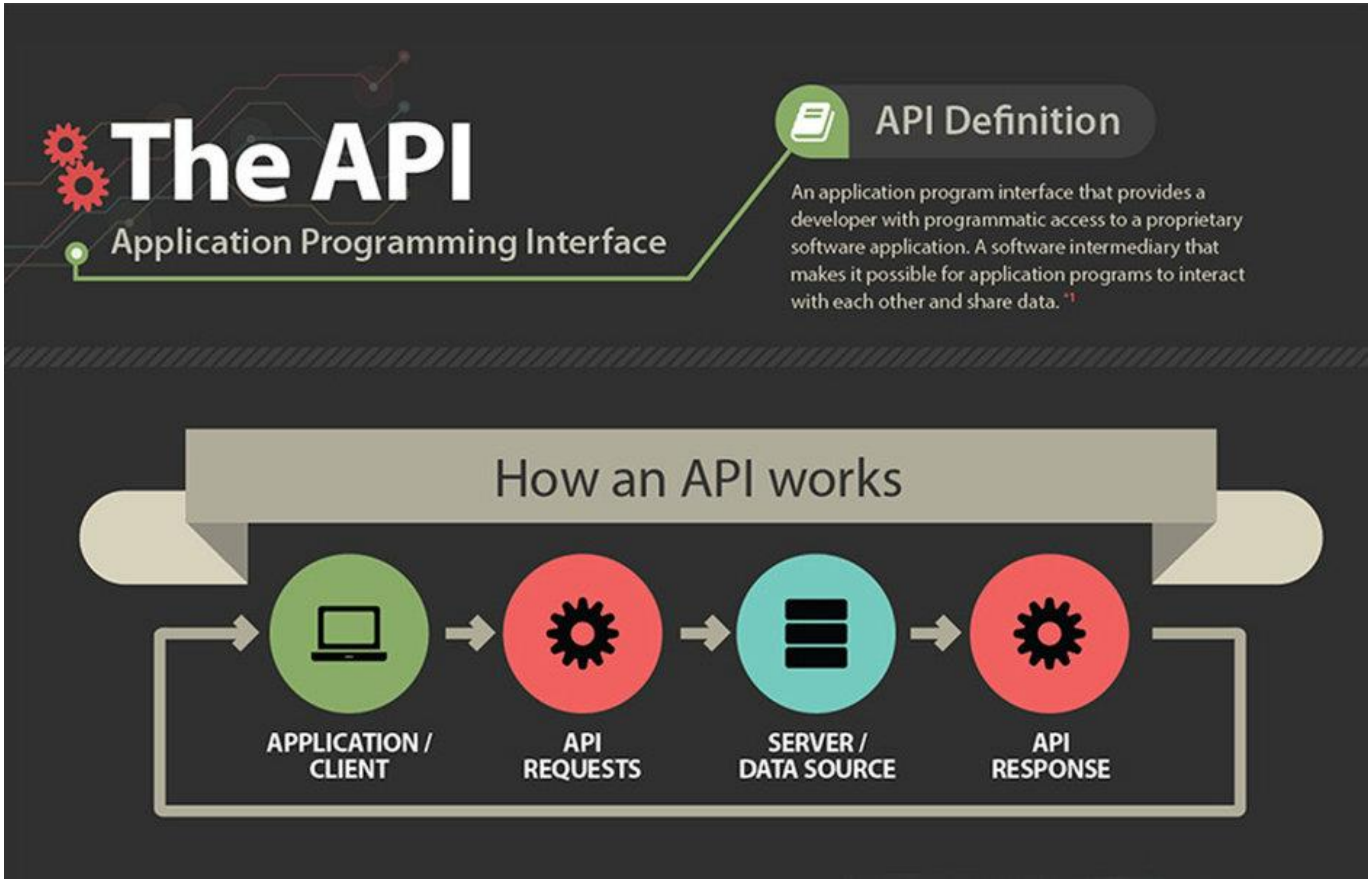
What is JDBC?

"An API that lets you access virtually any tabular data source from the Java programming language"

- What's an API?
- What's a tabular data source?

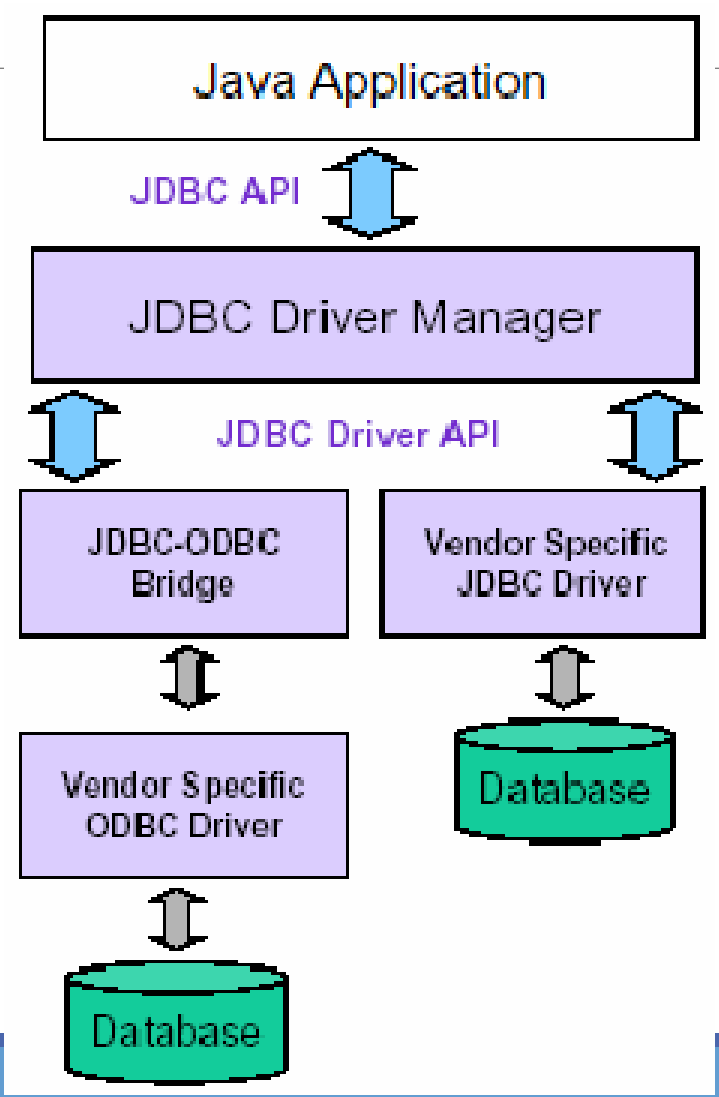


"… access virtually any data source, from relational databases to spreadsheets and flat files."



An application program interface that provides a developer with programmatic access to a proprietary software application. A software intermediary that makes it possible for application programs to interact with each other and share data. [1]

General Architecture



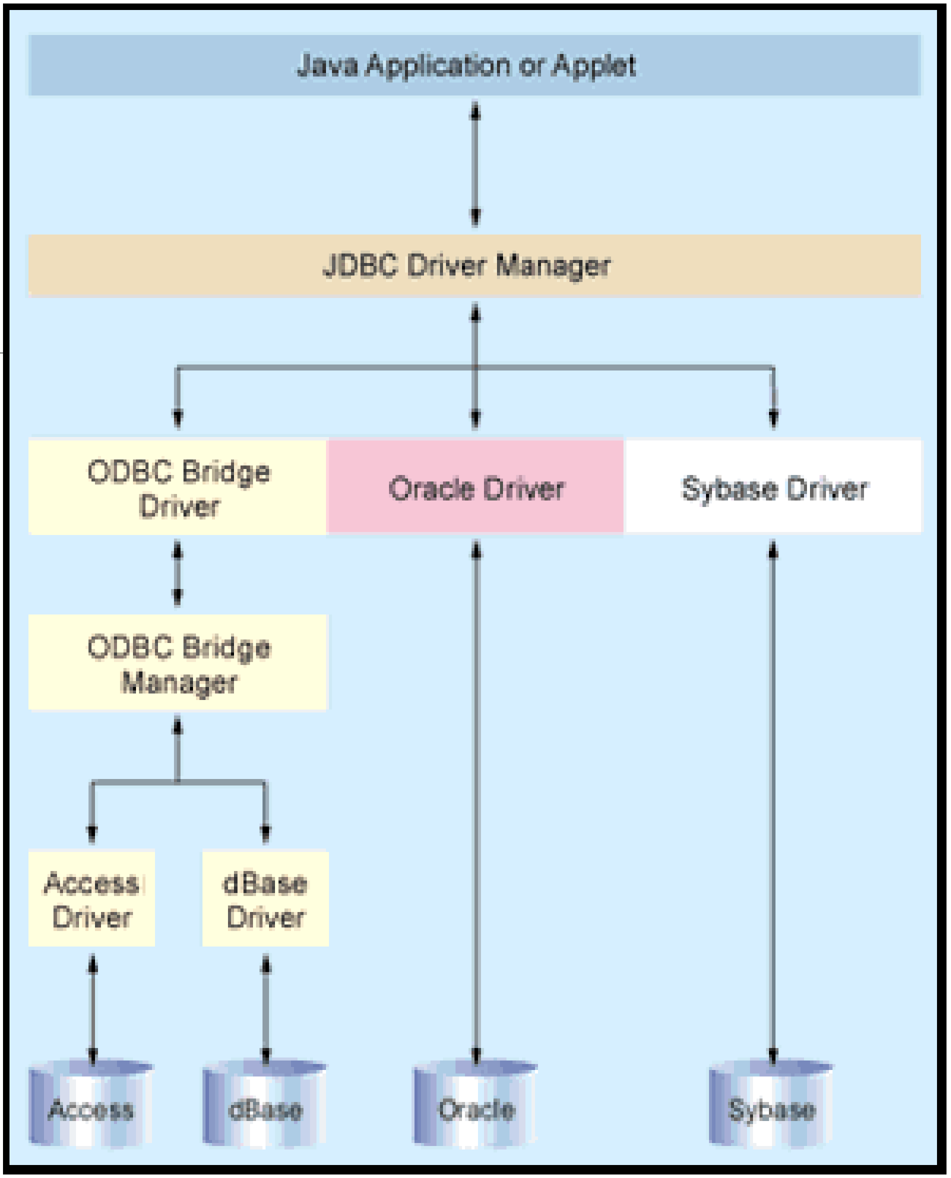


Figure 1. Anatomy of Data Access. The Driver Manager provides a consistent layer between your Java app and back-end database. JDBC works natively (such as with the Oracle driver in this example) or with any ODBC datasource.

SQL

Structured Query Language, pronounced S-Q-L, or Sequel

- To access or write applications for database systems, you need to use the Structured Query Language (SQL).
- SQL is the universal language for accessing relational database systems.
- Application programs may allow users to access database without directly using SQL, but these applications themselves must use SQL to access the database.

Examples of simple SQL statements

- Create table
- Drop table
- Describe table
- Select
- Insert
- Delete
- Update

```
create table Course (
  courseId char(5),
  subjectId char(4) not null,
  courseNumber integer,
  title varchar(50) not null,
  numOfCredits integer,
  primary key (courseId)
);
```

```
create table Student (
  ssn char(9),
  firstName varchar(25),
  mi char(1),
  lastName varchar(25),
  birthDate date,
  street varchar(25),
  phone char(11),
  zipCode char(5),
  deptId char(4),
  primary key (ssn)
);
```

Examples of simple SQL statements

Create table

Drop table

Describe table

Select

Insert

Delete

Update

```
drop table Enrollment;
drop table Course;
drop table Student;
```

Examples of simple SQL statements

- Create table
- Drop table
- Describe table
- Select
- Insert
- Delete
- Update

```
describe Course;  -- Oracle
```

UNIVERSITI TEKNOLOGI MARA

Examples of simple SQL statements

Create table

Drop table

Describe table

Select

Insert

Delete

Update

```
select firstName, mi, lastName
from Student
where deptId = 'CS';

select firstName, mi, lastName
from Student
where deptId = 'CS' and zipCode = '31411';

select *
from Student
where deptId = 'CS' and zipCode = '31411';
```

Examples of simple SQL statements

- Create table
- Drop table
- Describe table
- Select
- Insert
- Delete
- Update

```
insert into Course (courseId, subjectId, courseNumber, title)
values ('11113', 'CSCI', '3720', 'Database Systems', 3);
```

Examples of simple SQL statements

- Create table
- Drop table
- Describe table
- Select
- Insert
- Update
- Delete

```
update Course
set numOfCredits = 4
where title = 'Database Systems';
```

Examples of simple SQL statements

Create table

Drop table

Describe table

Select

Insert

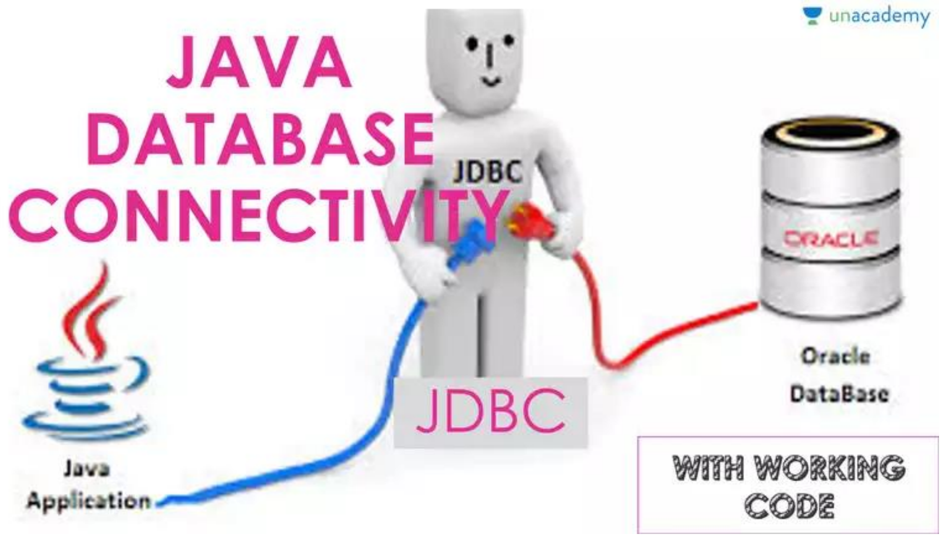
Update

Delete

```
delete Course
where title = 'Database System';
```

Basic steps to use a database in Java

- ✓1.Establish a **connection**
- ✓2.Create JDBC **Statements**
- ✓3.Execute **SQL** Statements
- ✓4.GET **ResultSet**
- ✓5.**Close** connections



1. Establish a connection

import java.sql.*;

Load the vendor specific driver

- Class.forName("oracle.jdbc.driver.OracleDriver");
  - What do you think this statement does, and how?
  - Dynamically loads a driver class, for Oracle database

Make the connection

- Connection con = DriverManager.getConnection("jdbc:oracle:thin:@oracle-prod:1521:OPROD", username, passwd);
  - What do you think this statement does?
  - Establishes connection to database by obtaining a *Connection* object

2. Create JDBC statement(s)

```
Statement stmt = con.createStatement() ;
```

Creates a Statement object for sending SQL statements to the database

Executing SQL Statements

```
String createLehigh = "Create table Lehigh " +
"(SSN Integer not null, Name VARCHAR(32), " +
"Marks Integer)";
stmt.executeUpdate(createLehigh);
//What does this statement do?


String insertLehigh = "Insert into Lehigh values" +
        "(123456789,abc,100)";
stmt.executeUpdate(insertLehigh);
```

Get ResultSet

```
String queryLehigh = "select * from Lehigh";

ResultSet rs = Stmt.executeQuery(queryLehigh);
//What does this statement do?

while (rs.next()) {
 int ssn = rs.getInt("SSN");
 String name = rs.getString("NAME");
 int marks = rs.getInt("MARKS");
}
```

Close connection

```
stmt.close();

con.close();
```

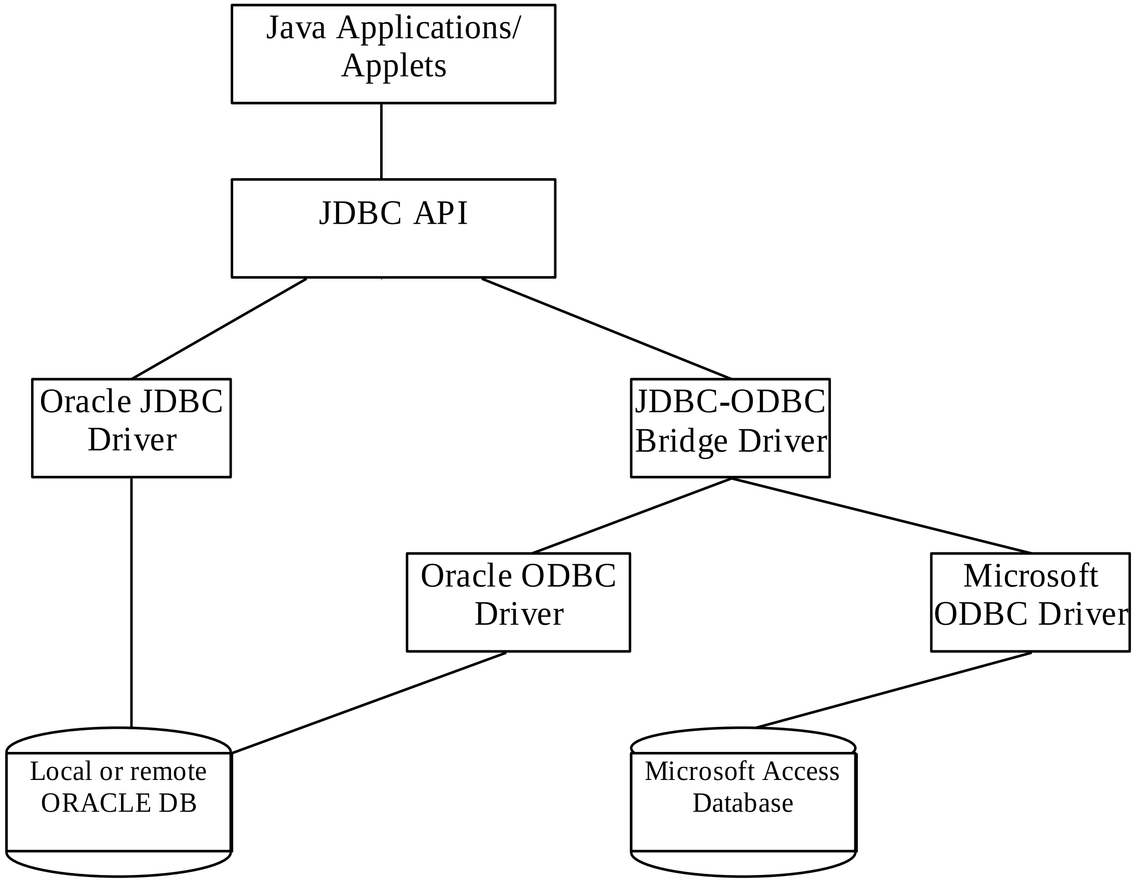
Why Java for Database Programming?

- First, Java is platform independent. You can develop platform-independent database applications using SQL and Java for any relational database systems.
- Second, the support for accessing database systems from Java is built into Java API, so you can create database applications using all Java code with a common interface.
- Third, Java is taught in almost every university either as the first programming language or as the second programming language.

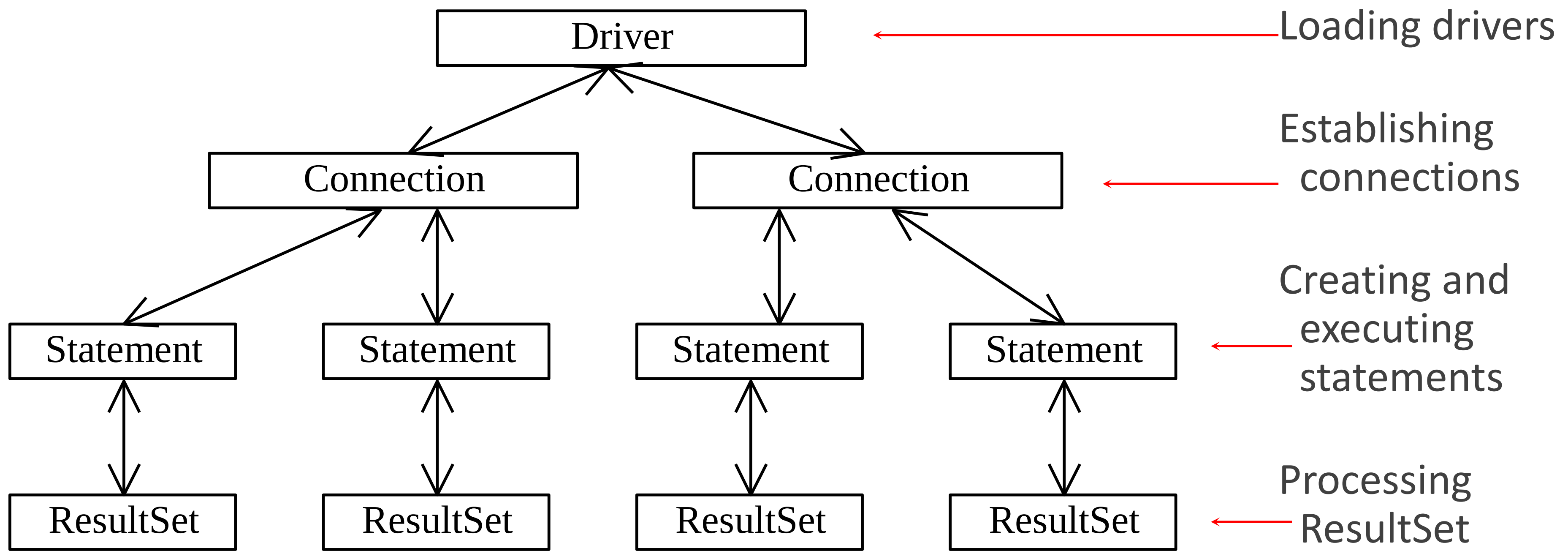
Database Applications Using Java

- ✓GUI
- ✓Client/Server
- ✓Server-Side programming

The Architecture of JDBC



The JDBC Interfaces



Developing JDBC Programs

Loading drivers

Establishing connections

Creating and executing statements

Processing ResultSet

Statement to load a driver:

```
Class.forName("JDBCDriverClass");
```

A driver is a class. For example:

Database	Driver Class	Source
Access	sun.jdbc.odbc.JdbcOdbcDriver	Already in JDK
MySQL	com.mysql.jdbc.Driver	Website
Oracle	oracle.jdbc.driver.OracleDriver	Website

The JDBC-ODBC driver for Access is bundled in JDK.

MySQL driver class is in mysqljdbc.jar

Oracle driver class is in classes12.jar

To use the MySQL and Oracle drivers, you have to add mysqljdbc.jar and classes12.jar in the classpath using the following DOS command on Windows:

```
classpath=%classpath%;c:\book\mysqljdbc.jar;c:\book\classes12.jar
```

Developing JDBC Programs

Loading drivers

Establishing connections

Creating and executing statements

Processing ResultSet

```
Connection connection = DriverManager.getConnection(databaseURL);
```

Database	URL Pattern
Access	jdbc:odbc:dataSource
MySQL	jdbc:mysql://hostname/dbname
Oracle	jdbc:oracle:thin:@hostname:port#:oracleDBSID

Examples:

For Access:

```
Connection connection = DriverManager.getConnection
  ("jdbc:odbc:ExampleMDBDataSource");
```

See Supplement IV.D for creating an ODBC data source

For MySQL:

```
Connection connection = DriverManager.getConnection
  ("jdbc:mysql://localhost/test");
```

-

For Oracle:

```
Connection connection = DriverManager.getConnection
  ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl",  "scott", "tiger");
```

Developing JDBC Programs

Loading drivers

Establishing connections

Creating and executing statements

Processing ResultSet

Creating statement:

```
        Statement statement = connection.createStatement();
```

Executing statement (for update, delete, insert):

```
        statement.executeUpdate
           ("create table Temp (col1 char(5), col2 char(5))");
```

Executing statement (for select):

```
       // Select the columns from the Student table
       ResultSet resultSet = statement.executeQuery
         ("select firstName, mi, lastName from Student where lastName "
           + " = 'Smith'");
```

Developing JDBC Programs

Loading drivers

Establishing connections

Creating and executing statements

Processing ResultSet

Executing statement (for select):

```
        // Select the columns from the Student table
        ResultSet resultSet = stmt.executeQuery
          ("select firstName, mi, lastName from Student where lastName "
            + " = 'Smith'");
```

Processing ResultSet (for select):

```
        // Iterate through the result and print the student names
        while (resultSet.next())
          System.out.println(resultSet.getString(1) + " " + resultSet.getString(2)
            + ". " + resultSet.getString(3));
```

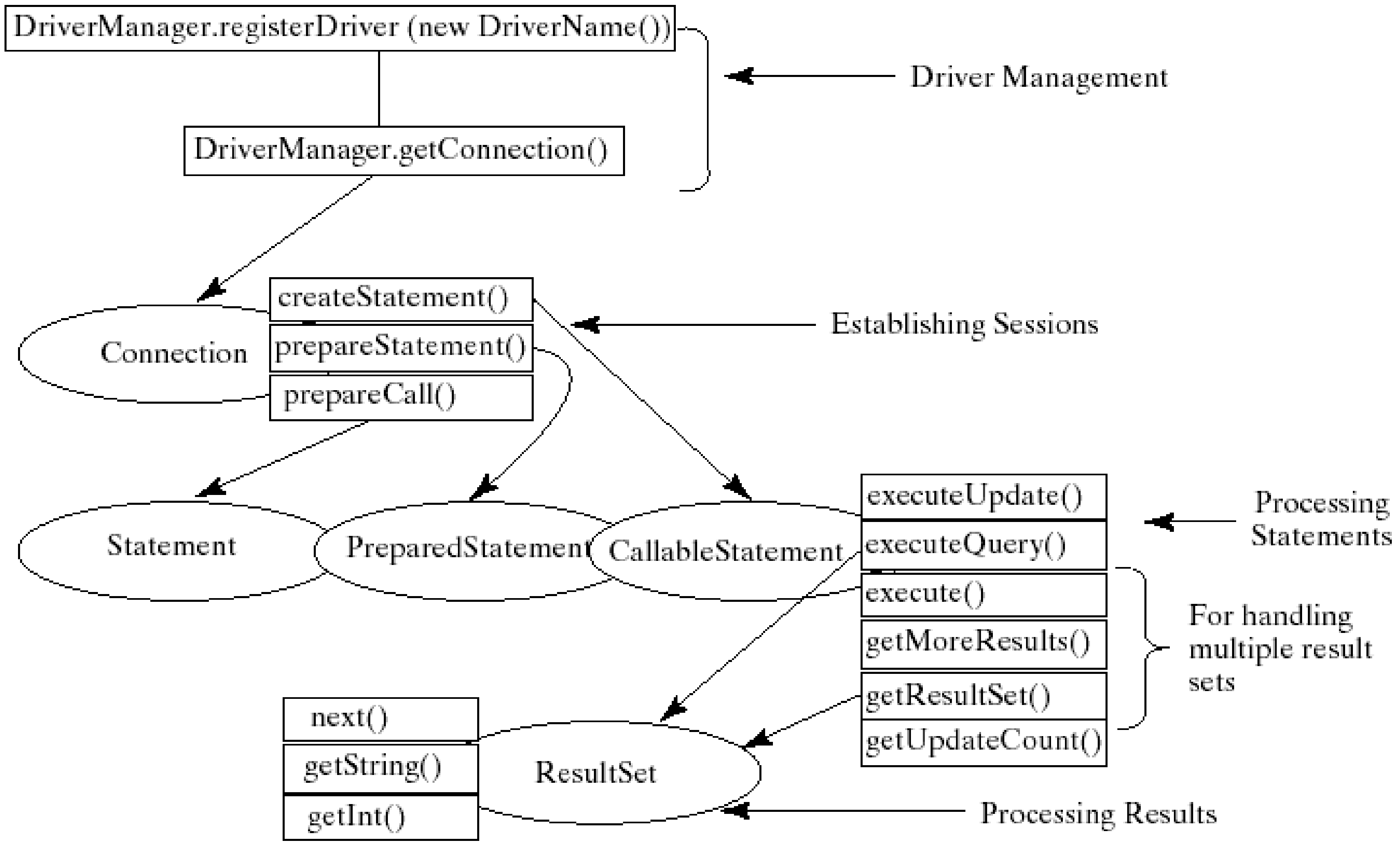
```
import java.sql.*;
public class SimpleJdbc {
  public static void main(String[] args)
      throws SQLException, ClassNotFoundException {
    // Load the JDBC driver
    Class.forName("com.mysql.jdbc.Driver");
    System.out.println("Driver loaded");

    // Establish a connection
    Connection connection = DriverManager.getConnection
      ("jdbc:mysql://localhost/test");
    System.out.println("Database connected");

    // Create a statement
    Statement statement = connection.createStatement();

    // Execute a statement
    ResultSet resultSet = statement.executeQuery
      ("select firstName, mi, lastName from Student where lastName
   "
        + " = 'Smith'");

    // Iterate through the result and print the student names
    while (resultSet.next())
      System.out.println(resultSet.getString(1) + "\t" +
        resultSet.getString(2) + "\t" + resultSet.getString(3));

    // Close the connection
    connection.close();
  }
}
```

Processing Statements

- ✓Once a connection to a particular database is established, it can be used to send SQL statements from your program to the database.

- ✓JDBC provides the Statement, PreparedStatement, and CallableStatement interfaces to facilitate sending statements to a database for execution and receiving execution results from the database.

Processing Statements Diagram



The execute, executeQuery, and executeUpdate Methods

- The methods for executing SQL statements are execute, executeQuery, and executeUpdate, each of which accepts a string containing a SQL statement as an argument.

- This string is passed to the database for execution. The execute method should be used if the execution produces multiple result sets, multiple update counts, or a combination of result sets and update counts.

The execute, executeQuery, and executeUpdate Methods, cont.

- ✓The executeQuery method should be used if the execution produces a single result set, such as the SQL select statement.
- ✓The executeUpdate method should be used if the statement results in a single update count or no update count, such as a SQL INSERT, DELETE, UPDATE, or DDL statement.

PreparedStatement

✓The PreparedStatement interface is designed to execute dynamic SQL statements and SQL-stored procedures with IN parameters. These SQL statements and stored procedures are precompiled for efficient use when repeatedly executed.

```
Statement pstmt = connection.prepareStatement

 ("insert into Student (firstName, mi, lastName) +

values (?, ?, ?)");
```

Handling Errors with Exceptions

- ✓Programs should recover and leave the database in a consistent state.
- ✓If a statement in the try block throws an exception or warning, it can be caught in one of the corresponding catch statements
- ✓How might a finally {…} block be helpful here?
- ✓E.g., you could rollback your transaction in a catch { …} block or close database connection and free database related resources in finally {…} block

Sample program

```
import java.sql.*;
class Test  {
  public static void main(String[] args)  {
    try {
      Class.forName("sun.jdbc.odbc.JdbcOdbcDriver"); //dynamic loading of driver
      String filename = "c:/db1.mdb"; //Location of an Access database
      String database = "jdbc:odbc:Driver={Microsoft Access Driver (*.mdb)};DBQ=";
      database+= filename.trim() + ";DriverID=22;READONLY=true}"; //add on to end
      Connection con = DriverManager.getConnection( database ,"","");
      Statement s = con.createStatement();
      s.execute("create table TEST12345 ( firstcolumn integer )");
      s.execute("insert into TEST12345 values(1)");
      s.execute("select firstcolumn from TEST12345");
```

Sample program(cont)

```
    ResultSet rs = s.getResultSet();
    if (rs != null) // if rs == null, then there is no ResultSet to view
    while ( rs.next() ) // this will step through our data row-by-row
    {   /* the next line will get the first column in our current row's ResultSet
       as a String ( getString( columnNumber) ) and output it to the screen */
       System.out.println("Data from column_name: " + rs.getString(1) );
    }
    s.close(); // close Statement to let the database know we're done with it
    con.close(); //close connection
  }
 catch (Exception err) { System.out.println("ERROR: " + err);  }
}
}
```

UNIVERSITI TEKNOLOGI MARA

Activity 1

JDBC exercise

- Fill in the blanks

Example: JDBC with JSP

JSP Syntax

Comment

- `<%-- Comment --%>`

Expression

- `<%= java expression %>`

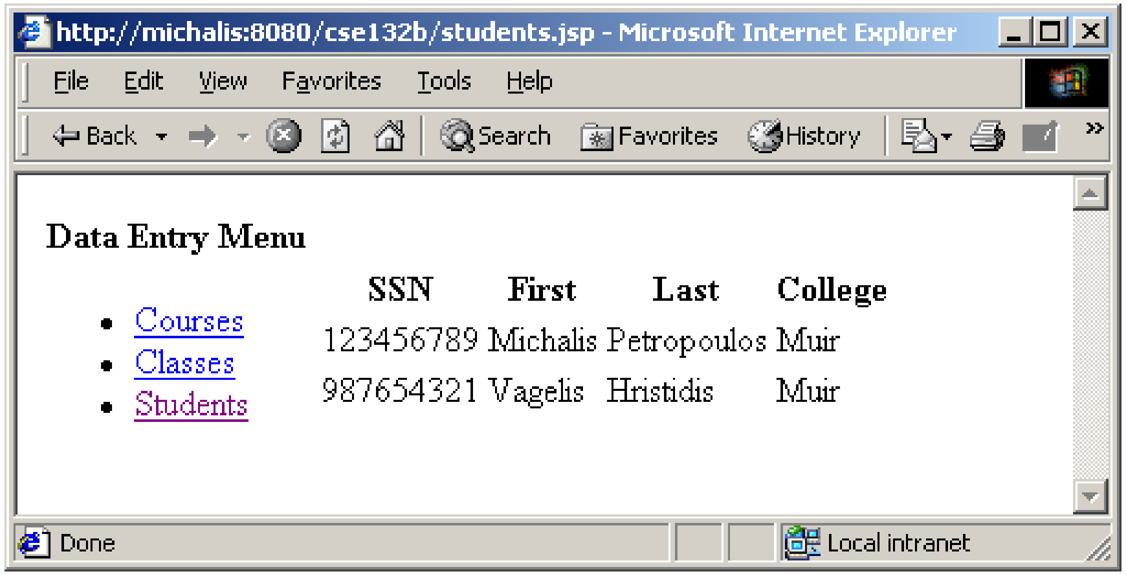
Scriplet

- `<% java code fragment %>`

Include

- `<jsp:include page="relativeURL" />`

Entry Form - First Attempt


http://michalis:8080/cse132b/students.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
Local intranet

Data Entry Menu

- Courses
- Classes
- Students

SSN	First	Last	College
123456789	Michalis	Petropoulos	Muir
987654321	Vagelis	Hristidis	Muir

Entry Form - First Attempt

Menu HTML Code

```
<b>Data Entry Menu</b>
<ul>
    <li>
        <a href="courses.jsp">Courses<a>
    </li>
    <li>
        <a href="classes.jsp">Classes<a>
    </li>
    <li>
        <a href="students.jsp">Students<a>
    </li>
</ul>
```

Entry Form - First Attempt

JSP Code

```
<html>
<body>
    <table>
        <tr>
            <td>
                <jsp:include page="menu.html" />
            </td>
            <td>
                Open connection code
                Statement code
                Presentation code
                Close connection code
            </td>
        </tr>
    </table>
</body>
</html>
```

Entry Form - First Attempt

Open Connectivity Code

```
<%-- Set the scripting language to java and --%>
<%-- import the java.sql package --%>
<%@ page language="java" import="java.sql.*" %>

<%
  try {
    // Load Oracle Driver class file
    DriverManager.registerDriver
    (new oracle.jdbc.driver.OracleDriver());

    // Make a connection to the Oracle datasource
    Connection conn = DriverManager.getConnection
    ("jdbc:oracle:thin:@feast.ucsd.edu:1521:source",
     "user", "pass");
%>
```

Entry Form - First Attempt

Statement Code

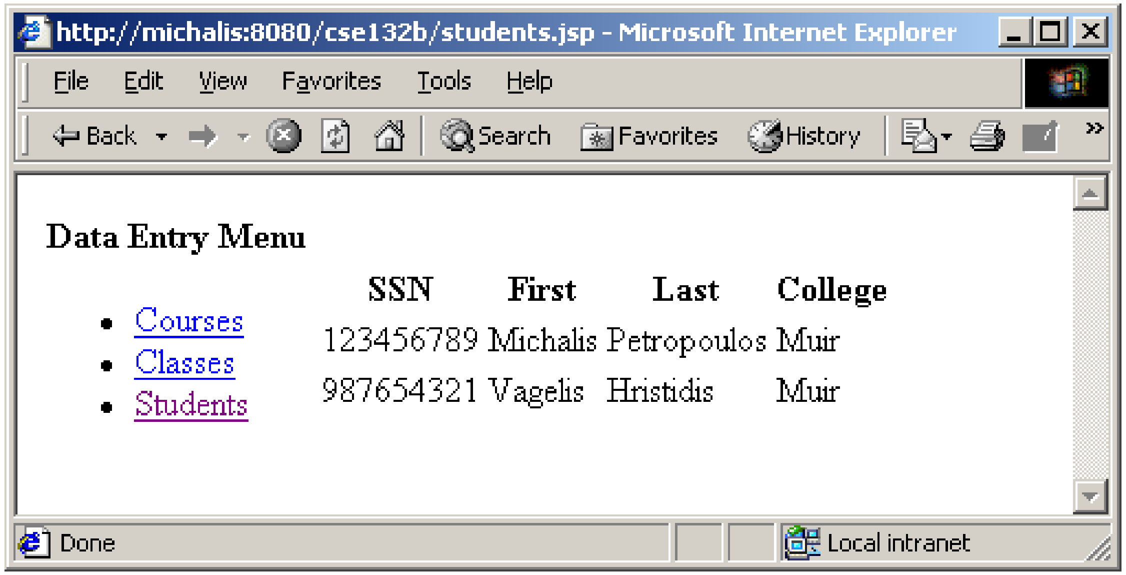
```
<%
// Create the statement
Statement statement = conn.createStatement();

// Use the statement to SELECT the student attributes
// FROM the Student table.
ResultSet rs = statement.executeQuery
  ("SELECT * FROM Student");
%>
```

Entry Form - First Attempt

Presentation Code

```
<table>
  <tr>
    <th>SSN</th>
    <th>First</th>
    <th>Last</th>
    <th>College</th>
  </tr>

<%
  // Iterate over the ResultSet
  while ( rs.next() ) {
%>
    Iteration Code
<%
  }
%>
</table>
```

Entry Form - First Attempt


http://michalis:8080/cse132b/students.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
Local intranet

Data Entry Menu

- Courses
- Classes
- Students

SSN	First	Last	College
123456789	Michalis	Petropoulos	Muir
987654321	Vagelis	Hristidis	Muir

Entry Form - First Attempt

Iteration Code

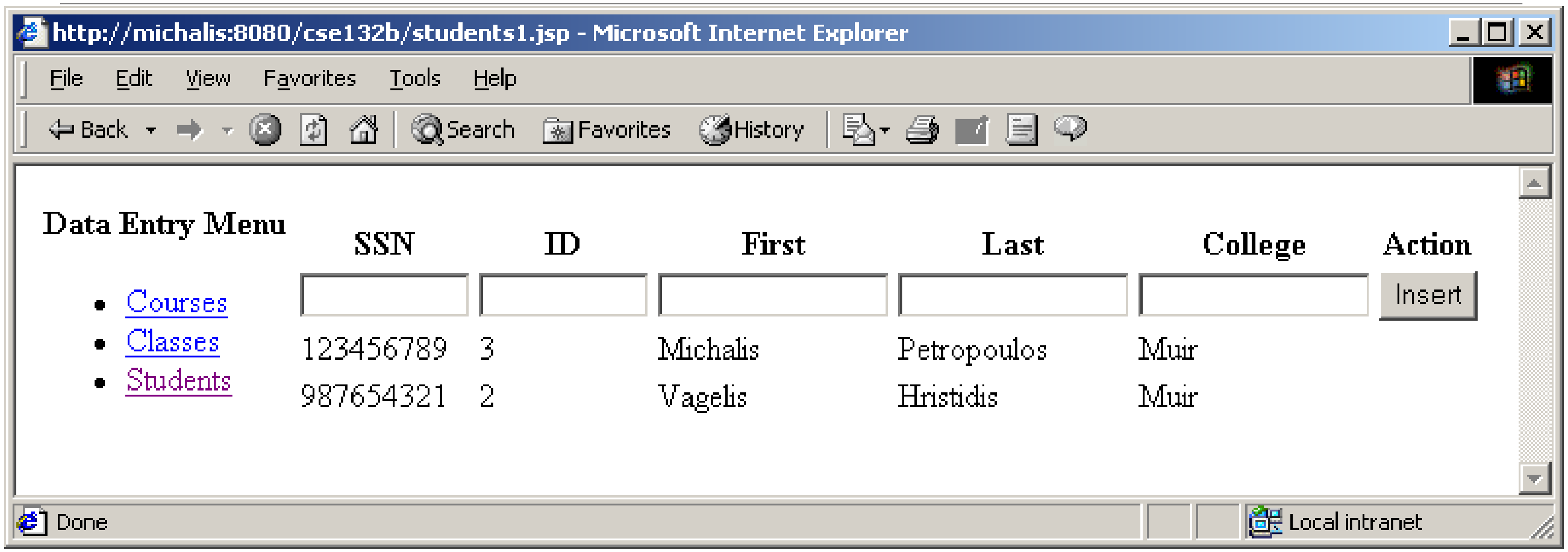
```
<tr>
  <%-- Get the SSN, which is a number --%>
  <td><%= rs.getInt("SSN") %></td>

  <%-- Get the ID --%>
  <td><%= rs.getString("ID") %></td>

  <%-- Get the FIRSTNAME --%>
  <td><%= rs.getString("FIRSTNAME") %></td>

  <%-- Get the LASTNAME --%>
  <td><%= rs.getString("LASTNAME") %></td>

  <%-- Get the COLLEGE --%>
  <td><%= rs.getString("COLLEGE") %></td>
</tr>
```

Entry Form - First Attempt

Close Connectivity Code

```
<%
// Close the ResultSet
rs.close();

// Close the Statement
statement.close();

// Close the Connection
conn.close();

} catch (SQLException sqle) {
  out.println(sqle.getMessage());
} catch (Exception e) {
  out.println(e.getMessage());
}
%>
```

Entry Form - Second Attempt


http://michalis:8080/cse132b/students1.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
Local intranet

Data Entry Menu

- Courses
- Classes
- Students

SSN	ID	First	Last	College	Action
					Insert
123456789	3	Michalis	Petropoulos	Muir	
987654321	2	Vagelis	Hristidis	Muir	

Entry Form - Second Attempt

JSP Code

```
<html>
<body>
    <table>
        <tr>
            <td>
                Open connection code
                Insertion Code
                Statement code
                Presentation code
                Close connection code
            </td>
        </tr>
    </table>
</body>
</html>
```

Entry Form - Second Attempt

Insertion Code

```
// Check if an insertion is requested
String action = request.getParameter("action");
if (action != null && action.equals("insert")) {
conn.setAutoCommit(false);

// Create the prepared statement and use it to
// INSERT the student attrs INTO the Student table.
PreparedStatement pstmt = conn.prepareStatement(
("INSERT INTO Student VALUES (?, ?, ?, ?, ?)"));
pstmt.setInt(1,Integer.parseInt(request.getParameter("SSN
")));
pstmt.setString(2, request.getParameter("ID"));
…
pstmt.executeUpdate();
conn.commit();
conn.setAutoCommit(true);
}
```

Entry Form - Second Attempt

Presentation Code

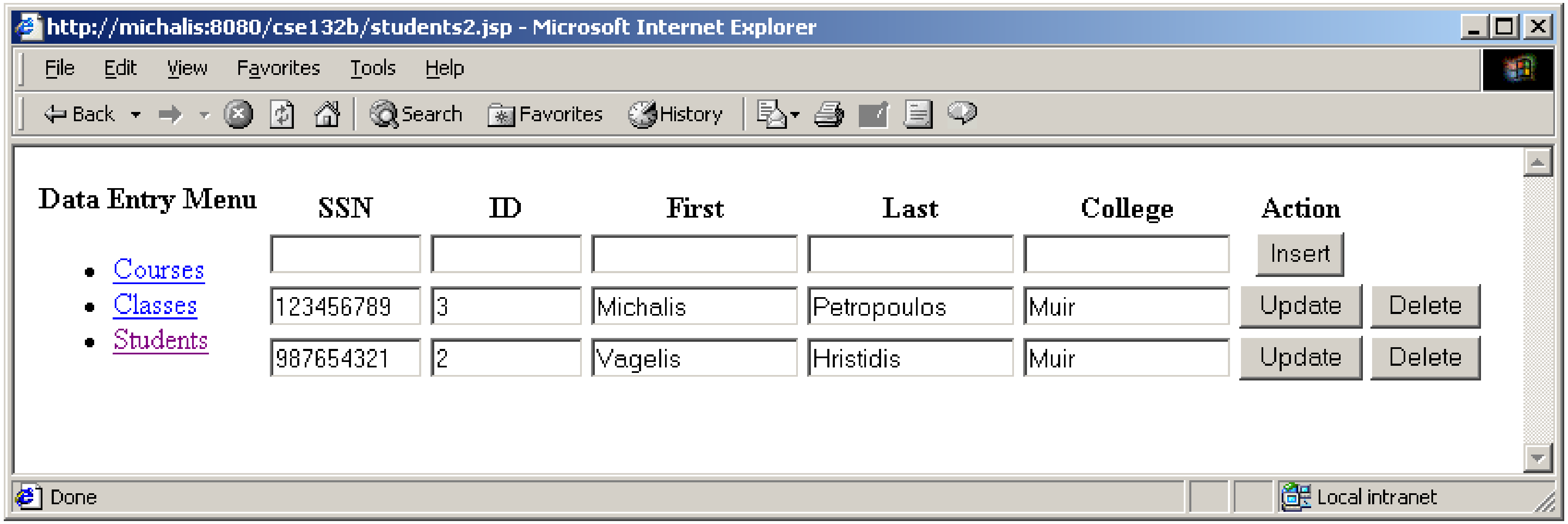
```
<table>
  <tr>
    <th>SSN</th>
    <th>First</th>
    <th>Last</th>
    <th>College</th>
  </tr>
  Insert Form Code
<%
  // Iterate over the ResultSet
  while ( rs.next() ) {
%>
    Iteration Code
<%
  }
%>
</table>
```

Entry Form - Second Attempt

Insert Form Code

```
<tr>
  <form action="students.jsp" method="get">
    <input type="hidden" value="insert" name="action">
    <th><input value="" name="SSN" size="10"></th>
    <th><input value="" name="ID" size="10"></th>
    <th><input value="" name="FIRSTNAME" size="15"></th>
    <th><input value="" name="LASTNAME" size="15"></th>
    <th><input value="" name="COLLEGE" size="15"></th>
    <th><input type="submit" value="Insert"></th>
  </form>
</tr>
```

Entry Form - Third Attempt


http://michalis:8080/cse132b/students2.jsp - Microsoft Internet Explorer
File Edit View Favorites Tools Help
Back Search Favorites History
Done
Local intranet

Data Entry Menu

- Courses
- Classes
- Students

SSN	ID	First	Last	College	Action	
					Insert	
123456789	3	Michalis	Petropoulos	Muir	Update	Delete
987654321	2	Vagelis	Hristidis	Muir	Update	Delete

Entry Form - Third Attempt

UNIVERSITI TEKNOLOGI MARA

JSP Code

```
<html>
<body>
    <table>
        <tr>
            <td>
                Open connection code
                Insertion Code
                Update Code
                Delete Code
                Statement code
                Presentation code
                Close connection code
            </td>
        </tr>
    </table>
</body>
</html>
```

Entry Form - Third Attempt

Update Code

```
// Check if an update is requested
if (action != null && action.equals("update")) {

conn.setAutoCommit(false);

// Create the prepared statement and use it to
// UPDATE the student attributes in the Student table.
PreparedStatement pstatement = conn.prepareStatement(
"UPDATE Student SET ID = ?, FIRSTNAME = ?, " +
"LASTNAME = ?, COLLEGE = ? WHERE SSN = ?");

pstatement.setString(1, request.getParameter("ID"));
pstatement.setString(2, request.getParameter("FIRSTNAME"));
…
int rowCount = pstatement.executeUpdate();

conn.setAutoCommit(false);
conn.setAutoCommit(true);
}
```

Entry Form - Third Attempt

Delete Code

```
// Check if a delete is requested
if (action != null && action.equals("delete")) {

conn.setAutoCommit(false);

// Create the prepared statement and use it to
// DELETE the student FROM the Student table.
PreparedStatement pstmt = conn.prepareStatement(
"DELETE FROM Student WHERE SSN = ?");

pstmt.setInt(1,
 Integer.parseInt(request.getParameter("SSN")));
int rowCount = pstmt.executeUpdate();

conn.setAutoCommit(false);
conn.setAutoCommit(true);
}
```

Entry Form - Third Attempt

Presentation Code

```
<table>
  <tr>
    <th>SSN</th>
    <th>First</th>
    <th>Last</th>
    <th>College</th>
  </tr>
  Insert Form Code
<%
  // Iterate over the ResultSet
  while ( rs.next() ) {
%>
    Iteration Code
<%
  }
%>
</table>
```

Entry Form - Third Attempt

Iteration Code

```
<tr>
  <form action="students.jsp" method="get">
    <input type="hidden" value="update" name="action">
    <td><input value="<%= rs.getInt("SSN") %>" name="SSN"></td>
    <td><input value="<%= rs.getString("ID") %>" name="ID"></td>
…
    <td><input type="submit" value="Update"></td>
  </form>
  <form action="students2.jsp" method="get">
    <input type="hidden" value="delete" name="action">
    <input type="hidden" value="<%= rs.getInt("SSN") %>" name="SSN">
    <td><input type="submit" value="Delete"></td>
  </form>
</tr>
```

JDBC and beyond

(JNDI) Java Naming and Directory Interface
- API for network-wide sharing of information about users, machines, networks, services, and applications
- Preserves Java's object model

(JDO) Java Data Object
- Models persistence of objects, using RDBMS as repository
- Save, load objects from RDBMS

(SQLJ) Embedded SQL in Java
- Standardized and optimized by Sybase, Oracle and IBM
- Java extended with directives: # sql
- SQL routines can invoke Java methods
- Maps SQL types to Java classes

JDBC references

JDBC Data Access API – JDBC Technology Homepage
- http://java.sun.com/products/jdbc/index.html

JDBC Database Access – The Java Tutorial
- http://java.sun.com/docs/books/tutorial/jdbc/index.html

JDBC Documentation
- http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/index.html

java.sql package
- http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html

JDBC Technology Guide: Getting Started
- http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html

JDBC API Tutorial and Reference (book)
- http://java.sun.com/docs/books/jdbc/

JDBC

JDBC Data Access API – JDBC Technology Homepage

- http://java.sun.com/products/jdbc/index.html

JDBC Database Access – The Java Tutorial

- http://java.sun.com/docs/books/tutorial/jdbc/index.html

JDBC Documentation

- http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/index.html

java.sql package

- http://java.sun.com/j2se/1.4.2/docs/api/java/sql/package-summary.html

JDBC Technology Guide: Getting Started

- http://java.sun.com/j2se/1.4.2/docs/guide/jdbc/getstart/GettingStartedTOC.fm.html

JDBC API Tutorial and Reference (book)

- http://java.sun.com/docs/books/jdbc/