Programming in Java

Dr. Nyein Aye Maung Maung
Dr. Eng (Ritsumeikan University, Japan)
Lecturer
Computer Engineering and Information Technology Dept.
Yangon Technological University

Course Schedule

Week	Topics
Week 1	Overview of JAVA, Data Types, Variables and Arrays
Week 2	Operators and Control Statements
Week 3	Classes and A closer look at Methods and Classes
Week 4	Inheritance, Polymorphism, Abstraction and Encapsulation
Week 5	Packages and Interfaces
Week 6	Exception Handling and Multi-threaded Programming
Week 7	String Handling
Week 8	Exploring java.lang and More utilities classes
Week 9	Java Collections Framework
Week 10	Java I/O
Week 11	Basic Graphical User Interface
Week 12	Event Handling
Week 13	Database Programming
Week 14	Applet and Networking

Lecture 13

Database Programming

Outline of Class

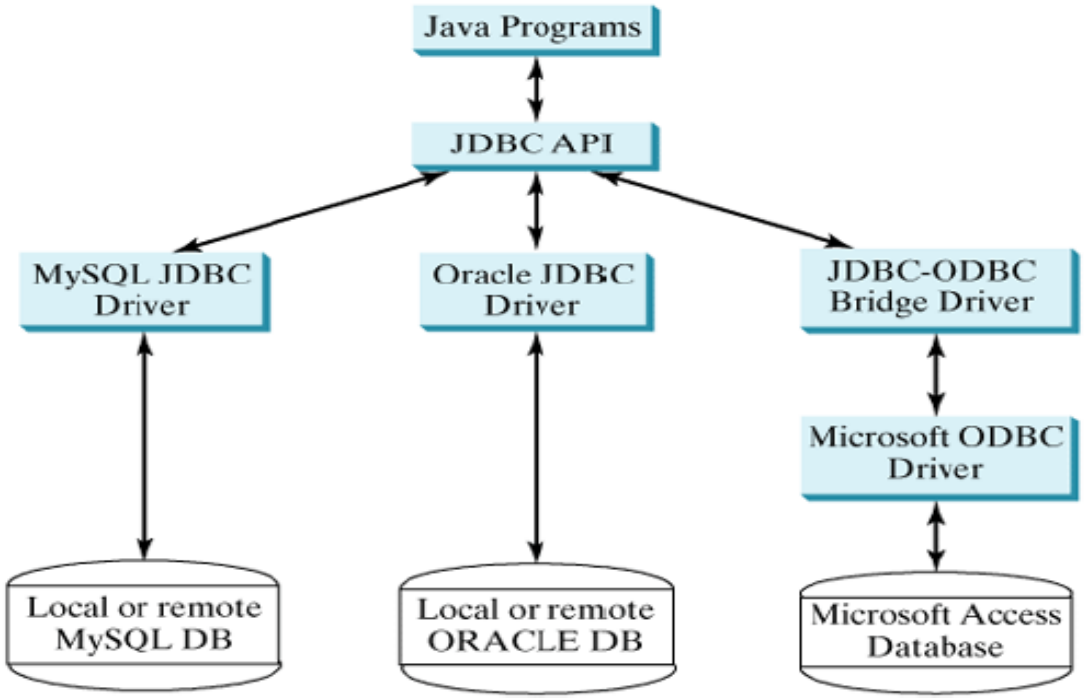
- JDBC
- Database Programming

Overview of JDBC Technology

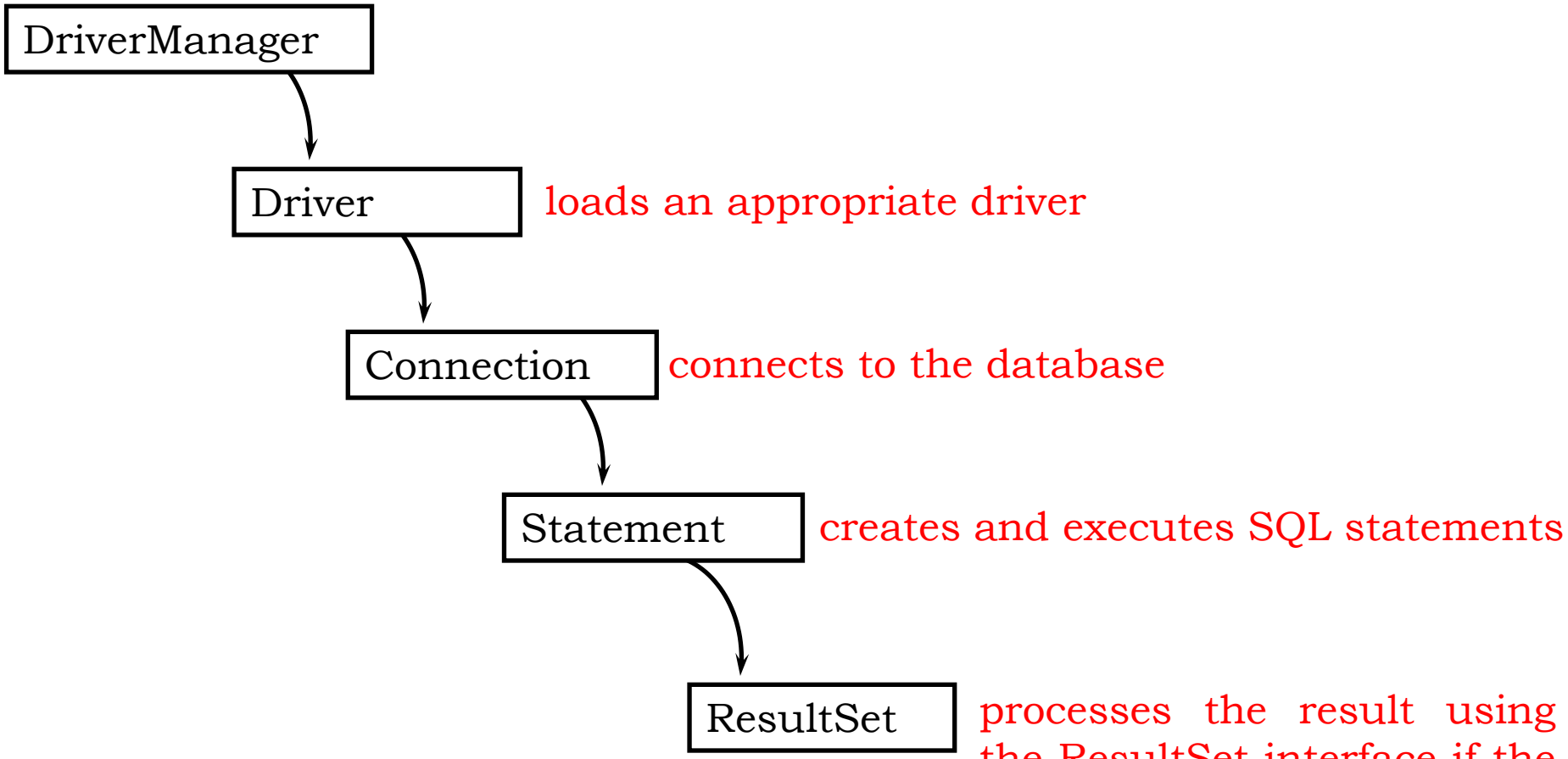
- JDBC (**Java Database Connectivity**) API provides a standard library for accessing and manipulating a wide range of relational databases
- API standardizes
  - ✓ way to establish connection to database
  - ✓ approach to initiating queries
  - ✓ method to create stored (parameterized) queries
  - ✓ the data structure of query result (table)
  - ✓ Determining the number of columns
  - ✓ Looking up metadata, etc.
- API does not standardize SQL syntax

Functions of JDBC

- Using JDBC, it is easy to send SQL statements to virtually any relational database.
- JDBC makes it possible to do three things:
  - ***establish*** a connection with a database
  - ***send*** SQL statements
  - ***process*** the results



JDBC Class Usage



JDBC API

- The Java environment provides you the JDBC API necessary to create Java applications that are capable of interacting with a database.
- Some classes contained in the java.sql package.
  - **java.sql.DriverManager** : loads driver, and creates the connection to the database
  - **java.sql.Driver** : represents a driver
  - **java.sql.Connection** : represents a connection to the database
  - **Java.sql.Statement** : executes statements
  - **Java.sql.ResultSet** : this holds results of executing statement

(1) Loading Drivers

- Import JDBC Packages import java.sql.*;
- Load drivers.

```
try {
        Class.forName("oracle.jdbc.driver.OracleDriver");

} catch (ClassNotFoundException  cnfe) {
        System.out.println("Error loading driver: " cnfe);
}
```

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

(2) Establishing connections

- To connect to a database, use the static method getConnection(databaseURL) in the DriverManager class, as follows:

  Connection connection = DriverManager.getConnection(databaseURL);

  where databaseURL is the unique identifier of the database on the Internet.

```
Connection connection = DriverManager.getConnection
                        ("jdbc:mysql://localhost/test" , “username”, “password);
Connection connection = DriverManager.getConnection
    ("jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl" , "scott" , "tiger" );
```

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

(3) Creating **S**tatements

- **Statement** stmt = connection.**createStatement()**;

eg. Statement stmt = conn.createStatement(
ResultSet.TYPE_SCROLL_INSENSITIVE,
ResultSet.CONCUR_UPDATABLE);

- // TYPE_SCROLL_INSENSITIVE: ResultSet is scrollable and does not

reflect changes by others while it is opened for processing.

- // CONCUR_UPDATABLE: Resultset is updatable

(4) Executing statements

- An SQL update statement can be executed using
  1. **executeUpdate**(String sql)
     - ✓it may be create, drop, insert, update, delete etc.
  2. **executeQuery**(String sql)
     - ✓to execute SELECT query. It returns the object of ResultSet.

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

ResultSet resultSet = statement.executeQuery ("select firstName, mi, l
astName from Student where lastName " + " = "Smith'" );
```

(5) Processing ResultSet

- The ResultSet maintains a table whose current row can be retrieved
- The initial row position is null
- Use the next method to move to the next row and the various get methods to retrieve values from a current row

  - **getInt(int columnIndex):** return the data of specified column index of the current row as int.
  - **getInt(String columnName):** return the data of specified column name of the current row as int.
  - **getString(int columnIndex):** return the data of specified column index of the current row as String
  - **getString(String columnName):** return the data of specified column name of the current row as String.

column indexes

```
while  (resultSet.next())
        System.out.println(resultSet.getString(1)
      + " "+ resultSet.getString(2 )
       + ". "  + resultSet.getString(3 ));
//1→ firstName, 2→mi, 3→lastName
```

- Example 1: Write a program that connects to a local MySQL database and displays the students whose last name is Smith .

```
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", "root", "admin");
    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();
}}
```

Prepared Statements

- Enable you to create parameterized SQL statements.
- Pre-compiled SQL statements which is useful if the same SQL is to be executed repeatedly, for example, in a loop.
- Straightforward syntax: just insert question marks for any parameters that you'll be substituting later on before you send the SQL to the database.
- Example

```
Statement  pstmt = connection.prepareStatement
("insert into Student (firstName, mi, lastName) " +"values (?, ?, ?)");
pstmt.setString(1, "Jack");
pstmt.setString(2, "A");
pstmt.setString(3, "Ryan");
pstmt. executeUpdate();
```

- setInt(int paramIndex, int value), setFloat(int paramIndex, float value) and setDouble(int paramIndex, double value) methods can be used for other primitive data types

Retrieving Metadata

- JDBC provides
  - The **DatabaseMetaData interface**
    - for obtaining database URL, username, JDBC driver name

      DatabaseMetaData dbMetaData =connection.**getMetaData**();
  - The **ResultSetMetaData interface**
    - for obtaining information on the specific ResultSet, such as column count and column names.

      ResultSetMetaData rsMetaData = resultSet.**getMetaData**();
- Obtaining Database Tables
  - use **getTables** method to get the names of tables in the database
    - public java.sql.ResultSet getTables(String catalog, String schema, String table, String[] types)

Example 2: Retrieving Database Metadata

```
import java.sql.*;
public class MetaDataTest {
public static void main(String[] args)
throws SQLException, ClassNotFoundException {
Class.forName("com.mysql.jdbc.Driver"); // Load the JDBC driver
System.out.println("Driver loaded");
Connection connection = DriverManager.getConnection
("jdbc:mysql://localhost/test");
System.out.println("Database connected");
DatabaseMetaData dbMetaData = connection.getMetaData();
System.out.println("database URL: " + dbMetaData.getURL());
System.out.println("database username: " + dbMetaData.getUserName());
System.out.println("database product name: " +
dbMetaData.getDatabaseProductName());
System.out.println("database product version: " +
dbMetaData.getDatabaseProductVersion());
System.out.println("JDBC driver name: " + dbMetaData.getDriverName());
```

```
System.out.println("JDBC driver version: "
+dbMetaData.getDriverVersion());
System.out.println("JDBC driver major version: " +
dbMetaData.getDriverMajorVersion());
System.out.println("JDBC driver minor version: "
+dbMetaData.getDriverMinorVersion());
System.out.println("Max number of connections: " +
dbMetaData.getMaxConnections());
System.out.println("MaxTableNameLength: " +
dbMetaData.getMaxTableNameLength());
System.out.println("MaxColumnsInTable: " +
dbMetaData.getMaxColumnsInTable());
// Close the connection
connection.close();
}
}
```

Example 3: Get the names of table in a database

```
import java.sql.*;
public class getTableTest {

  public static void main(String[] args) throws Exception {
    Connection conn = getMySqlConnection();
    System.out.println("Got Connection.");
    Statement st = conn.createStatement();
    st.executeUpdate("create table survey (id int,name varchar);");
    st.executeUpdate("insert into survey (id,name ) values (1,'nameValue')");

    ResultSet rs = null;
    DatabaseMetaData meta = conn.getMetaData();
    rs = meta.getTables(null, null, null, new String[]{"TABLE"});

    while (rs.next()) {
      String tableName = rs.getString("TABLE_NAME");
      System.out.println("tableName=" + tableName);
    }
    st.close();
    conn.close();
  }
```

```
public static Connection getMySqlConnection() throws Exception {
    String driver = " com.mysql.jdbc.Driver ";
    String url = "jdbc:mysql://localhost/SurveyDB";
    String username = "root";
    String password = "root";

    Class.forName(driver); // load MySQL driver
    Connection conn = DriverManager.getConnection(url, username,
password);
    return conn;
  }
}
```

ResultSetMetaData

- What's the number of columns in the ResultSet?
- What's a column's name?
- What's a column's SQL type?
- What's the column's normal max width in chars?
- What's the suggested column title for use in printouts and displays?
- What's a column's number of decimal digits?
- Does a column's case matter?
- Is the column a cash value?
- Will a write on the column definitely succeed?
- Can you put a NULL in this column?
- Is a column definitely not writable?
- Can the column be used in a where clause?
- Is the column a signed number?
- Is it possible for a write on the column to succeed?
- and so on...

ResultSetMetaData rsMetaData = resultSet.**getMetaData**();

getColumnCount(): method to find the number of columns in the result

getColumnName(int): method to get the column names

Example 4: Display column names and contents

```
import java.sql.*;
public class ResultSetMetaTest {
public static void main(String[] args)
throws SQLException, ClassNotFoundException {
// Load the JDBC driver
Class.forName("com.mysql.jdbc.Driver");
System.out.println("Driver loaded");
Connection connection =
DriverManager.getConnection("jdbc:mysql://localhost/test");
System.out.println("Database connected");
Statement statement = connection.createStatement();
ResultSet resultSet = statement.executeQuery("select * from Enrollment");
ResultSetMetaData rsMetaData = resultSet.getMetaData();
        for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
        System.out.printf("%-12s\t", rsMetaData.getColumnName(i));
        System.out.println();
```

```
//Iterate through the result and print the student names
while (resultSet.next()) {
for (int i = 1; i <= rsMetaData.getColumnCount(); i++)
        System.out.printf("%-12s\t", resultSet.getObject(i));
System.out.println();
}
// Close the connection
connection.close();
}
}
```

```
Driver loaded
Database connected
ssn             courseId        dateRegistered  grade
444111110       11111           2007-04-13      A
444111110       11113           2007-04-13      C
444111111       11111           2007-04-13      D
```

RowSet Interface

- Configure the database connection and prepares query statements automatically
- Provide several set methods that allow you
  - to specify the properties needed to establish a connection (such as the database URL, user name and password of the database) and
  - to create a Statement (such as a query)
- Provide several get methods that return these properties
- Two types of RowSet objects—
  - Connected (**JdbcRowSet**)
    - A connected RowSet object connects to the database once and remains connected until the application terminates
  - Disconnected (**CachedRowSet**)
    - A disconnected RowSet object connects to the database, executes a query to retrieve the data from the database and then closes the connection

Example 5: Write a program that displays the contents of the authors table using JdbcRowSet.

```
2  // Displaying the contents of the authors table using JdbcRowSet.
3  import java.sql.ResultSetMetaData;
4  import java.sql.SQLException;
5  import javax.sql.rowset.JdbcRowSet;
6  import com.sun.rowset.JdbcRowSetImpl; // Sun's JdbcRowSet implementation
7
8  public class JdbcRowSetTest
9  {
10    // JDBC driver name and database URL
11    static final String DRIVER = "com.mysql.jdbc.Driver";
12    static final String DATABASE_URL = "jdbc:mysql://localhost/books";
13    static final String USERNAME = "jhtp7";
14    static final String PASSWORD = "jhtp7";
15
16    // constructor connects to database, queries database, processes
17    // results and displays results in window
18    public JdbcRowSetTest()
19    {
20       // connect to database books and query database
21       try
22       {
23          Class.forName( DRIVER );
24
25          // specify properties of JdbcRowSet
26          JdbcRowSet rowSet = new JdbcRowSetImpl();
27          rowSet.setUrl( DATABASE_URL ); // set database URL
28          rowSet.setUsername( USERNAME ); // set username
29          rowSet.setPassword( PASSWORD ); // set password
30          rowSet.setCommand( "SELECT * FROM authors" ); // set query
31          rowSet.execute(); // execute query
32
```

```
33            // process query results
34            ResultSetMetaData metaData = rowSet.getMetaData();
35            int numberOfColumns = metaData.getColumnCount();
36            System.out.println( "Authors Table of Books Database:\n" );
37
38            // display rowset header
39            for ( int i = 1; i <= numberOfColumns; i++ )
40               System.out.printf( "%-8s\t", metaData.getColumnName( i ) );
41            System.out.println();
42
43            // display each row
44            while ( rowSet.next() )
45            {
46               for ( int i = 1; i <= numberOfColumns; i++ )
47                  System.out.printf( "%-8s\t", rowSet.getObject( i ) );
48               System.out.println();
49            } // end while
50
51            // close the underlying ResultSet, Statement and Connection
52            rowSet.close();
53         } // end try
```

```
54         catch ( SQLException sqlException )
55         {
56            sqlException.printStackTrace();
57            System.exit( 1 );
58         } // end catch
59         catch ( ClassNotFoundException classNotFound )
60         {
61            classNotFound.printStackTrace();
62            System.exit( 1 );
63         } // end catch
64      } // end DisplayAuthors constructor
65
66      // launch the application
67      public static void main( String args[] )
68      {
69         JdbcRowSetTest application = new JdbcRowSetTest();
70      } // end main
71   } // end class JdbcRowSetTest
```

Authors Table of Books Database:

authorID	firstName	lastName
1	Harvey	Deitel
2	Paul	Deitel
3	Andrew	Goldberg
4	David	Choffnes

Practical Exercises on Java Database Programming

Manual

- XAMPP Configuration
  - Install XAMPP
  - Launch XAMPP Manager
  - Start Apache Web Server and MySQL Server

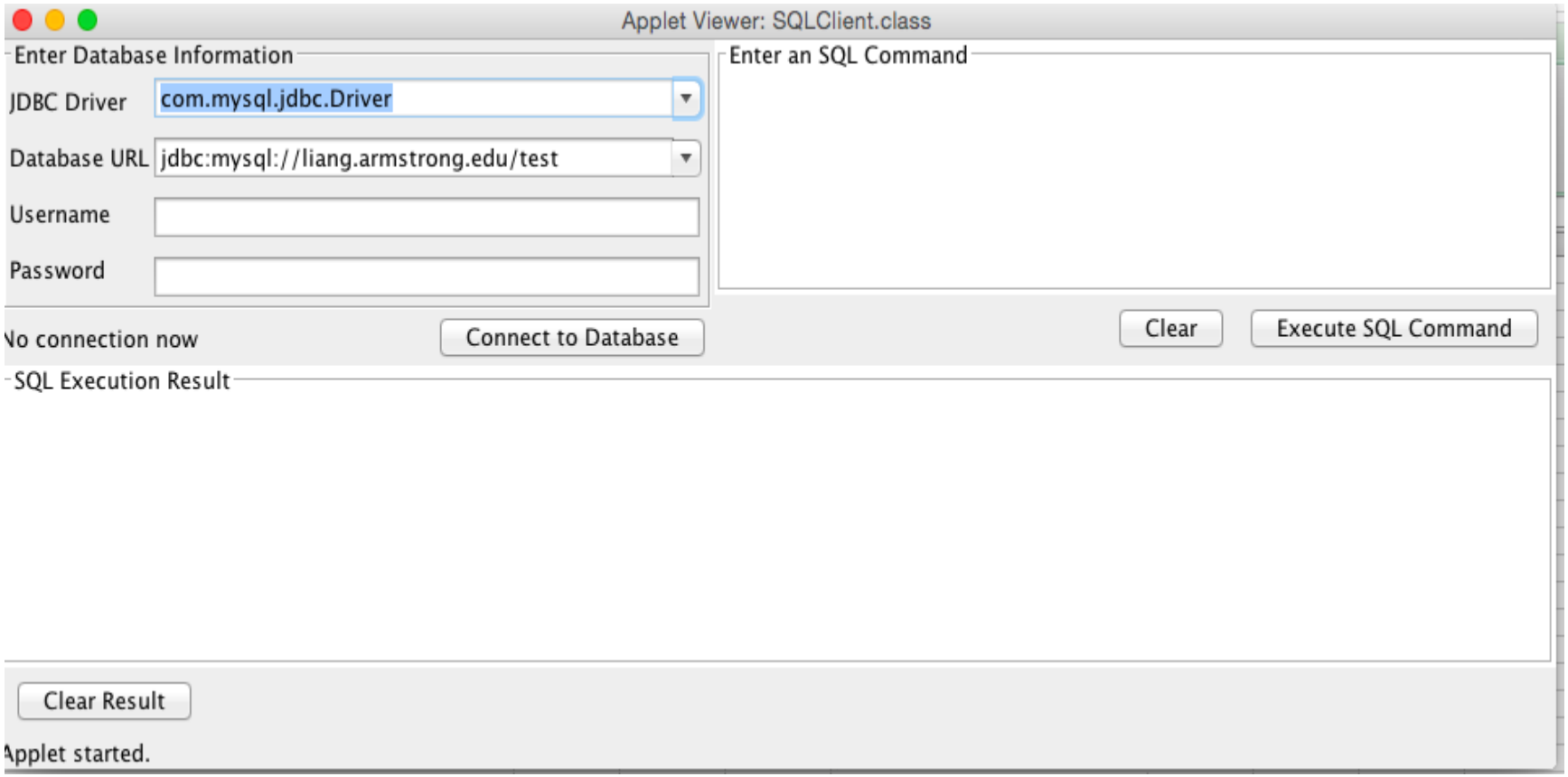
- Setting up a database
  - Web browser → type: http://localhost
    - go to phpMyAdmin tab
    - Create Database – Week9

    (1) Create Table – Student (firstName, mi, lastName)

    (2) Create Table –

    (3) Create Table -

- **Exercise 1**: SQL Client GUI



```
import libraries
class SQLClient extends Jframe{
//Variable declarations
//Constructor
SQLClient(){
//Design GUI Layout
//Add listener to Connect to Database button
//Add listener to Execute SQL Command button
//Add listener to Clear button
//Add listener to Clear Result button
}
//Method to be performed when Connect to Database button is clicked
//Method to be performed when Execute SQL Command button is clicked
//Main method
}
```

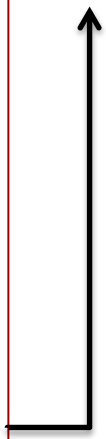
```
public class SQLClient extends JFrame{
        private Connection connection;
        // Statement to execute SQL commands
        private Statement statement;
        // Text area to enter SQL commands
        private JTextArea jtasqlCommand = new JTextArea();
        // Text area to display results from SQL commands
        private JTextArea jtaSQLResult = new JTextArea();
        // JDBC info for a database connection
        JTextField jtfUsername = new JTextField();
        JPasswordField jpfPassword = new JPasswordField();
        JComboBox jcboURL = new JComboBox(new String[] {
        "jdbc:mysql://localhost/Week9",
        "jdbc:odbc:exampleMDBDataSource",
        "jdbc:oracle:thin:@liang.armstrong.edu:1521:orcl"});
        JComboBox jcboDriver = new JComboBox(new String[] {
        "com.mysql.jdbc.Driver", "sun.jdbc.odbc.JdbcOdbcDriver",
        "oracle.jdbc.driver.OracleDriver"});

        JButton jbtExecuteSQL = new JButton("Execute SQL Command");
        JButton jbtClearSQLCommand = new JButton("Clear");
        JButton jbtConnectDB1 = new JButton("Connect to Database");
        JButton jbtClearSQLResult = new JButton("Clear Result");
        // Create titled borders
        Border titledBorder1 = new TitledBorder("Enter an SQL Command");
        Border titledBorder2 = new TitledBorder("SQL Execution Result");
        Border titledBorder3 = new TitledBorder("Enter Database Information");

        JLabel jlblConnectionStatus = new JLabel("No connection now");
```

```
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.TitledBorder;
import java.awt.*;
import java.awt.event.*;
import com.mysql.jdbc.*;
import java.sql.*;
import java.sql.Connection;
import java.sql.Statement;
```

```
public SQLClient()
{
        JScrollPane jScrollPane1 = new JScrollPane(jtasqlCommand);
        jScrollPane1.setBorder(titledBorder1);
        JScrollPane jScrollPane2 = new JScrollPane(jtaSQLResult);
        jScrollPane2.setBorder(titledBorder2);

        JPanel jPanel1 = new JPanel(new FlowLayout(FlowLayout.RIGHT));
        jPanel1.add(jbtClearSQLCommand);
        jPanel1.add(jbtExecuteSQL);

        JPanel jPanel2 = new JPanel();
        jPanel2.setLayout(new BorderLayout());
        jPanel2.add(jScrollPane1, BorderLayout.CENTER);
        jPanel2.add(jPanel1, BorderLayout.SOUTH);
        jPanel2.setPreferredSize(new Dimension(100, 100));

        JPanel jPanel3 = new JPanel();
        jPanel3.setLayout(new BorderLayout());
        jPanel3.add(jlblConnectionStatus, BorderLayout.CENTER);
        jPanel3.add(jbtConnectDB1, BorderLayout.EAST);

        JPanel jPanel4 = new JPanel();
        jPanel4.setLayout(new GridLayout(4, 1, 10, 5));
        jPanel4.add(jcboDriver);
        jPanel4.add(jcboURL);
        jPanel4.add(jtfUsername);
        jPanel4.add(jpfPassword);
```

```
JPanel jPanel5 = new JPanel();
jPanel5.setLayout(new GridLayout(4, 1));
jPanel5.add(new JLabel("JDBC Driver"));
jPanel5.add(new JLabel("Database URL"));
jPanel5.add(new JLabel("Username"));
jPanel5.add(new JLabel("Password"));

JPanel jPanel6 = new JPanel();
jPanel6.setLayout(new BorderLayout());
jPanel6.setBorder(titledBorder3);
jPanel6.add(jPanel4, BorderLayout.CENTER);
jPanel6.add(jPanel5, BorderLayout.WEST);

JPanel jPanel7 = new JPanel();
jPanel7.setLayout(new BorderLayout());
jPanel7.add(jPanel3, BorderLayout.SOUTH);
jPanel7.add(jPanel6, BorderLayout.CENTER);

JPanel jPanel8 = new JPanel();
jPanel8.setLayout(new BorderLayout());
jPanel8.add(jPanel2, BorderLayout.CENTER);
jPanel8.add(jPanel7, BorderLayout.WEST);

JPanel jPanel9 = new JPanel(new FlowLayout(FlowLayout.LEFT));
```

```
jPanel9.add(jbtClearSQLResult);

jcboURL.setEditable(true);
jcboDriver.setEditable(true);

add(jPanel8, BorderLayout.NORTH);
add(jScrollPane2, BorderLayout.CENTER);
add(jPanel9, BorderLayout.SOUTH);
```



```
jbtExecuteSQL.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
executeSQL();
}
});
jbtConnectDB1.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
connectToDB();
}
});
jbtClearSQLCommand.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
jtasqlCommand.setText(null);
}
});
jbtClearSQLResult.addActionListener(new ActionListener() {
public void actionPerformed(ActionEvent e) {
jtaSQLResult.setText(null);
}
});
}
```

```
/** Connect to DB */
private void connectToDB() {
// Get database information from the user input
String driver = (String)jcboDriver.getSelectedItem();
String url = (String)jcboURL.getSelectedItem();
String username = jtfUsername.getText().trim();
 String password = new String(jpfPassword.getPassword());

// Connection to the database
try {
Class.forName(driver);
connection = DriverManager.getConnection(url, username, password);
jlblConnectionStatus.setText("Connected to " + url);
}
catch (java.lang.Exception ex) {
ex.printStackTrace();
}
}
```

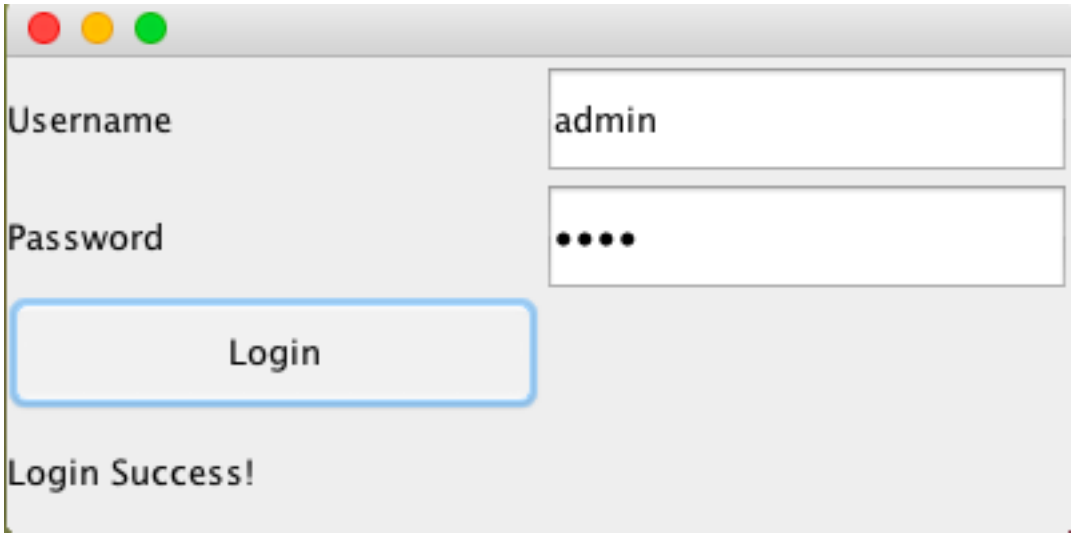
```
/** Execute SQL commands */
private void executeSQL() {
if (connection == null) {
jtaSQLResult.setText("Please connect to a database first");
return;
}
else {
String sqlCommands = jtasqlCommand.getText().trim();
String[] commands = sqlCommands.replace('\n', ' ').split(";");

for (String aCommand: commands) {
if (aCommand.trim().toUpperCase().startsWith("SELECT")) {
processSQLSelect(aCommand);
}
else {
processSQLNonSelect(aCommand);
}
}
}
}
```

```
/** Execute SQL SELECT commands */
private void processSQLSelect(String sqlCommand) {
try {
// Get a new statement for the current connection
statement = connection.createStatement();

// Execute a SELECT SQL command
ResultSet resultSet = statement.executeQuery(sqlCommand);
// Find the number of columns in the result set
int columnCount = resultSet.getMetaData().getColumnCount();
String row = " ";

// Display column names
for (int i = 1; i <= columnCount; i++) {
row += resultSet.getMetaData().getColumnName(i) + "\t";
}
jtaSQLResult.append(row + '\n');

while (resultSet.next()) {
// Reset row to empty
row = " ";
for (int i = 1; i <= columnCount; i++) {
            // A non-String column is converted to a string
row += resultSet.getString(i) + "\t";
}
```

```
jtaSQLResult.append(row + '\n');
}
}
catch (SQLException ex) {
jtaSQLResult.setText(ex.toString());
}
}
```

```
        /** Execute SQL DDL, and modification commands */
        private void processSQLNonSelect(String sqlCommand) {
        try {
        // Get a new statement for the current connection
        statement = connection.createStatement();

        // Execute a non-SELECT SQL command
        statement.executeUpdate(sqlCommand);

        jtaSQLResult.setText("SQL command executed");
        }
        catch (SQLException ex) {
        jtaSQLResult.setText(ex.toString());
        }
        }

    public static void main(String[] args) {
        // TODO Auto-generated method stub
        SQLClient client=new SQLClient();
        client.setSize(400, 300); // Set the client size
      client.setLocationRelativeTo(null); // New since JDK 1.4
      client.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
      client.setVisible(true); // Display the client
    }

}
```

Exercise 2: The table named 'User' holds username and password attributes. Write a program contains text fields for accepting username and password, and login button. When the user enters username and password, the program checks login user is valid or not based on User table by matching the entered user name and password against the values of the username and password columns in the table. If login user is valid, it will display Login Success Message. If not, it will display 'Invalid username or password'.



```
import java.sql.*;
import javax.swing.*;
import java.awt.event.*;
import java.awt.*;
```

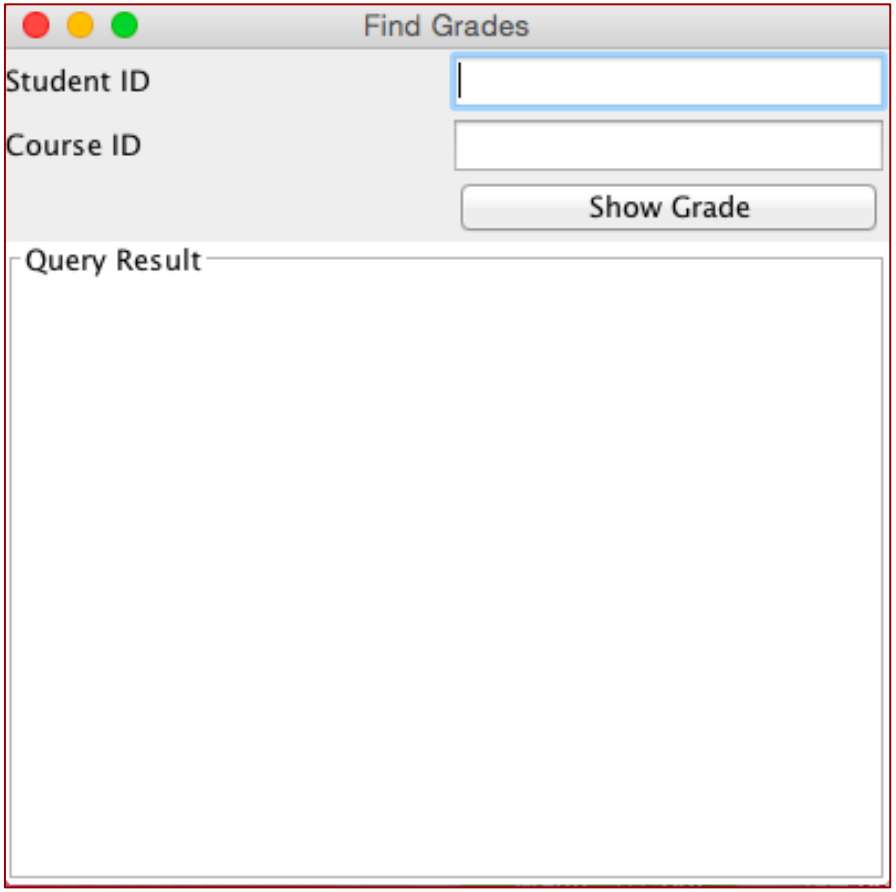
```
public class login extends JFrame{
JTextField uName=new JTextField(20);
JPasswordField pwd=new JPasswordField(15);
JButton login=new JButton("Login");
JLabel result=new JLabel();
Boolean loginSuccess=false;
Statement stmt;
public login()
{
        JPanel p=new JPanel(new GridLayout(4,2));
        p.add(new JLabel("Username"));
        p.add(uName);
        p.add(new JLabel("Password"));
        p.add(pwd);
        p.add(login);
        p.add(new JLabel(" "));
        p.add(result);
        p.add(new JLabel(" "));
        add(p);
        login.addActionListener(new ActionListener(){
                public void actionPerformed(ActionEvent e)
                {
                        initializeDB();
                        loginSuccess=CheckDB();
                        if(loginSuccess)
                                result.setText("Login Success!");
                        else
                                result.setText("Invalid Username and Password!");
                }
        });
}
```

```
Boolean CheckDB(){

   String userName= uName.getText();
   String password=pwd.getText();
   Boolean loginStatus =false;
   try {
    String queryString = "SELECT username FROM User WHERE username='"+userName+"' AND
                                    password='" +password+"';";
    ResultSet rset = stmt.executeQuery(queryString);
    if (rset.next()) {
                        loginStatus=true;
                        }

     else {
                        loginStatus=false;
    }
   }
   catch (SQLException ex) {
    ex.printStackTrace();
   }
   return loginStatus;
}
```

```
void initializeDB()
{
        try {
            // 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/Week9", "root", "");
            System.out.println("Database connected");
            // Create a statement
            stmt = connection.createStatement();
          }
          catch (Exception ex) {
            ex.printStackTrace();
          }
      }
```

```
public static void main(String[] args)
{
login frame = new login();
frame.setSize(400, 100); // Set the frame size
frame.setLocationRelativeTo(null); // New since JDK 1.4
frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
frame.setVisible(true); // Display the frame
}
}
```

Exercise 3: There are two tables: Student table(firstName, mi, lastName, SID) and Grade table (SID, courseID, registeredDate, Grade). Write a program that lets the user enter the student ID and the course ID and displays firstName, middle name, last name and grade of that student.



```
public class FindGrade extends JFrame {
  private JTextField jtfSSN = new JTextField(9);
  private JTextField jtfCourseId = new JTextField();
  private JButton jbtShowGrade = new JButton("Show Grade");
  JTextArea jgrade=new JTextArea();
  Border titledBorder1 = new TitledBorder("Query Result");
  // Statement for executing queries
  private Statement stmt;
FindGrade() {
   // Initialize database connection and create a Statement object
   initializeDB();
   JScrollPane jScrollPane1 = new JScrollPane(jgrade);
   jScrollPane1.setBorder(titledBorder1);
   JPanel jPanel1 = new JPanel(new GridLayout(3,2));
   jPanel1.add(new JLabel("SSN"));
   jPanel1.add(jtfSSN);
   jPanel1.add(new JLabel("Course ID"));
   jPanel1.add(jtfCourseId);
   jPanel1.add(new JLabel(" "));
   jPanel1.add(jbtShowGrade);
   JPanel jPanel2=new JPanel(new GridLayout(1,1));
   jPanel2.add(jScrollPane1);

   add(jPanel1, BorderLayout.NORTH);
   add(jPanel2, BorderLayout.CENTER);
```

```
import javax.swing.*;
import javax.swing.border.Border;
import javax.swing.border.TitledBorder;
import java.sql.*;
import java.awt.*;
import java.awt.event.*;
```

```
jbtShowGrade.addActionListener(new ActionListener() {
  @Override
  public void actionPerformed(ActionEvent e) {

            String ssn = jtfSSN.getText();
            String row = " ";
            String courseId = jtfCourseId.getText();
            try {
              String queryString = "SELECT Student.firstName , Student.mi, Student.lastName, "
                            + "Grade.Grade FROM Student,Grade WHERE Student.SID="+ssn+" AND
                               Grade.CourseID="  +courseId+" AND Student.SID=Grade.SID;";

              ResultSet rset = stmt.executeQuery(queryString);
              if (rset.next()) {
                    row = " ";
                            for (int i = 1; i <= 4; i++) {
                                    // A non-String column is converted to a string
                            row += rset.getString(i) + "\t";
                            }
                            row += '\n';
                            }

               else {
               // Display result in a dialog box
                      jgrade.setText("Not found");
              }
              jgrade.setText(row);
            }
            catch (SQLException ex) {
              ex.printStackTrace();
            }
    }
  });
}
```

```
private void initializeDB() {
 try {
  // 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/Week9", "root", "");
  System.out.println("Database connected");
  // Create a statement
  stmt = connection.createStatement();
 }
 catch (Exception ex) {
  ex.printStackTrace();
 }
}
```

```
/** Main method */
public static void main(String[] args) {
 FindGrade frame = new FindGrade();
 frame.setDefaultCloseOperation(JFrame.EXIT_ON_CLOSE);
 frame.setTitle("Find Grades");
 frame.setSize(400,400);
 frame.setLocationRelativeTo(null); // Center the frame
 frame.setVisible(true);
 }
}
```

Next Week Lecture

- Java Applet and Networking