**Assignment 6**

What is JDBC?

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

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

Establish a connection:

* import java.sql.\*;
* Load the vendor specific driver
* Make the connection
* Connection con = DriverManager.getConnection("jdbc:oracle:thin:@oracle-prod:1521:OPROD",username, passwd);
* Establishes connection to database by obtaining a Connection object

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);

* String insertLehigh = "Insert into Lehigh values“

+ "(123456789,abc,100)";

stmt.executeUpdate(insertLehigh);

Get ResultSet:

String queryLehigh = "select \* from Lehigh";

ResultSet rs = Stmt.executeQuery(queryLehigh);

while (rs.next()) {

int ssn = rs.getInt("SSN");

String name = rs.getString("NAME");

int marks = rs.getInt("MARKS");

}

Close connection:

stmt.close();

con.close();

Transactions and JDBC:

* JDBC allows SQL statements to be grouped together into a single transaction
* Transaction control is performed by the Connection object, default mode is auto-commit, I.e., each sql statement is treated as a transaction
* We can turn off the auto-commit mode with con.setAutoCommit(false);
* And turn it back on with con.setAutoCommit(true);
* Once auto-commit is off, no SQL statement will be committed until an explicit is invoked con.commit();
* At this point all changes done by the SQL statements will be made permanent in the database.

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

Metadata from DB:

* A Connection's database is able to provide schema information

describing its tables, its supported SQL grammar, its stored procedures the capabilities of this connection, and so on

What is a stored procedure?

Group of SQL statements that form a logical unit

and perform a particular task

This information is made available through a DatabaseMetaData object.

Metadata from DB – example:

...

Connection con = .... ;

DatabaseMetaData dbmd = con.getMetaData();

String catalog = null;

String schema = null;

String table = “sys%”;

String[ ] types = null;

ResultSet rs = dbmd.getTables(catalog , schema , table , types );

JDBC and beyond:

* (JNDI) Java Naming and Directory Interface

1. API for network-wide sharing of information about users, machines, networks, services, and applications
2. Preserves Java’s object model

* (JDO) Java Data Object

1. Models persistence of objects, using RDBMS as repository
2. Save, load objects from RDBMS

* (SQLJ) Embedded SQL in Java

1. Standardized and optimized by Sybase, Oracle and IBM
2. Java extended with directives: # sql
3. SQL routines can invoke Java methods
4. Maps SQL types to Java classes