

TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN
KHOA CÔNG NGHỆ PHẦN MỀM

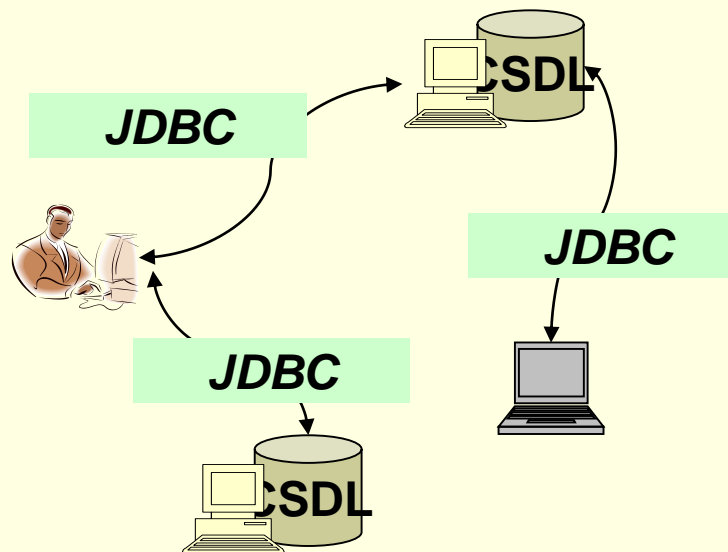
Chương 7:
LẬP TRÌNH JDBC
(JAVA DATABASE CONNECTIVITY)

NỘI DUNG

- ❖ Khái niệm cơ bản
- ❖ Kiến trúc JDBC & JDBC APIs
- ❖ Các bước làm việc với Database dùng JDBC
- ❖ Một số lớp và phương thức cơ bản trong JDBC API
- ❖ Các loại JDBC Drivers
- ❖ Ví dụ minh họa

Giới thiệu về JDBC

- JDBC (Java DataBase Connectivity) là một thư viện chuẩn dùng để truy xuất các cơ sở dữ liệu dạng bảng như MS Access, SQL Server, Oracle,... trong các ứng dụng Java bằng ngôn ngữ truy vấn SQL.
- Các hàm truy xuất cơ sở dữ liệu với JDBC nằm trong gói `java.sql.*`



Tại sao cần JDBC?

- ❖ JDBC giúp các Java Developers tạo nên các ứng dụng truy xuất cơ sở dữ liệu mà **không cần phải học và sử dụng các APIs do các công ty sản xuất phần mềm khác nhau bên thứ ba cung cấp.**
- ❖ JDBC đảm bảo rằng bạn sẽ có thể phát triển nên các ứng dụng truy cập cơ sở dữ liệu có **khả năng truy cập đến các RDBMS khác nhau** bằng cách sử dụng các JDBC driver khác nhau.

Có thể làm gì với JDBC?

- ❖ Kết nối với nguồn dữ liệu (database)
- ❖ Gửi các câu lệnh truy vấn, cập nhật dữ liệu đến csdl
- ❖ Truy vấn và xử lý kết quả

Kiến trúc JDBC



Các khái niệm cơ bản

- **JDBC API:** là một API hoàn toàn dựa trên Java.
- **JDBC DriverManager:** là trình quản lý JDBC giao tiếp trực tiếp với các trình điều khiển cơ sở dữ liệu cụ thể - giao tiếp thực sự với cơ sở dữ liệu.
- Các RDBMS hay các nhà sản xuất phần mềm thứ 3 phát triển các drivers cho java đều phải tuân thủ đặc tả JDBC.
- Các java developers dùng các JDBC drivers để phát triển các ứng dụng có truy cập, thao tác CSDL.

JDBC Drivers

- Tập các lớp giúp truy cập đến các hệ DBMS khác nhau dùng kỹ thuật JDBC
- Do các hãng xây dựng DBMS hoặc một đơn vị thứ 3 khác cung cấp.
- Gói trong .jar/.zip

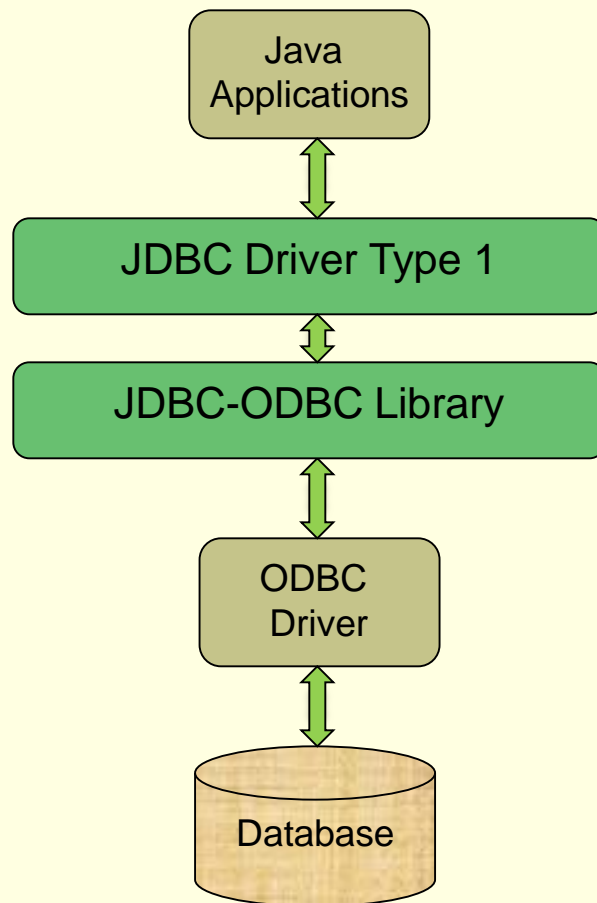
<https://www.codejava.net/java-se/jdbc/jdbc-driver-library-download>

Các loại JDBC Drivers

- *JDBC-ODBC Bridge plus ODBC Driver*
- *Java to Native API partly Java technology-enabled driver*
- *Pure Java Driver for Database Middleware*
- *Direct-to-Database Pure Java Driver*

Các loại JDBC Drivers

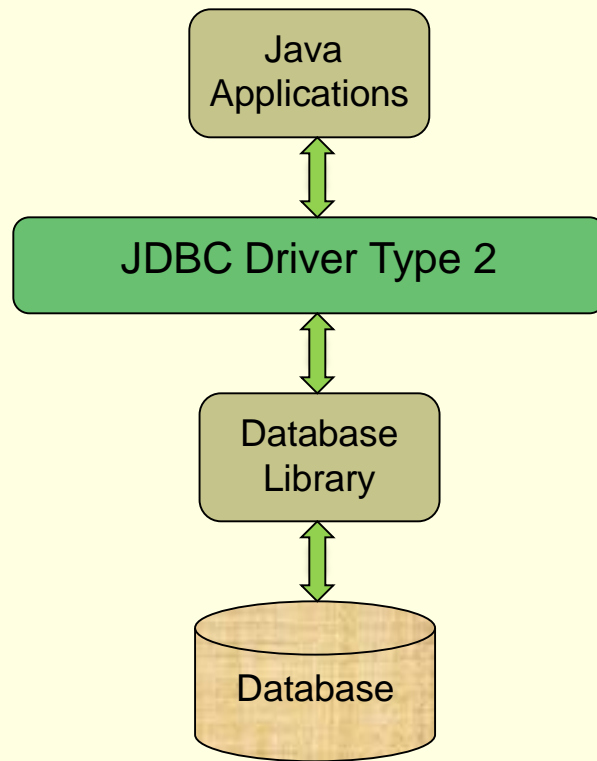
Type 1: JDBC-ODBC Bridge



- JDBC-ODBC ủy nhiệm công việc truy cập dữ liệu cho ODBC API. Chúng là trình điều khiển chậm nhất trong số còn lại.
- Phương thức truy xuất dữ liệu đòi hỏi trình điều khiển ODBC được cài đặt trên máy tính client → thiếu sự linh hoạt
- Sử dụng nếu DBMS không tương thích với các driver thuần Java

Các loại JDBC Drivers

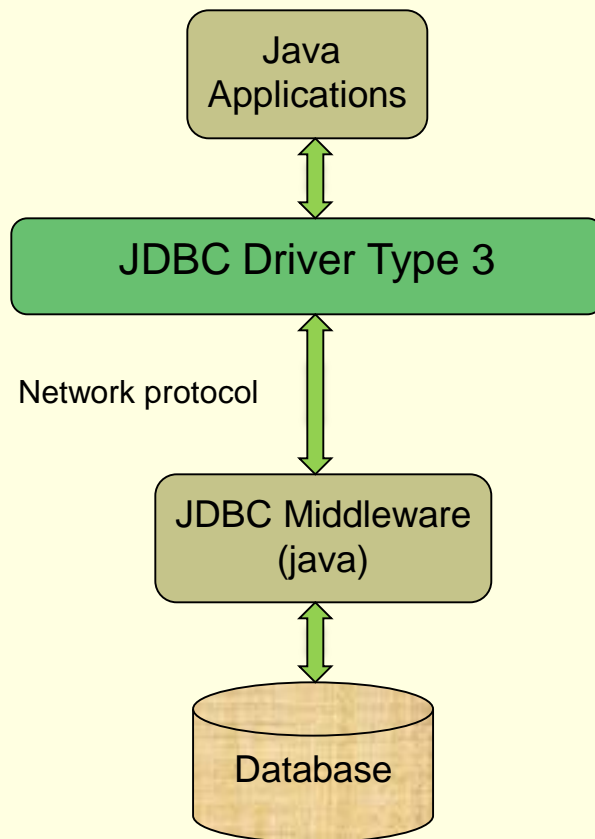
Type 2: Java to Native API



- Dùng Java Native API (JNI) để gọi đến các thư viện API của database.
- Các native library hỗ trợ kết nối với DBMS cụ thể phải được cài đặt trên client → thiếu sự linh hoạt

Các loại JDBC Drivers

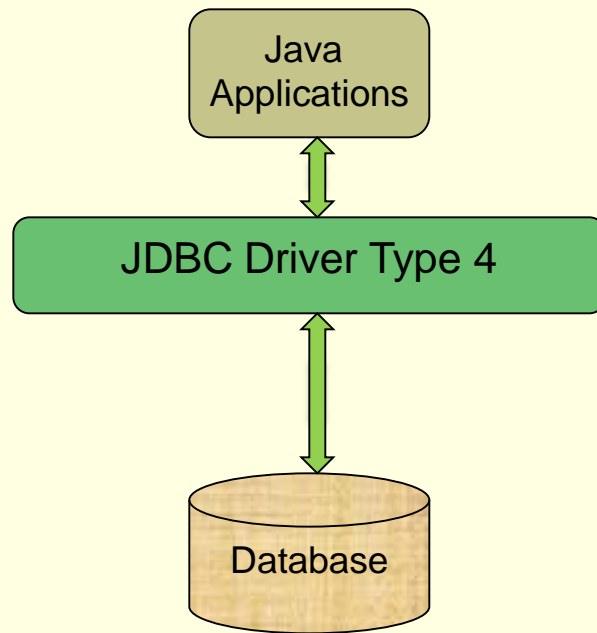
Type 3: Java to Network Protocol Or All - Java Driver



- Dùng database-independent protocol để giao tiếp với JDBC middleware server.
- Chuyển đổi các lời gọi JDBC thành giao thức mạng độc lập với bất kỳ giao thức DBMS đặc thù. Sau đó, một phần mềm trung gian (middleware) chạy trên máy server chuyển đổi giao thức mạng thành giao thức DBMS đặc thù.
- Sự chuyển này đặt ở phía server mà không đòi hỏi cài đặt trên máy tính client.

Các loại JDBC Drivers

Type 4: Java to Database Protocol



- Drivers thuần java và hiện thực/dùng network protocol để giao tiếp trực tiếp với CSDL cụ thể
- Client kết nối trực tiếp đến DBMS ➔ Là những JDBC drivers nhanh nhất

Cách nạp Database Driver?

Driver được cài đặt trong **JAR file**.

JAR phải được khai báo trong **classpath**:

1. Thêm *jar file* to vào IDE project

2. Thêm JAR file vào CLASSPATH

```
CLASSPATH = /my/path/mysql-connector.jar;.
```

3. Thêm JAR sử dụng Java command line:

```
java -cp /my/path/mysql-connector.jar ...
```

4. Đặt JAR file vào **JRE/lib/ext** directory:

```
C:/java/jre1.6.0/lib/ext/mysql-  
connector.jar
```

JDBC API

The screenshot displays the Java Platform Standard Ed. 6 API documentation in Mozilla Firefox. The browser window shows the file path `file:///C:/Sun/SDK/jdk/docs/api/index.html`. The page title is "Java™ Platform Standard Ed. 6". The navigation bar includes links for Overview, Package, Class, Use, Tree, Deprecated, Index, and Help. The main content area shows the "Package javax.sql" page, which provides the API for server side data source access and processing from the Java™ programming language. The page includes a "See: Description" link and an "Interface Summary" table.

Package javax.sql

Provides the API for server side data source access and processing from the Java™ programming language.

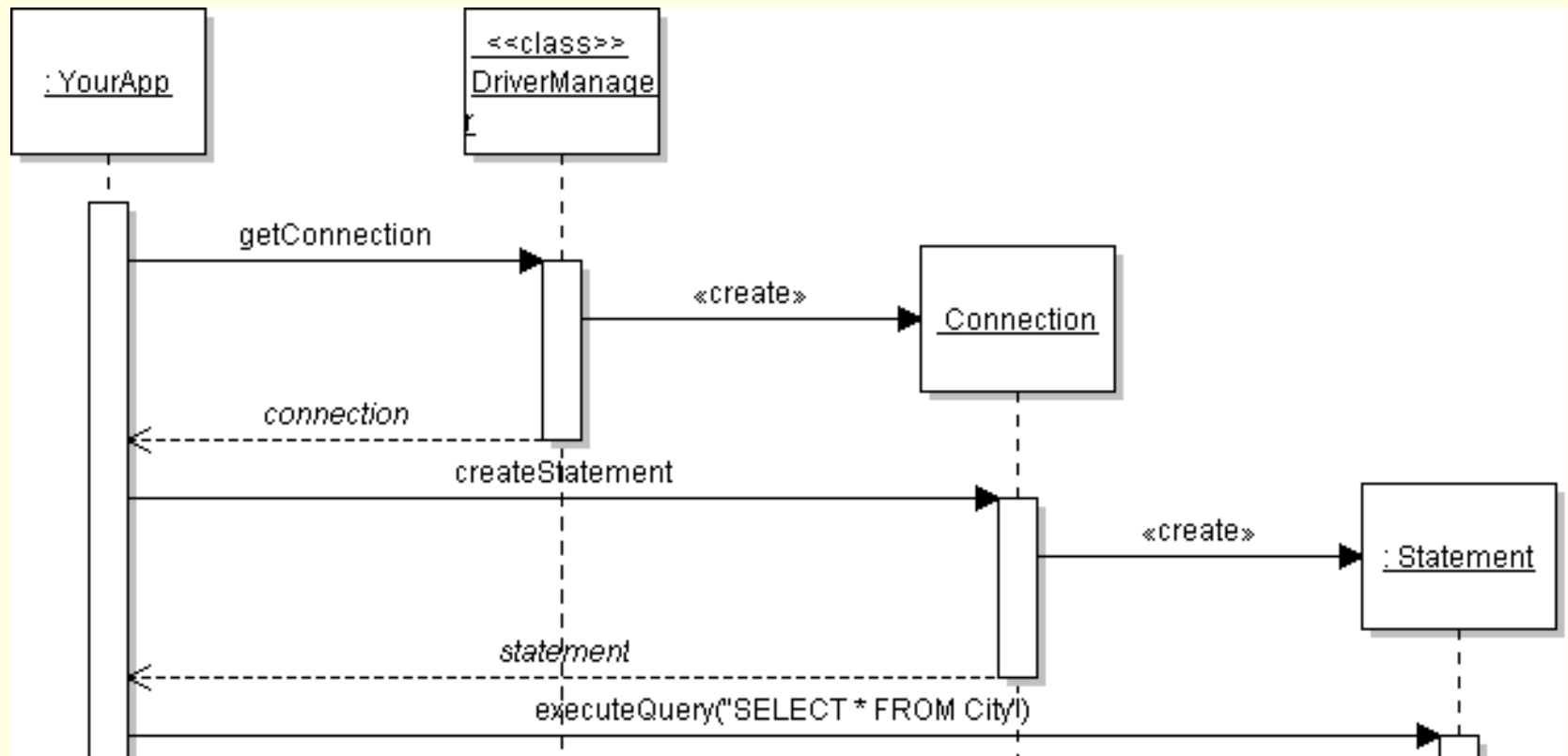
See: [Description](#)

Interface Summary	
CommonDataSource	Interface that defines the methods which are common between DataSource, XADataSource and ConnectionPoolDataSource.
ConnectionEventListener	An object that registers to be notified of events generated by a PooledConnection object.
ConnectionPoolDataSource	A factory for PooledConnection objects.
DataSource	A factory for connections to the physical data source that this

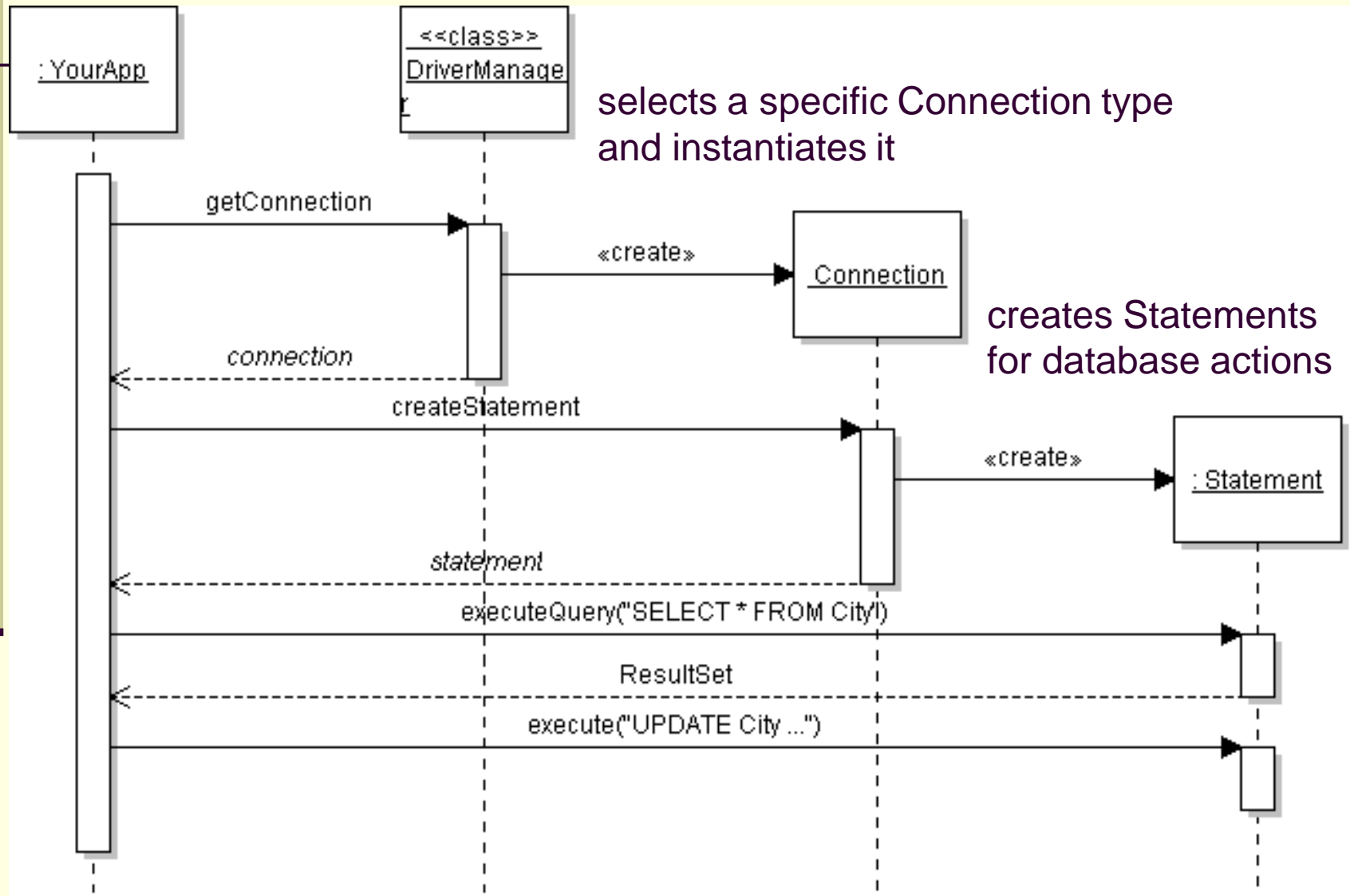
Các bước làm việc với CSDL

- ❖ Bước 1: Nạp JDBC driver
- ❖ Bước 2: Tạo kết nối với CSDL dùng driver đã nạp ở bước 1
- ❖ Bước 3: Thao tác với CSDL

Các bước làm việc với CSDL



Các bước làm việc với CSDL



Một số lớp và phương thức cơ bản

- ❖ ***DriverManager*** - Nạp các JDBC driver vào trong bộ nhớ. Có thể sử dụng nó để mở các kết nối tới một nguồn dữ liệu.
- ❖ ***Connection*** - Biểu thị một kết nối đến một nguồn dữ liệu. Được dùng để tạo ra các đối tượng Statement, PreparedStatement và CallableStatement.
- ❖ ***Statement*** - Biểu diễn một lệnh SQL tĩnh. Có thể sử dụng nó để thu về đối tượng ResultSet.
- ❖ ***PreparedStatement*** - Một giải pháp thay thế hoạt động tốt hơn đối tượng Statement, thực thi một câu lệnh SQL đã được biên dịch trước.
- ❖ Hướng dẫn tạo kết nối các database server với Netbean:
<https://netbeans.org/kb/docs/ide/java-db.html>

JDBC Code

```
static final String URL = "jdbc:mysql://dbserver/world";
static final String USER = "student";
static final String PASSWORD = "secret";

//Load driver

// 1. Get a Connection to the database.
Connection connection =
    DriverManager.getConnection( URL, USER, PASSWORD );

// 2. Create a Statement
Statement statement = connection.createStatement();

// 3. Execute the Statement with SQL command.
ResultSet rs = statement.executeQuery("SELECT * FROM ...");

// 4. Use the Result.
while ( rs.next( ) ) {
    String name = rs.getString("name");
}
```

Kết nối CSDL với JDBC

- **`java.sql.Connection`** là giao diện chuẩn để kết nối đến các DBMS.
- Mỗi DBMS sẽ phải cài đặt interface này.
 - MySQL driver
`mysql-connector-java-5.1.7-bin.jar`
 - Derby driver
- **`DriverManager`** dùng để chọn driver và thiết lập kết nối.

JDBC URL

- Chỉ định nguồn dữ liệu sẽ kết nối
jdbc:<subprotocol>:<dsn>:<others>

Trong đó:

- ***<subprotocol>***: được dùng để xác định trình điều khiển để kết nối với CSDL.
- ***<dsn>***: địa chỉ CSDL. Cú pháp của ***<dsn>*** phụ thuộc vào từng trình điều khiển cụ thể.
- ***<other>***: các tham số khác

Ví dụ:

- ***jdbc:odbc:dbname*** là URL để kết nối với CSDL tên dbname sử dụng cầu nối ODBC.
- ***jdbc:sqlserver://hostname:1433*** là URL để kết nối với CSDL Microsoft SQL Server. Trong đó hostname là tên máy cài SQL Server.

Ví dụ JDBC URL

RDBMS	Database URL format
MySQL	<code><i>jdbc:mysql://hostname:portNumber/databaseName</i></code>
ORACLE	<code><i>jdbc:oracle:thin:@hostname:portNumber:databaseName</i></code>
DB2	<code><i>jdbc:db2:hostname:portNumber/databaseName</i></code>
Java DB/Apache Derby	<code><i>jdbc:derby:dataBaseName</i></code> (embedded) <code><i>jdbc:derby://hostname:portNumber/databaseName</i></code> (network)
Microsoft SQL Server	<code><i>jdbc:sqlserver://hostname:portNumber;databaseName=dataBaseName</i></code>
Sybase	<code><i>jdbc:sybase:Tds:hostname:portNumber/databaseName</i></code>

Database URL

Định dạng chung của database URL:

```
String DB_URL = "jdbc:mysql://dbserver:3306/world";
```

Protocol Sub-protocol Hostname Port DatabaseName

- ❑ Port là TCP port mà hệ QTCSDL sử dụng để lắng nghe yêu cầu.
 - **3306** is the **default port** for MySQL
- ❑ Sử dụng "localhost" nếu CSDL nằm cùng 1 máy.

Database URL

Hostname và port là tùy chọn.

Đối với MySQL driver: **defaults** là localhost và port 3306

Ví dụ:

```
"jdbc:mysql://localhost:3306/world"
```

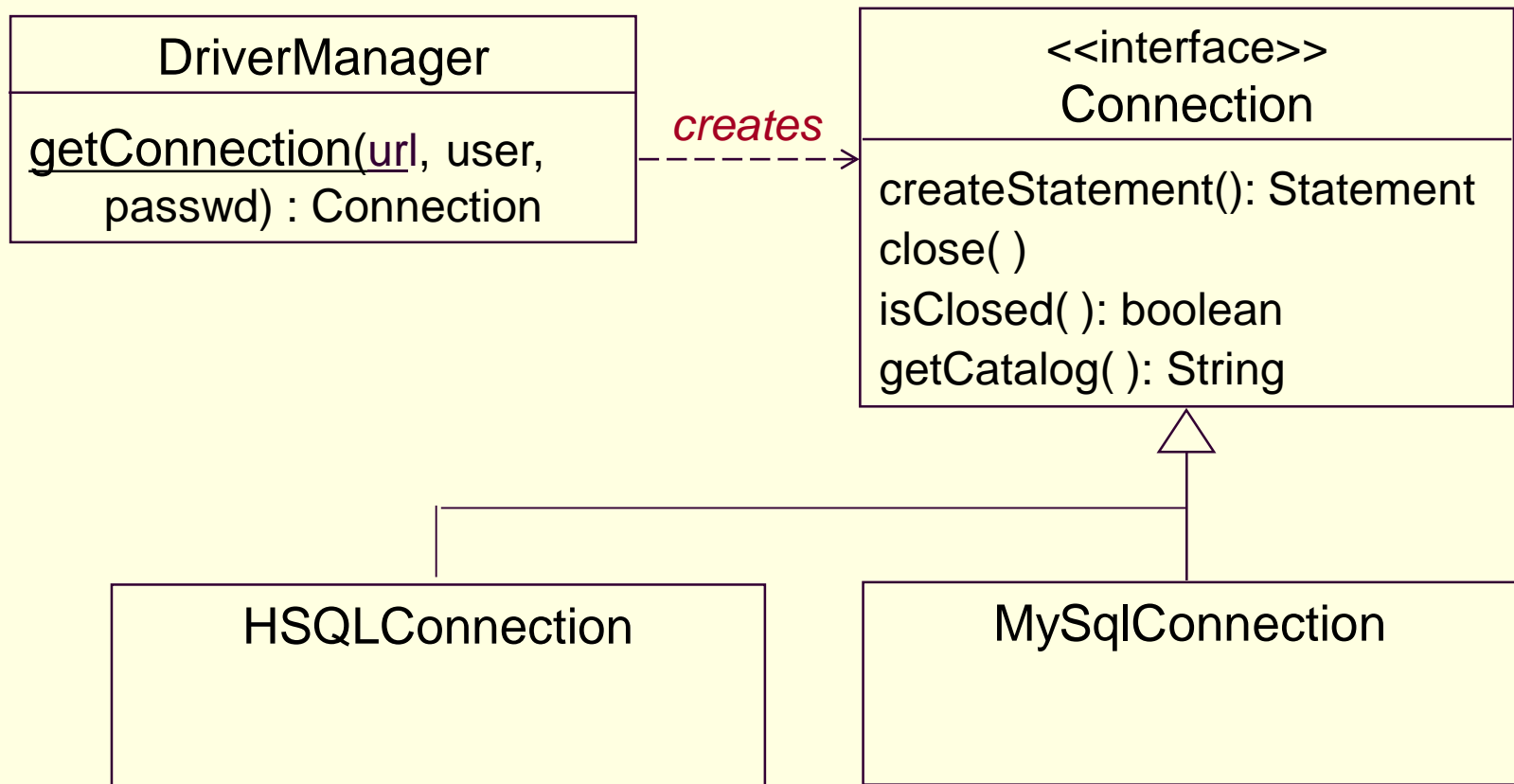
```
"jdbc:mysql://localhost/world"
```

```
"jdbc:mysql:///world"
```

```
"jdbc:mysql:/world"
```

Kết nối CSDL với JDBC

url = "jdbc:mysql://hostname/database"



Thi hành SQL Commands

- Để thi hành SQL command sử dụng phương thức *createStatement* của đối tượng Connection.
- **Statement** interface định nghĩa các phương thức để thi hành câu lệnh SQL.

```
Statement statement = connection.createStatement( );  
  
// execute an UPDATE command  
int count = statement.executeUpdate( "UPDATE City  
    SET population=30000 WHERE name='Bangsaen'" );  
  
System.out.println("Modified " + count + " records");
```

Thi hành câu lệnh SQL

- Câu lệnh `statement.executeQuery()` trả về 1 `ResultSet`.
- `ResultSet` là bảng chứa kết quả trả về của SQL.

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

// execute a SELECT command
ResultSet rs = statement.executeQuery(
    "SELECT * FROM city WHERE id = "+id );
rs.first(); // scroll to first result
do {
    String name = rs.getString(1);    // get by position
    int population = rs.getInt("population"); // by name
    ...
} while( rs.next() );
```

Ví dụ:

```
Scanner console = new Scanner(System.in);  
System.out.print("Name of city to find? ");  
String name = console.nextLine().trim();  
String query =  
    "SELECT * FROM city WHERE Name='" + name + "'";  
ResultSet rs = statement.executeQuery( query );
```

ResultSet Methods

- **ResultSet** chứa các "row" trả về từ câu query.
- **ResultSet** hỗ trợ các phương thức để lấy dữ liệu từ cột:
 - "get" by column number -- starts at 1 (not 0)!
 - "get" by column name -- field names in table/query.

```
String query = "SELECT * FROM Country WHERE ...";  
ResultSet rs = statement.executeQuery( query );
```

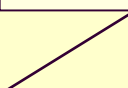
```
// go to first row of results  
rs.first( );
```

```
// display the values
```

```
System.out.println( rs.getString( 1 ) );
```

```
System.out.println( rs.getInt( "population" ) );
```

get by column number



get by name

ResultSet Methods

- ResultSet hỗ trợ các phương thức để lấy từng dòng và cột trong kết quả trả về

ResultSet

next() : boolean

go to next row of results. "false" if no more.

previous() : boolean

go to previous row. "false" if 1st result.

first() : boolean

go to first row of results.

last() : boolean

go to last row of results.

absolute(k)

go to k-th row of results.

getInt(name: String)

get int value of field "name"

getInt(index: int)

get int value of k-th column in a record

...

ResultSet Methods for Getting Data

ResultSet "get" methods return column data:

`getLong(3)` : get by column index (most efficient)

`getLong("population")` : get by field name (safest)

<code>getInt()</code>	<code>getLong()</code>	- get Integer field value
<code>getFloat()</code>	<code>getDouble()</code>	- get floating pt. value
<code>getString()</code>		- get Char or Varchar field value
<code>getDate()</code>		- get Date or Timestamp field value
<code>getBoolean()</code>		- get a Bit field value
<code>getBytes()</code>		- get Binary data
<code>getBigDecimal()</code>		- get Decimal field as BigDecimal
<code>getBlob()</code>		- get Binary Large Object
<code>getObject()</code>		- get any field value

Thi hành câu lệnh SQL Commands

Statement interface định nghĩa nhiều phương thức

```
ResultSet rs =  
    statement.executeQuery("SELECT ...");  
    ■ use for statements that return data values (SELECT)  
  
int count =  
    statement.executeUpdate("UPDATE ...");  
    ■ use for INSERT, UPDATE, and DELETE  
  
boolean b =  
    statement.execute("DROP TABLE test");  
    ■ use to execute any SQL statement(s)
```

Các bước làm việc với CSDL

❖ Thao tác với CSDL:

```
Statement stmt = conn.createStatement();
try {
    ResultSet rs = stmt.executeQuery( "SELECT * FROM MyTable" );
    try {
        while ( rs.next() ) {
            int numColumns = rs.getMetaData().getColumnCount();
            for ( int i = 1 ; i <= numColumns ; i++ ) {
                // Column numbers start at 1.
                // Also there are many methods on the result set to return
                // the column as a particular type. Refer to the Sun documentation
                // for the list of valid conversions.
                System.out.println( "COLUMN " + i + " = " + rs.getObject(i) );
            }
        }
    } finally {
        rs.close();
    }
} finally {
    stmt.close();
}
```

Ví dụ minh họa – JDBC ODBC

...

Connection *myCon;*

Statement *myStatement;*

ResultSet *myResultSet;*

String *sUsername, sPassword;*

try {

myCon = DriverManager.getConnection("jdbc:odbc:ThuchanhJ2EE", "", "");

myStatement = myCon.createStatement();

*myResultSet = myStatement.executeQuery("Select * from Account");*

Ví dụ minh họa - JDBC ODBC

```
while (myResultSet.next())    {  
    sUsername = myResultSet.getString(1);  
    sPassword = myResultSet.getString(2);  
    if (sUsername.equals("admin") && sPassword.equals("admin"))  
        return true;  
}  
myResultSet.close(); myStatement.close(); myCon.close();  
}  
catch(Exception e) {  
    System.out.println(e.toString());  
}
```

JDBC Driver SQL Server

```
...  
Connection      myCon;  
  
Try {  
    // JDBC Driver for SQL Server 2000  
  
    myCon =  
    DriverManager.getConnection("jdbc:sqlserver://localhost;databaseName=QLNV;  
    User=sa; Password=sa");  
  
    ...  
}  
  
catch (Exception e) {  
  
    ...  
}
```

PreparedStatement

- Đối tượng PreparedStatement chứa 1 câu lệnh SQL đã được biên dịch trước.
 - Khi thực thi DBMS không cần phải biên dịch câu lệnh SQL.
- Thường được dùng với các câu lệnh SQL có tham số.

PreparedStatement

- Được tạo ra từ đối tượng Connection.
- Ví dụ đối tượng PreparedStatement có chứa 2 tham số:

```
“SELECT lastName, firstName, title ” +  
“FROM authors, titles, authorISBN ” +  
“WHERE authors.authorID = authorISBN.authorID ” +  
    “AND titles.ISBN = authorISBN.isbn AND ” +  
    “lastName = ? AND firstName = ?” );
```

Cung cấp giá trị cho tham số của PreparedStatement

- Trước khi thi hành, chúng ta cần cung cấp giá trị cho tham số trong đối tượng PreparedStatement.
- Thực hiện thông qua các phương thức setXXX.
 - `authorBooks.setString(1, “Deitel”);`
 - `authorBooks.setString(2, “Paul”);`

Ví dụ

```
PreparedStatement updateSales;  
String updateString = "update COFFEES " +  
    "set SALES = ? where COF_NAME like ?";  
updateSales = con.prepareStatement(updateString);  
int [] salesForWeek = {175, 150, 60, 155, 90};  
String [] coffees = {"Colombian", "French_Roast", "Espresso",  
    "Colombian_Decaf", "French_Roast_Decaf"};  
int len = coffees.length;  
for(int i = 0; i < len; i++) {  
    updateSales.setInt(1, salesForWeek[i]);  
    updateSales.setString(2, coffees[i]);  
    updateSales.executeUpdate();  
}
```

Lớp Quản Lý DB Connection

Tạo lớp ConnectionManager with a static factory method

ConnectionManager

- connection : Connection

+getConnection() : Connection

+close() : void

```
// example how to use
```

```
Statement statement =
```

```
    ConnectionManager.getConnection().createStatement();
```

Simple version of manager (1)

```
public class ConnectionManager {  
    // literal constants in Java code is baaaad code.  
    // we will change to a configuration file later.  
    private static String driver = "com.mysql.jdbc.Driver";  
    private static String url = "jdbc:mysql://hostname/world";  
    private static String user = "student";  
    private static String password = "student";  
    /* a single shared database connection */  
    private static Connection connection = null;  
  
    private ConnectionManager() { /* no object creation */ }
```

Simple version of ConnectionManager (2)

```
/* the public accessor uses lazy instantiation */  
public static Connection getConnection( ) throws ... {  
    if ( connection == null ) connection = makeConnection();  
    return connection;  
}
```

Simple version of ConnectionManager (2)

```
private static Connection makeConnection( )
                                throws SQLException {
    try {
        // load the database driver class
        connection = DriverManager.getConnection(
                                url, user, password );
    } catch ( FileNotFoundException ex ) {
        logger.error("connection error", ex ); // Logging
        throw new SQLException( ex );
    }
}

/* the public accessor uses lazy instantiation */
public static Connection getConnection( ) throws ... {
    if ( connection == null ) connection = makeConnection();
    return connection;
}
```

ResultSet

- ResultSet gắn kết với 1 statement và 1 connection.
 - Nếu statement or connection bị đóng, kết quả sẽ mất
 - Nếu thi hành câu query khác, kết quả mất
- ResultSet thay đổi sau khi thi hành câu query
- ResultSet có thể cập nhật database

```
Statement stmt = connection.createStatement(  
    ResultSet.TYPE_SCROLL_SENSITIVE,  
    ResultSet.CONCUR_UPDATABLE );  
ResultSet rs = statement.executeQuery( query );
```

ResultSet cập nhật database

- Xác định thuộc tính `ResultSet.CONCUR_UPDATABLE` khi tạo Statement.
- Đòi hỏi sự hỗ trợ của database driver

```
// rs is scrollable, will not show changes made
// by others, and will be updatable
Statement statement = connection.createStatement(
    ResultSet.TYPE_SCROLL_INSENSITIVE,
    ResultSet.CONCUR_UPDATABLE );
ResultSet rs = statement.executeQuery(query);
rs.next();
int population = rs.getInt("population");
// add 10,000 to the population
rs.updateInt( "population", population+10000 );
rs.updateRow( );
```

JTable

- Swing object hiển thị dữ liệu dưới dạng bảng.

The screenshot shows a Java Swing window titled "JTable SQL Query Browser". It contains a "Database Info" section with fields for Hostname, Database, User, and Password. Below this is a table with 5 columns: name, continent, region, population, and GNP. The table lists various countries and their corresponding data. At the bottom, there is a "Query:" field with a text input containing a SQL query, and a "Submit" button.

Database Info

Hostname: se Database: world User: student Passwd: *****

name	continent	region	population	GNP
Afghanistan	Asia	Southern and Central ...	22720000	5,976
Netherlands	Europe	Western Europe	15864000	371,362
Netherlands Antilles	North America	Caribbean	217000	1,941
Albania	Europe	Southern Europe	3401200	3,205
Algeria	Africa	Northern Africa	31471000	49,982
American Samoa	Oceania	Polynesia	68000	33
Andorra	Europe	Southern Europe	78000	1,630
Angola	Africa	Central Africa	12878000	6,648
Anguilla	North America	Caribbean	8000	63.2
Antigua and Barbuda	North America	Caribbean	60000	61.2

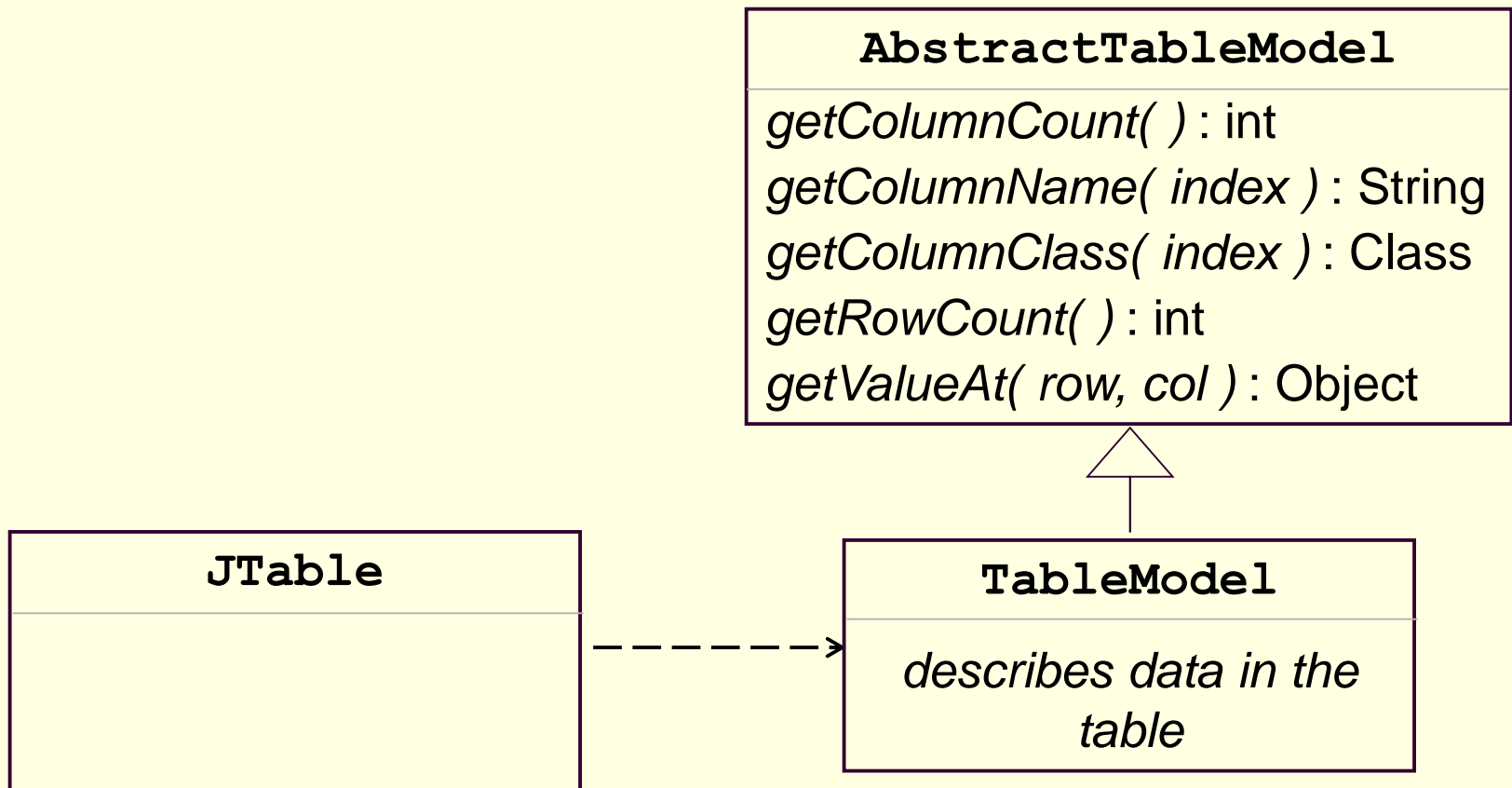
Query: select name, continent, region, population, GNP from Country

Submit

A JTable

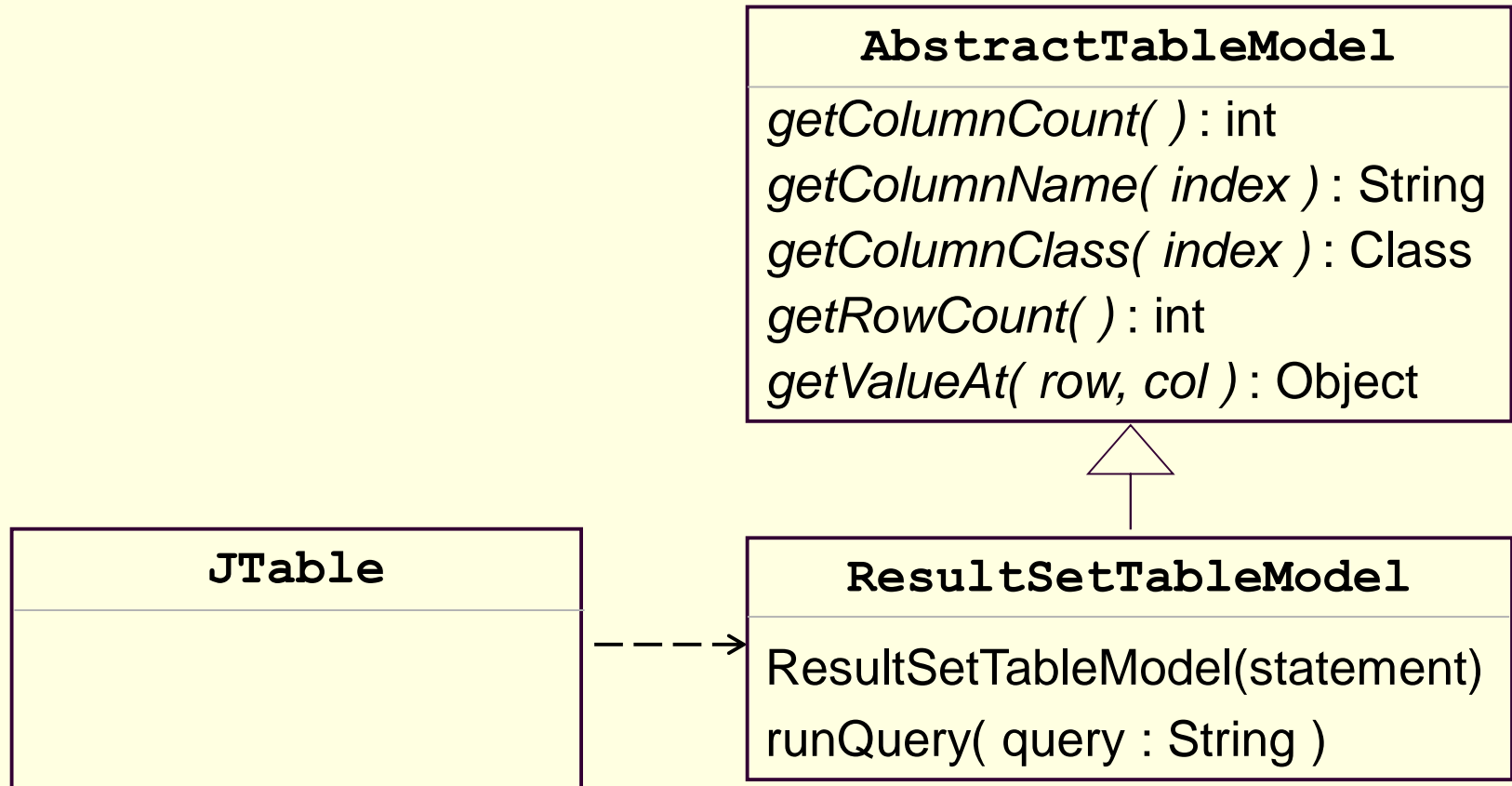
JTable Class Diagram

- JTable hiển thị kết quả trả về bởi TableModel.



Design a TableModel for Queries

- Design a TableModel to manage a ResultSet



Cài đặt TableModel

- **ResultSet** chưa dữ liệu cần hiển thị.

```
class ResultSetTableModel extends AbstractTableModel {
    private Statement statement;
    private ResultSet rs;

    public Object getValueAt(int row, int col) {
        if ( rs == null ) return null;
        rs.absolute( row + 1 );
        rs.getObject( col );
    }

    public int getRowCount() {
        if ( rs == null ) return 0;
        rs.last(); // move to last row
        rowCount = rs.getRow();
        return rowCount;
    }
}
```

Implementing TableModel (2)

- **ResultSet** is missing some information.

```
public int getColumnCount( ) {  
  
}  
  
public String getColumnName( int col ) {  
  
  
  
}
```

ResultSet Meta-data

- **ResultSet** có **getMetaData ()** trả về các thông tin.
- **ResultSetMetaData** chứa thông tin miêu tả.

```
try {  
    ResultSet resultSet = statement.executeQuery( query );  
    ResultSetMetaData metadata = resultSet.getMetaData();  
  
    int numberOfColumns = metadata.getColumnCount();  
  
    for(int col=1; col<=numberOfColumns; col++) {  
        // get name and SQL datatype for each column  
        String name = metadata洗getColumnName( col );  
        int type = metadata洗getColumnType( col );  
        int typeName = metadata洗getColumnTypeName( col );  
    } catch( SQLException sqle ) { ... }
```

Closing the Connection

- Khuyến cáo nên đóng connection sau khi hoàn tất

```
Connection connection = DriverManager.getConnection(...);  
/* use the database */  
...  
/* done using database */  
public void close( ) {  
    if ( connection == null ) return;  
    try {  
        connection.close();  
    }  
    catch ( SQLException sqle ) { /* ignore it */ }  
    finally { connection = null; }  
}
```