

## 객체 생성

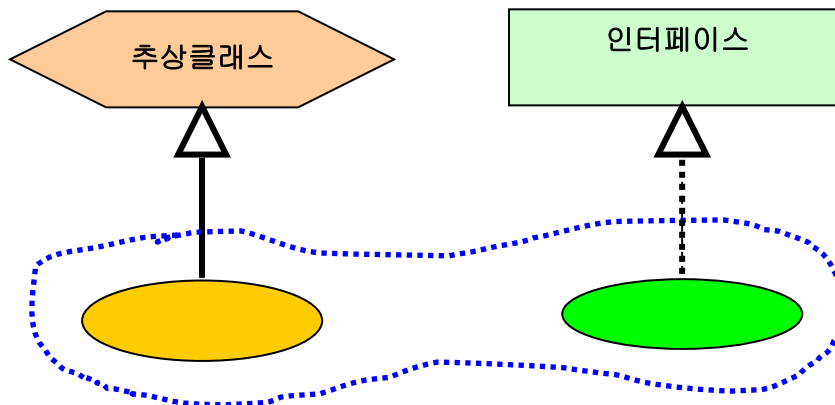
일반 클래스의 객체 생성은 new 와 생성자 메서드를 사용한다. new 와 생성자 메서드를 사용하는 객체 생성은 Java 구문에서 지원되는 일반적인 객체 생성 방법이다.

그러나 클래스들 중에는 이 일반적인 객체 생성을 통해서 객체를 생성할 수 없는 것들도 있다. 이런 경우에는 해당 클래스에서 제공되는 static 형 메서드를 호출하여 객체 생성을 대신 할 수 있다.

그렇다면 왜 생성자 메서드를 통해서 객체를 생성하지 않고 해당 클래스에서 제공되는 static 형 메서드를 호출하여 객체 생성을 대신하도록 하는 것일까?

- 여러 이유로 자식 클래스의 객체 생성을 대신 하여 사용되도록 하려는 경우
- 클래스의 객체 생성을 여러 번 하더라도 하나의 객체만을 생성하려는 경우

추상 클래스와 인터페이스의 경우에는 new 와 생성자 메서드를 사용하여 객체를 생성할 수 없다. 자식 클래스를 만들어 대신 객체 생성하여 사용한다.



JDBC 의 경우 대부분의 API 가 인터페이스이다.

Connection, Statement, ResultSet, PreparedStatement .....

대부분의 JDBC 프로그램에서는 위의 API 들에서 제공되는 메서드를 호출해야 한다.

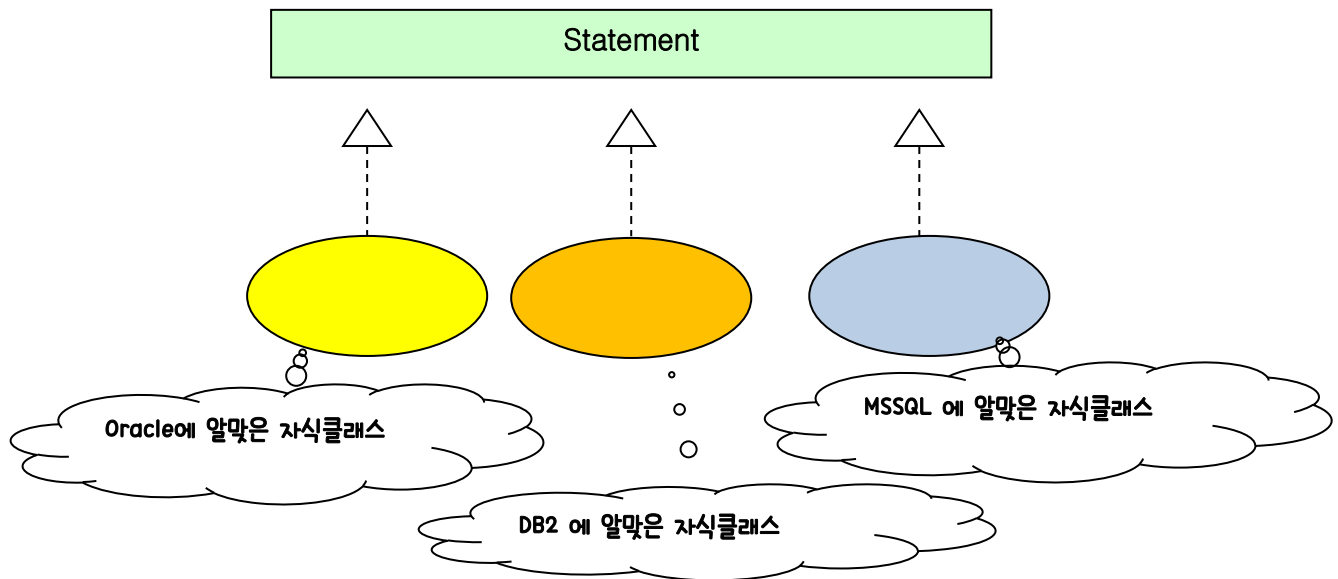
예를 들어.....

Connection : createStatement(), getMetaData().....

Statement : executeQuery(), executeUpdate().....

ResultSet : next(), getXXX().....

이 API 들의 객체를 생성하기 위해서는 이 API 들을 상속하여 구현하고 있는 자식 클래스가 필요한데 그 자식 클래스들을 바로 JDBC 드라이버가 제공한다. JDBC 드라이버라는 것은 JDBC 에서 인터페이스를 설계되어 있는 API 들의 자식 클래스들을 제공하는 프로그램이라고 할 수 있다. 즉 어떠한 DB 서버용 드라이버냐에 따라서 제공되는 자식 클래스들의 수행 코드가 다르게 만들어져 있는 것이다. JDBC API 내에서는 JDBC 드라이버가 제공하는 각 인터페이스들의 자식 클래스가 어떠한 이름의 클래스인지 모르고도 프로그래밍 가능하도록 팩토리 메서드라는 것을 제공하고 있다. 일반 메서드로서 다른 클래스의 객체생성을 대신해주는 메서드를 팩토리 메서드라고 한다.



JDBC는 Java 프로그램에서 DB 서버를 접속하여 데이터를 처리하는 기능을 구현하고자 할 때 사용되는 Java 기술

#### - JDBC 기술의 구성

- JDBC API (java.sql, javax.sql) -> 공통적(모든 DB 서버에 대해)
- JDBC Driver -> DB 서버마다 달라진다.

#### - 주요 JDBC API

##### [ 인터페이스 ]

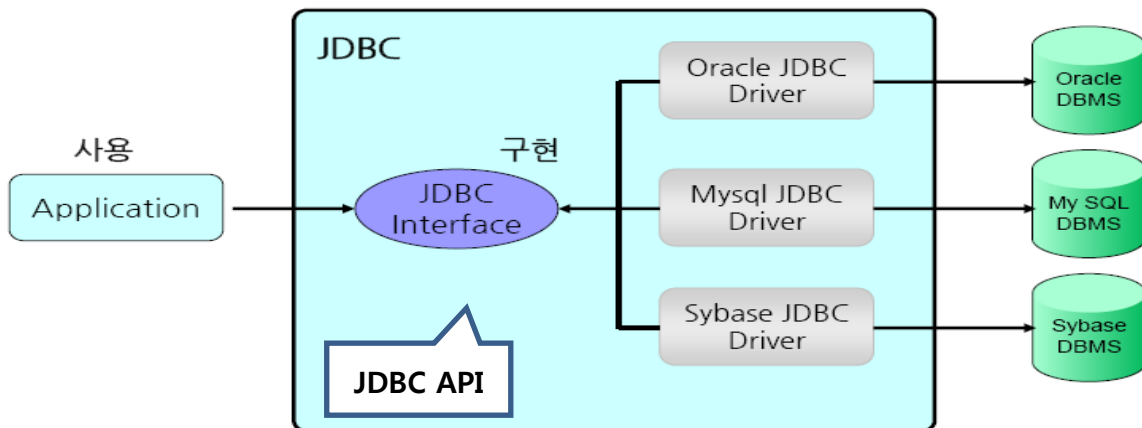
Connection, Statement, PreparedStatement, ResultSet  
DatabaseMetaData, ResultSetMetaData

##### [ 클래스 ]

DriverManager, Date, Time, Timestamp

## ❖ JDBC의 개념

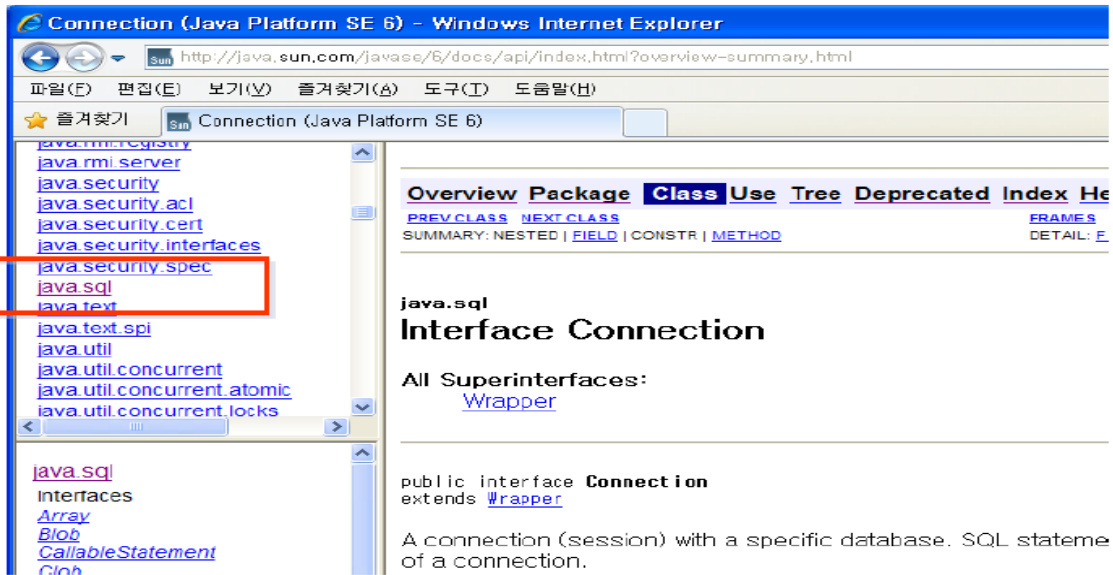
- 자바 언어에서 Database에 접근할 수 있게 해주는 Programming API



```
// 파일명 : ConnectDB.java
import java.sql.*;
class ConnectDB {
    public static void main (String args []) {
        try {
            Class.forName("oracle.jdbc.driver.OracleDriver");
            Connection conn = DriverManager.getConnection
                ("JDBCURL문자열", "계정", "암호");
            DatabaseMetaData meta=conn.getMetaData();

            System.out.print("Database: "+meta.getDatabaseProductName());
            System.out.println(" version "+meta.getDatabaseProductVersion());
            System.out.println("User name: "+meta.getUserName());
            conn.close();
        } catch (Exception ex) {
            System.out.println(ex);
        }
    }
}
```

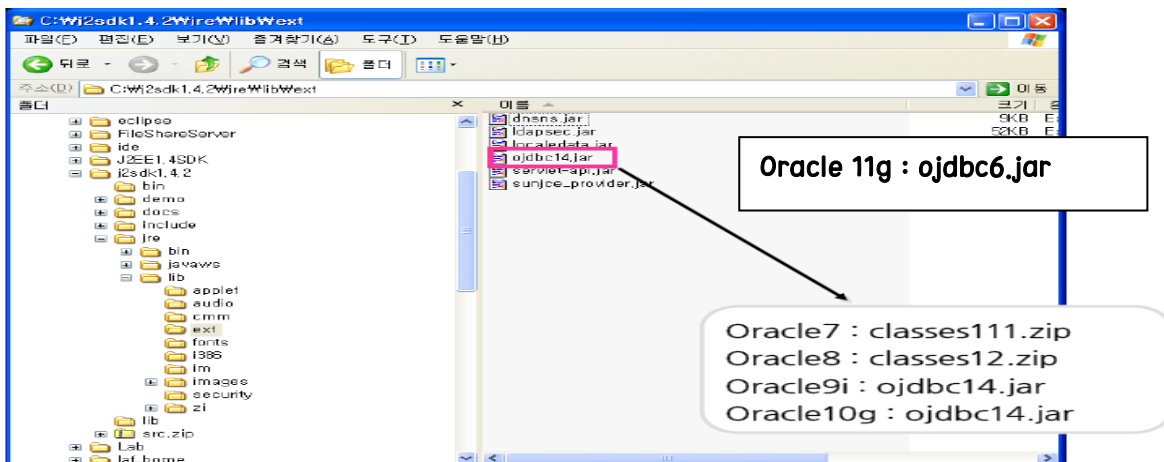
**JDBC API** JDBC 프로그래밍에 사용되는 API 로 java.sql 과 javax.sql 패키지로 구성



## JDBC driver 설치

JDBC driver 인 `ojdbc.jar` 를 개발툴(이클립스)에서 인식하도록 설정하거나

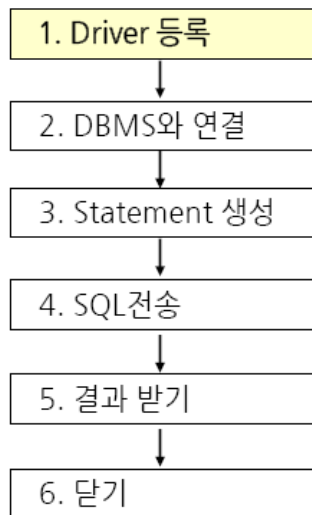
❖ `JAVA_HOME\jre\lib\ext` 에 driver를 추가해야 함 : `ojdbc14.jar`



## JDBC 프로그래밍절차



## 1. DriverManager에 해당 DBMS Driver 등록

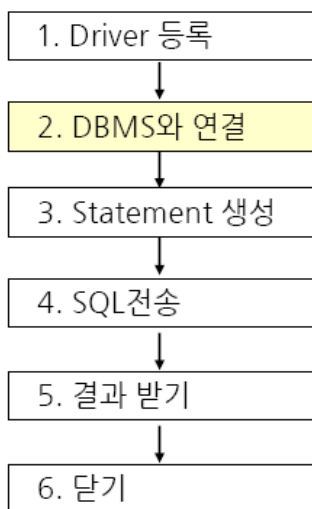


DriverManager

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

```
cf) Class.forName( "com.microsoft.jdbc.sqlserver.SQLServerDriver" );  
Class.forName( "org.gjt.mm.mysql.Driver" );
```

## 2. 해당 Driver로부터 Connection instance를 획득

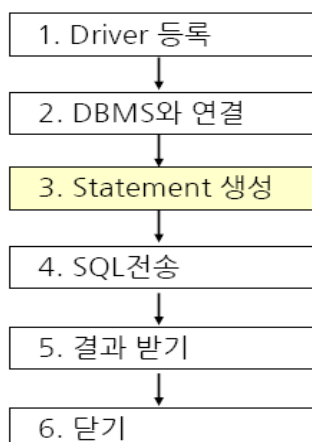


DriverManager Connection

```
public static Connection getConnection( String url,  
                                       String user,  
                                       String password )  
throws SQLException
```

```
Connection conn =  
    DriverManager.getConnection(  
        "jdbc:oracle:thin:@localhost:1521:xe",  
        "user",  
        "password" );
```

## 3. Connection instance로부터 Statement instance 획득

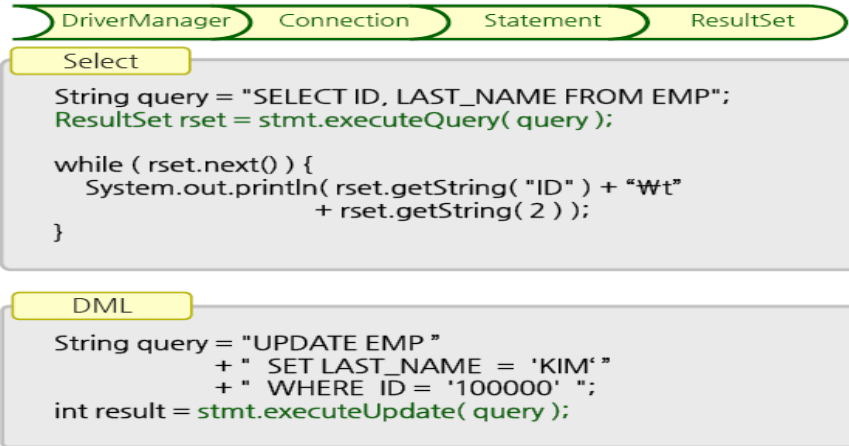
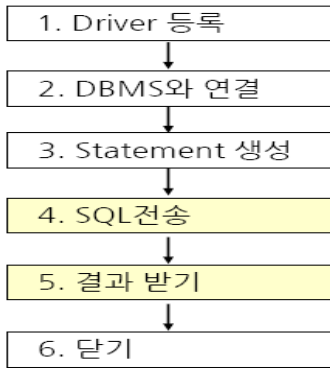


DriverManager Connection Statement

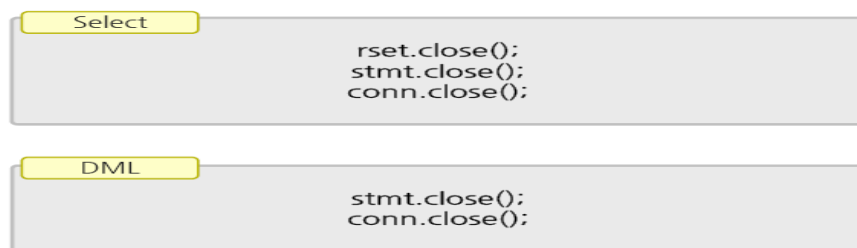
```
Statement stmt = conn.createStatement();
```

#### 4. Statement method를 이용하여 SQL 실행

#### 5. 실행 후 결과를 ResultSet(SELECT) 혹은 int형 변수(DML)로 받아 처리



#### 6. 사용한 자원 반납



#### [ 테이블 생성과 삭제 예제 ]

```

import java.sql.*;

public class CreateTable {

    public static void main (String args []) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
        Statement stmt = conn.createStatement();
        if (args.length == 1 && args[0].equals("create")) {
            stmt.executeUpdate("CREATE TABLE product " +
                "(id char(5), classid char(2), name varchar(50),balance int, price float)");
            System.out.println("테이블이 생성되었습니다.");
        } else {
            stmt.executeUpdate("DROP TABLE product");
            System.out.println("테이블이 삭제되었습니다.");
        }
        stmt.close();
        conn.close();
    }
}
  
```


## [ 테이블에 데이터를 삽입하는 예제 ]

```
import java.sql.*;

public class InsertTable {
    public static void main (String args []) throws Exception {
        try {
            // JDBC 드라이버를 로드한다.
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 데이터베이스에 접속한다. 적절한 JDBC URL 설정한다.
            Connection conn = DriverManager.getConnection( "JDBCURL문자열", "계정", "암호");
            Statement stmt = conn.createStatement();
            stmt.executeUpdate("INSERT into product values('00001', '10', '자바 프로그래밍',
                50, 16500)");
            stmt.executeUpdate("INSERT into product values ('00002', '10', 'J2EE 구현 패턴',
                40, 12000)");
            stmt.executeUpdate("INSERT into product values ('00003', '10', 'JSP 2.0', 60, 18000)");
            System.out.println("데이터들을 추가하였습니다.");
            stmt.close();
            conn.close();
        } catch (SQLException e) {
            System.out.println("오류 발생 : " + e);
        }
    }
}
```

## SELECT 명령의 결과 집합과 ResultSet 객체

```
ResultSet rs = stmt.executeQuery("SELECT id, last_name from emp");
while (rs.next()) {
    System.out.print(rs.getString("id") + "   ");
    System.out.print(rs.getString(2) + "   ");
}
```



BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

rs.next() // true 리턴  
String id = rset.getString( "ID" );  
String lastName = rset.getString( 2 );

	1	2
BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

ID = 10001  
LAST\_NAME = BOSS

rs.next() // true 리턴  
String id = rset.getString( "ID" );  
String lastName = rset.getString( 2 );

	1	2
BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

ID = 10002  
LAST\_NAME = JACKSON

rs.next() // false 리턴

	1	2
BOF	ID	LAST_NAME
ROW 1	10001	BOSS
ROW 2	10002	JACKSON
ROW 3	10003	HITE
...	...	...
EOF		

## [ 데이터 추출 예제 ]

```
import java.sql.*;
public class SelectTable {
    public static void main (String args []) throws Exception {
        Class.forName("oracle.jdbc.driver.OracleDriver");
        Connection conn =
            DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
        Statement stmt = conn.createStatement();
        ResultSet rs = stmt.executeQuery("SELECT * from product");
        while (rs.next()) {
            System.out.print(rs.getString("id") + "  ");
            System.out.print(rs.getString("classid") + "  ");
            System.out.print(rs.getString("name") + "  ");
            System.out.print(rs.getInt("balance") + "  ");
            System.out.println(rs.getFloat("price"));
        }
        rs.close();
        stmt.close();
        conn.close();
    }
}
```



## ❖Exception Handling 로직 추가

```
try {
    // 1. DBMS Driver 로딩
    Class.forName( "oracle.jdbc.driver.OracleDriver" );
    ...
    // 4. SQL 실행
    String query = "SELECT ID, LAST_NAME FROM EMP";
    rs = stmt.executeQuery( query);
    .....
} catch( ClassNotFoundException ce){
    ce.printStackTrace();
} catch( SQLException se){
    se.printStackTrace();
} finally {
    // 6. 사용할 Resource 반납
    try {
        rs.close();
        stmt.close();
        conn.close();
    } catch ( SQLException e ) {
        e.printStackTrace();
    }
}
```

## PreparedStatement 활용

```
public static void main( String[] args ) {
    Connection conn = null;
    PreparedStatement pstmt = null;
    String url = "
        JDBC URL 문자열
    ";
    try {
        Class.forName( "oracle.jdbc.driver.OracleDriver" );
        conn = DriverManager.getConnection( url , "student01" , "student01" );
        conn.setAutoCommit( false );

        String query = "UPDATE EMP SET LAST_NAME = ? WHERE ID = ? ";
        pstmt = conn.prepareStatement( query );
        pstmt.setString( 1, "HITE2" );
        pstmt.setString( 2, "10005" );
        pstmt.executeUpdate();
        conn.commit();
    } catch( ClassNotFoundException ce){
        .....
    } finally {
        .....
        pstmt.close();
        .....
    }
}
```

```
Statement stmt = conn.createStatement();
stmt.executeUpdate("insert into account values('"+name+"','"+ passwd+"')");
```

```
PreparedStatement pstmt = conn.prepareStatement("insert into account values (?, ?)");
pstmt.setString(1, name);
pstmt.setString(2, passwd);
pstmt.executeUpdate();
```

## [ 데이터 수정 예제 ]

```
import java.sql.*;

public class UpdateTable {

    public static void main (String args []) throws Exception {

        // JDBC 드라이버를 로드한다.

        Class.forName("oracle.jdbc.driver.OracleDriver");

        // 데이터베이스에 접속한다. 적절한 JDBC URL 설정한다.

        Connection conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");

        PreparedStatement pstmt = conn.prepareStatement(

            "UPDATE product SET balance=100 WHERE id = ?");

        pstmt.setString(1, args[0]);

        int update_su = pstmt.executeUpdate();

        System.out.println("수정된 레코드 수 : " + update_su);

        System.out.println("상품번호가 " +args[0] +"인 상품의 balance 필드 수정완료");

        pstmt.close();

        conn.close();

    }

}
```

## [ 데이터 삭제 예제 ]

```
import java.sql.*;

public class DeleteTable {

    public static void main (String args []) throws Exception {

        // DB 서버에 접속하는 기능을 추가합니다.

        :

        PreparedStatement pstmt = conn.prepareStatement(

            "DELETE from product WHERE balance < ?");

        pstmt.setInt(1, Integer.parseInt(args[0]));

        int del_su = pstmt.executeUpdate();

        System.out.println("재고가 " +args[0] +"이하인 상품 정보를 "+ del_su +

            "개 삭제하였습니다.");

        pstmt.close();

        conn.close();

    }

}
```

## [ 날짜 데이터 입력하고 추출하는 다양한 예제들 ]

```
Connection conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
PreparedStatement pstmt = conn.prepareStatement("INSERT INTO datetest_t VALUES (?, ?)");
pstmt.setString(1, args[0]);
pstmt.setDate(2, new java.sql.Date(new java.util.Date().getTime()));
//pstmt.setTimestamp(2, new java.sql.Timestamp(new java.util.Date().getTime()));
// setDate() 를 사용하면 날짜만 들어감 시분은 무조건 0시 0분이됨
pstmt.executeUpdate();
```

```
ResultSet rs = stmt.executeQuery("select name, reservedate from datetest_t");
System.out.println("예약자명 wt getString wtwtgetDate wtgateTimeString");
while( rs.next() ) {
    System.out.print(rs.getString("name"));
    System.out.print("wt"+rs.getString("reservedate"));
    System.out.print("wt"+rs.getDate("reservedate"));
    System.out.println("wt"+rs.getTimestamp("reservedate"));
}
```

```
conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
pstmt = conn.prepareStatement("INSERT INTO datetest_t VALUES (?, sysdate)");
pstmt.setString(1, args[0]);
pstmt.executeUpdate();
```

```
conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
pstmt = conn.prepareStatement("INSERT INTO datetest_t VALUES " +
    " (?, to_date(?, 'yyyyw"년w"mmw"월w"ddw"일w"hhw"시w"miw"분w'))");
System.out.print("입력을 입력하세요 : ");
Scanner scann = new Scanner(System.in);
String name = scann.nextLine();
System.out.print("예약 년도를 입력하세요(XXXX년) : ");
String year = scann.nextLine();
System.out.print("예약 월을 입력하세요(X월) : ");
String month = scann.nextLine();
System.out.print("예약 날짜를 입력하세요(X일) : ");
String date = scann.nextLine();
System.out.print("예약 시간을 입력하세요(X시) : ");
String hour = scann.nextLine();
System.out.print("예약 분을 입력하세요(X분) : ");
```

```
String minute = scann.nextLine();
String reservation = year+month+date+hour+minute;
pstmt.setString(1, name);
pstmt.setString(2, reservation);
pstmt.executeUpdate();
```

```
conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
pstmt = conn.prepareStatement("INSERT INTO datetest_t VALUES (?, ?)");
GregorianCalendar cal = new GregorianCalendar(year,month-1,date,hour,minute);
pstmt.setString(1, name);
pstmt.setTimestamp(2, new java.sql.Timestamp(cal.getTimeInMillis()));
pstmt.executeUpdate();
```

```
conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
pstmt = conn.prepareStatement("INSERT INTO datetest_t VALUES (?, ?)");
SimpleDateFormat df = new SimpleDateFormat("yyyy년MM월dd일 HH시mm분");
java.util.Date d=df.parse(year+"년"+month+"월"+date+"일 "+hour+"시"+minute+"분");
pstmt.setString(1, name);
pstmt.setTimestamp(2, new java.sql.Timestamp(d.getTime()));
pstmt.executeUpdate();
```

```
Connection conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select name, to_char(reservedate,
    'yyyyW"년W"mmW"월W"ddW"일W"hhW"시W"miW"분W"') dd from datetest");
while( rs.next() ) {
    System.out.print(rs.getString("name"));
    System.out.println("Wt"+rs.getString("dd"));
}
Connection conn = DriverManager.getConnection("JDBCURL문자열", "계정", "암호");
Statement stmt = conn.createStatement();
ResultSet rs = stmt.executeQuery("select name, "+
    "to_char(reservedate, 'yyyyW"년W"mmW"월W"ddW"일W"') 일자,"+
    "to_char(reservedate, 'hhW"시W"miW"분W"') 시간 from datetest");
while( rs.next() ) {
    System.out.print(rs.getString("name"));
    System.out.print("Wt"+rs.getString("일자"));
    System.out.println("Wt"+rs.getString("시간"));
}
```