

```
for object to mirror_mod.mirror_object
operation == "MIRROR_X":
    mirror_mod.use_x = True
    mirror_mod.use_y = False
    mirror_mod.use_z = False
operation == "MIRROR_Y":
    mirror_mod.use_x = False
    mirror_mod.use_y = True
    mirror_mod.use_z = False
operation == "MIRROR_Z":
    mirror_mod.use_x = False
    mirror_mod.use_y = False
    mirror_mod.use_z = True

#selection at the end -add
mirror_ob.select= 1
modifier_ob.select=1
context.scene.objects.active
("Selected" + str(modifier_ob.name))
mirror_ob.select = 0
= bpy.context.selected_objects[0]
data.objects[one.name].select

print("please select exactly one object")

-- OPERATOR CLASSES -----

types.Operator):
    "X mirror to the selected object.mirror_mirror_x"
    "mirror X"
```

Java 기초

JDBC

1. I/O의 개요

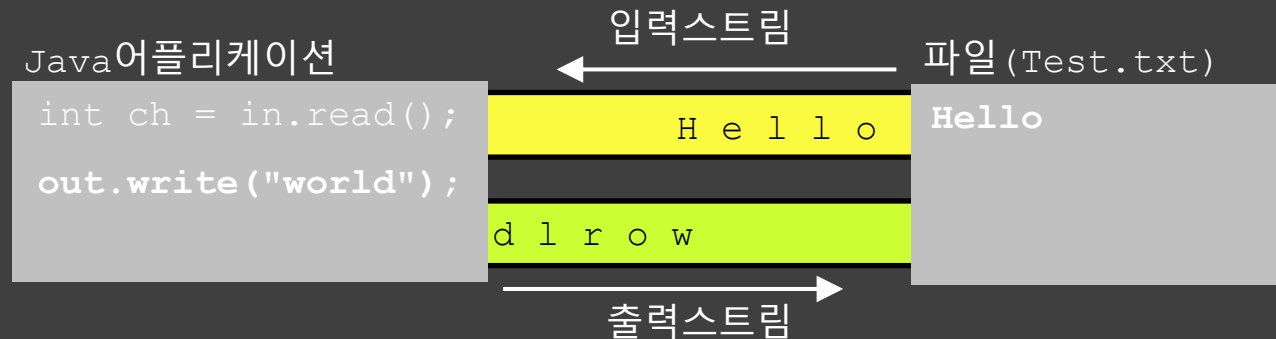
1. 입출력(I/O)와 스트림(Stream)

▶ 입출력(I/O)

- 입출력 : 두 개의 단말기 혹은 대상간에 데이터를 주고받는 것

▶ 스트림

- 데이터를 운반(입출력)하는데 사용되는 연결통로
- 연속적인 데이터의 흐름을 물(stream)에 비유해서 붙여진 이름
- 하나의 스트림으로 입출력을 동시에 수행할 수 없다.(단방향 통신)
- 입출력을 동시에 수행하려면, 2개의 스트림이 필요하다.



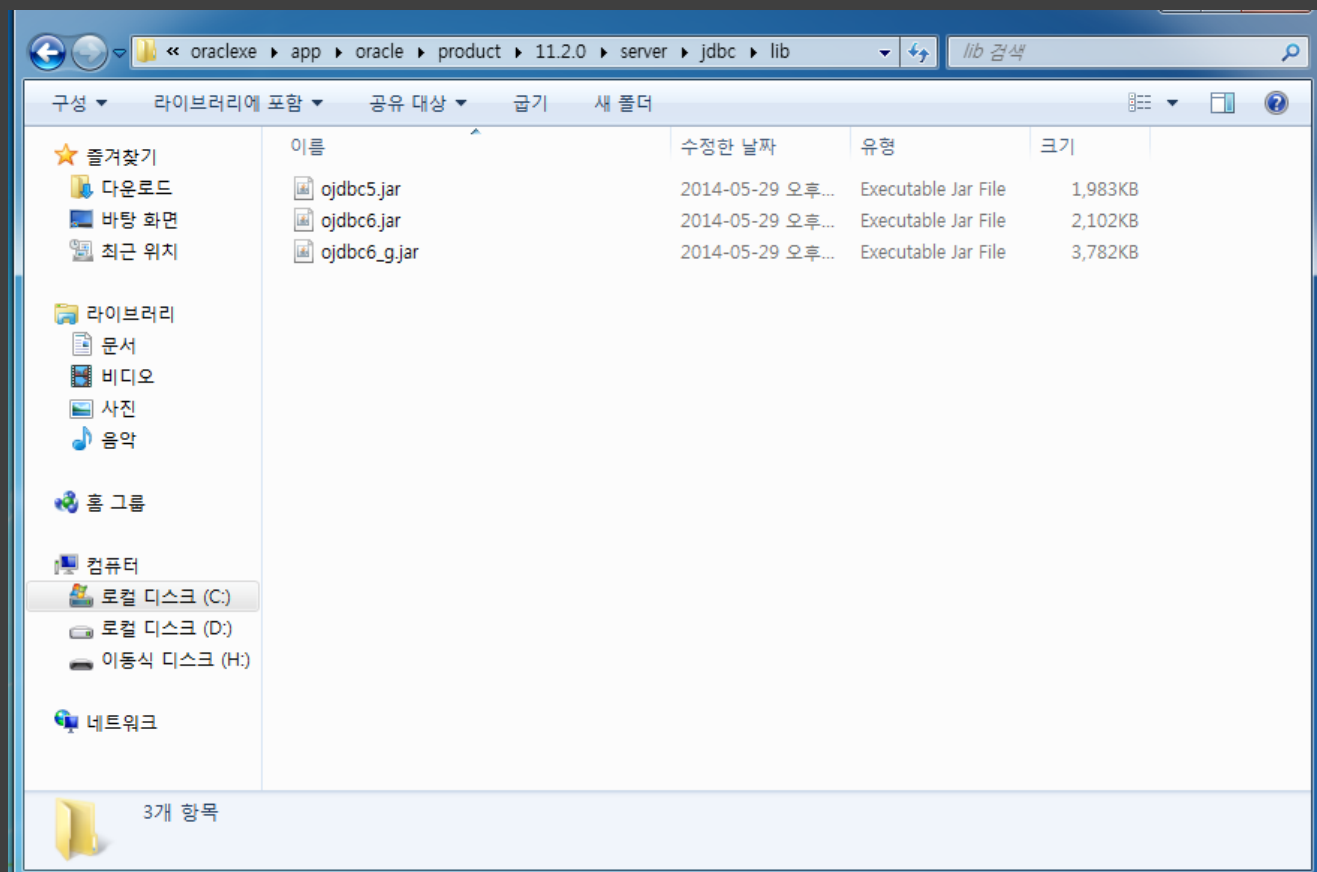
1. JDBC

- 데이터베이스에 접근하여 SQL문을 실행하기 위한 자바 라이브러리
- 썬 에서 RDBMS에 접근하여 SQL문을 실행하기 위한 자바 라이브러리를 만들어 표준으로 제공한 것.
- JDBC에는 구현클래스가 거의 없고 대부분이 인터페이스이다.
- 인터페이스를 구현한 클래스를 만들어 제공하는 것은 각 벤더의 책임이다.

2.1 JDBC의 Oracle 연동

- JAVA에서 Oracle DataBase연동을 위해선 OJDBC(Oracle JDBC)가 필요하다.

=> C:\Woraclexe\Wapp\Woracle\Wproduct\W11.2.0\server\Wjdbc\Wlib



2.2 JDBC의 MySQL 연동

- JAVA에서 MySQL DataBase연동을 위해선 ConnectionJ가 필요하다.

=> <https://dev.mysql.com/downloads/connector/j/>


Connector/J 5.1.41

Select Operating System:

Platform Independent ▼

Looking for previous GA versions?

Platform Independent (Architecture Independent), Compressed TAR Archive (mysql-connector-java-5.1.41.tar.gz)	5.1.41	3.7M	Download
		MD5: de520f86476298364576b541e0f8aef Signature	
Platform Independent (Architecture Independent), ZIP Archive (mysql-connector-java-5.1.41.zip)	5.1.41	4.1M	Download
		MD5: aaab750c83a3eca10f8c406a9e902a20 Signature	

 We suggest that you use the [MD5 checksums](#) and [GnuPG signatures](#) to verify the integrity of the packages you download.



2.3 JDBC의 MariaDB 연동

- JAVA에서 MariaDB DataBase 연동을 위해선 ConnectionJ가 필요하다.

=> <https://downloads.mariadb.org/connector-java/>

MariaDB Connector/J 1.5 Series

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases. The client library is LGPL licensed.

See [this article](#) for more information

Download 1.5.9 Stable Now!

Release Notes

Changelog

MariaDB Connector/J 1.4 Series

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases. The client library is LGPL licensed.

See [this article](#) for more information

Download 1.4.6 Stable

Release Notes

Changelog

MariaDB Connector/J 1.3 Series

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases. The client library is LGPL licensed.

See [this article](#) for more information

Download 1.3.7 Stable

Release Notes

Changelog

MariaDB Connector/J 1.2 Series

MariaDB Connector/J is used to connect applications developed in Java to MariaDB and MySQL databases. The client library is LGPL licensed.

See [this article](#) for more information

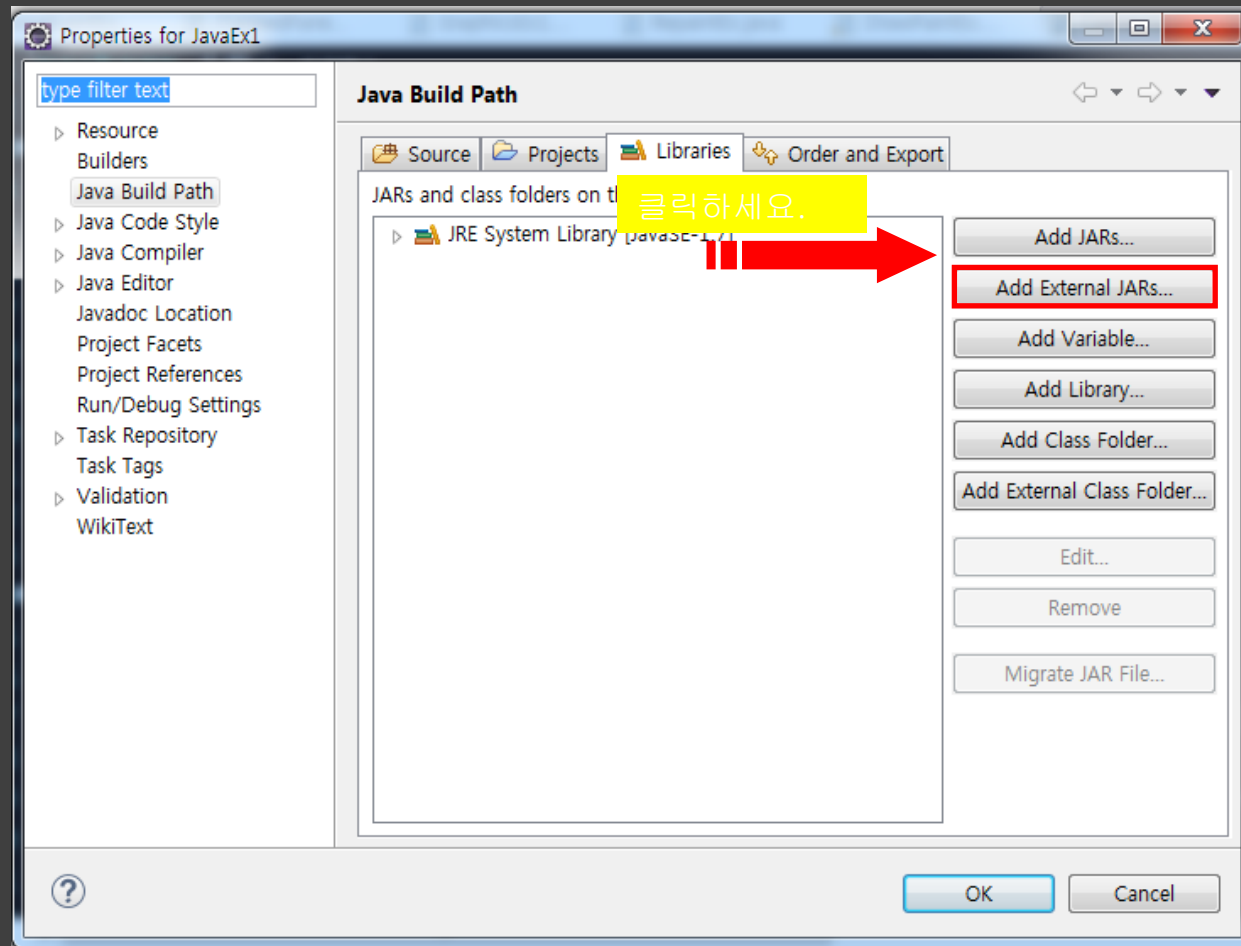
Download 1.2.3 Stable

Release Notes

Changelog

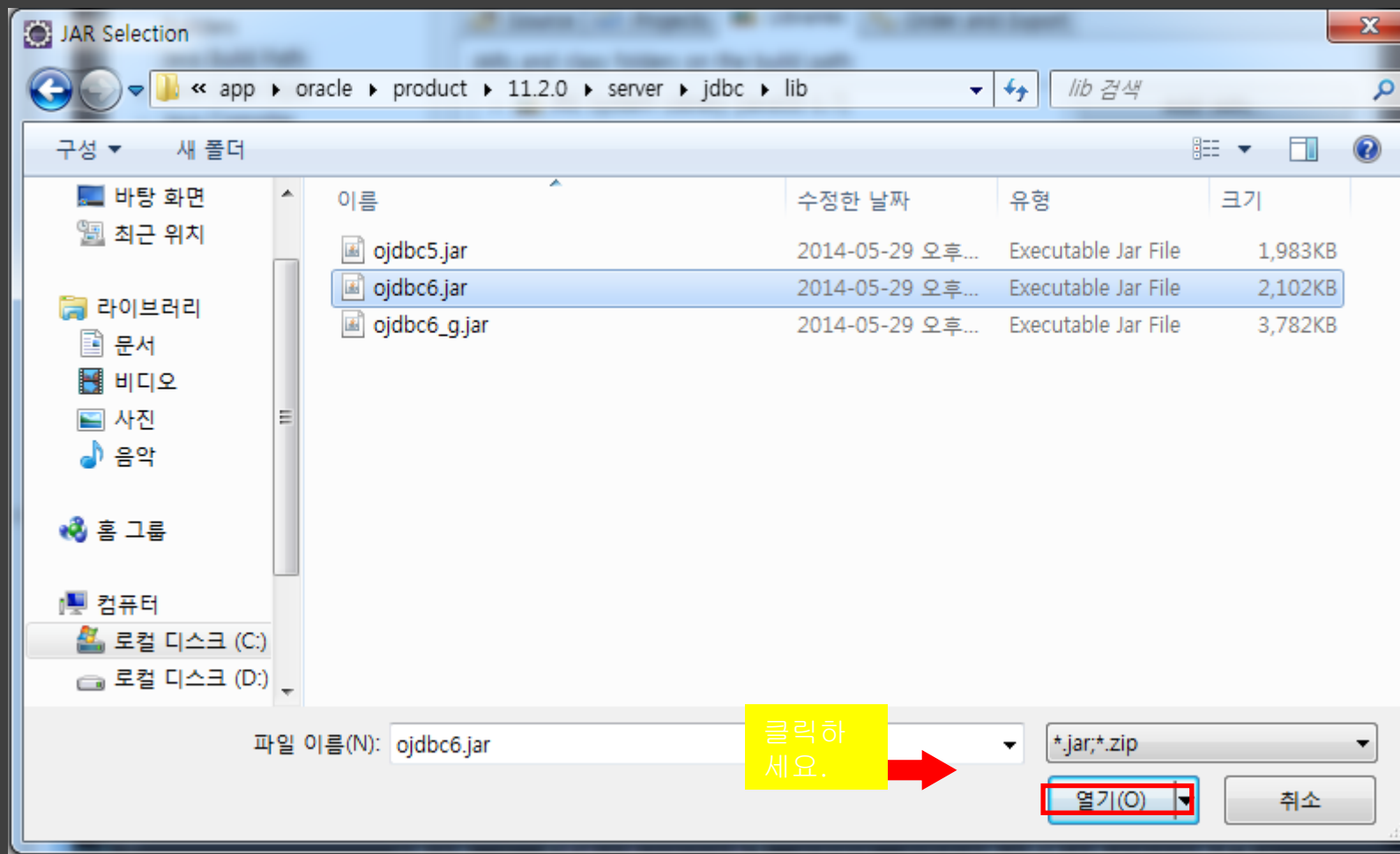
2. JDBC의 Oracle 연동

- properties -> java build path -> libraries 탭 클릭



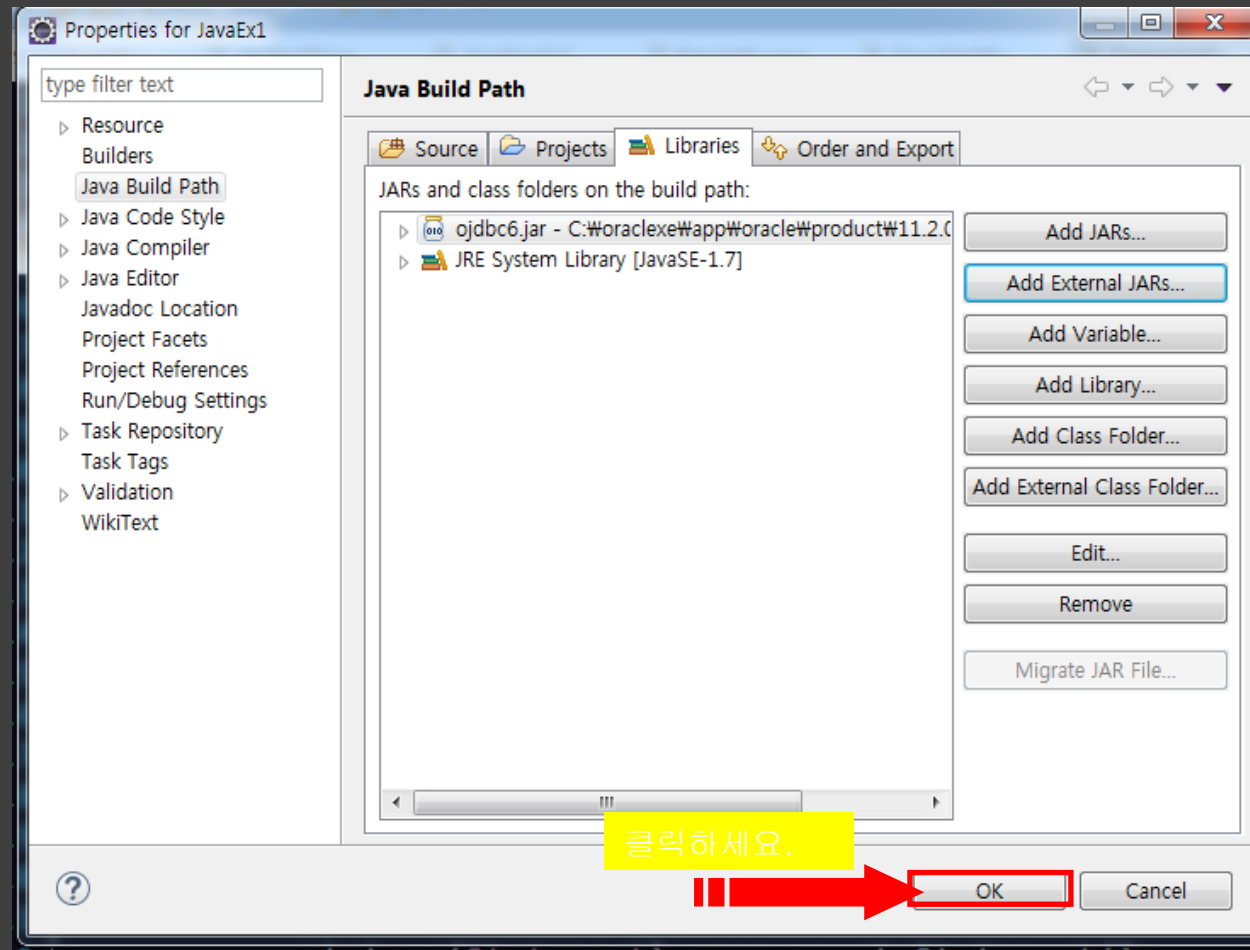
2. JDBC의 Oracle 연동

- jdk 1.6 이상은 ojdbc6 혹은 ojdbc6_g를 클릭 후 열기



2. JDBC의 Oracle 연동

- properties -> java build path -> libraries 탭 클릭



3. JDBC – Statement 예제

```
public class ConnectorEx {
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521:xe";
    String user = "scott";
    String passwd = "tiger";
    Connection conn;
    Statement stat;
    ResultSet rs;
    String connect = "jdbc:mysql://localhost:3306/scott";
    String connect = "jdbc:mariadb://localhost:3306/scott";
    public ConnectorEx() {
        try {
            // 1.driver 등록
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 2.connection 얻기 (네트워크 연결)
            conn = DriverManager.getConnection(connect, user, passwd);
            // 3. statement 얻기
            stat = (Statement) conn.createStatement();
            String query = "select * from emp";
            System.out.println(query);
            // 4.query 실행 후 결과 집합 얻기
            rs = stat.executeQuery(query);
            // 5.결과집합 처리
            while(rs.next()){
                int num = rs.getInt(1); // 1,2,3 은 인덱스
                // int num = rs.getString("EMPNO");// 컬럼값으로 줘도 무방하다
                String name = rs.getString(2);
                String job = rs.getString(3);
                System.out.println(num+"--" + name+"--" + job);
            }
        }
    }
}
```

3. JDBC – Statement 예제

```
// 5.결과집합 처리
while(rs.next()){
    int num = rs.getInt(1); // 1,2,3 은 인덱스
    // int num = rs.getString("EMPNO");// 컬럼값으로 줘도 무방하다
    String name = rs.getString(2);
    String job = rs.getString(3);
    System.out.println(num+"--" + name+"--" + job);
}

} catch (Exception e) {
    e.printStackTrace();
} finally {
    // 6.마무리 작업
    try {
        rs.close();
        stat.close();
        conn.close();
    } catch (SQLException e) {
        e.printStackTrace();
    }
}

}

public static void main(String[] args) {
    new ConnectorEx();
}
```

```
select * from emp
7369--SMITH--CLERK
7499--ALLEN--SALESMAN
7521--WARD--SALESMAN
7566--JONES--MANAGER
7654--MARTIN--SALESMAN
7698--BLAKE--MANAGER
7782--CLARK--MANAGER
7839--KING--PRESIDENT
7844--TURNER--SALESMAN
7900--JAMES--CLERK
7902--FORD--ANALYST
7934--MILLER--CLERK
```

4. JDBC – PreparedStatement(Select문) 예제

```
public class PreparedStatementEx1 {
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521/xe";
    String user = "scott";
    String passwd = "tiger";
    Connection conn;
    PreparedStatement pstmt;
    ResultSet rs;
    public PreparedStatementEx1() {
        try {
            // 1.driver 등록
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 2.connection 얻기 (네트워크 연결)
            conn = DriverManager.getConnection(connect, user, passwd);
            // 3. PreparedStatement 얻기
            String query = "select * from emp where deptno = ? ";
            System.out.println(query);
            pstmt = conn.prepareStatement(query);
            pstmt.setInt(1, 10);
            // 4.query 실행 후 결과 집합 얻기
            rs = pstmt.executeQuery();
            // 5.결과집합 처리
            while(rs.next()){
                int num = rs.getInt(1); // 1,2,3 은 인덱스
                // int num = rs.getString("EMPNO");// 컬럼값으로 줘도 무방하다
                String name = rs.getString(2);
                String job = rs.getString(3);
                System.out.println(num+"--" + name+"--" + job);
            }
        }
    }
}
```

4. JDBC – PreparedStatement(Select문) 예제

```
public class PreparedStatementEx1 {
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521/xe";
    String user = "scott";
    String passwd = "tiger";
    Connection conn;
    PreparedStatement pstmt;
    ResultSet rs;
    public PreparedStatementEx1() {
        try {
            // 1.driver 등록
            Class.forName("oracle.jdbc.driver.OracleDriver");
            // 2.connection 얻기 (네트워크 연결)
            conn = DriverManager.getConnection(connect, user, passwd);
            // 3. PreparedStatement 얻기
            String query = "select * from emp where deptno = ? ";
            System.out.println(query);
            pstmt = conn.prepareStatement(query);
            pstmt.setInt(1, 10);
            // 4.query 실행 후 결과 집합 얻기
            rs = pstmt.executeQuery();
            // 5.결과집합 처리
            while(rs.next()){
                int num = rs.getInt(1); // 1,2,3 은 인덱스
                // int num = rs.getString("EMPNO");// 컬럼값으로 줘도 무방하다
                String name = rs.getString(2);
                String job = rs.getString(3);
                System.out.println(num+"--" + name+"--" + job);
            }
        }
    }
}
```

select * from emp where deptno = ?
7782--CLARK--MANAGER
7839--KING--PRESIDENT
7934--MILLER--CLERK

4. JDBC – PreparedStatement(Insert문) 예제

```
public class PreparedStatementEx2 {  
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521/xe";  
    String user = "scott";  
    String passwd = "tiger";  
    Connection conn;  
    PreparedStatement pstmt;  
    public PreparedStatementEx2() {  
        try {  
            // 1.driver 등록  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            // 2.connection 열기 (네트워크 연결)  
            conn = DriverManager.getConnection(connect, user, passwd);  
            // 3. PreparedStatement 열기  
            String query = "insert into dept(deptno,dname,loc) values(?,?,?) ";  
            System.out.println(query);  
            pstmt = conn.prepareStatement(query);  
            pstmt.setInt(1, 50);  
            pstmt.setString(2, "AAAA");  
            pstmt.setString(3, "BBBB");  
            // 4.query 실행  
            pstmt.executeUpdate();  
        }  
    }  
}
```

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	AAAA	BBBB

4. JDBC – PreparedStatement(update문) 예제

```
public class PreparedStatementEx3 {  
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521/xe";  
    String user = "scott";  
    String passwd = "tiger";  
    Connection conn;  
    PreparedStatement pstmt;  
    public PreparedStatementEx3() {  
        try {  
            // 1.driver 등록  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            // 2.connection 얻기 (네트워크 연결)  
            conn = DriverManager.getConnection(connect, user, passwd);  
            // 3. PreparedStatement 얻기  
            String query = "update dept set loc = ? where deptno = ? ";  
            System.out.println(query);  
            pstmt = conn.prepareStatement(query);  
            pstmt.setString(1, "CCCC");  
            pstmt.setInt(2, 50);  
            // 4.query 실행  
            pstmt.executeUpdate();  
        }  
    }  
}
```

SQL> select * from dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON
50	AAAA	CCCC

4. JDBC – PreparedStatement(delete문) 예제

```
public class PreparedStatementEx4 {  
    String connect = "jdbc:oracle:thin:@127.0.0.1:1521/xe";  
    String user = "scott";  
    String passwd = "tiger";  
    Connection conn;  
    PreparedStatement pstmt;  
    public PreparedStatementEx4() {  
        try {  
            // 1.driver 등록  
            Class.forName("oracle.jdbc.driver.OracleDriver");  
            // 2.connection 얻기 (네트워크 연결)  
            conn = DriverManager.getConnection(connect, user, passwd);  
            // 3. PreparedStatement 얻기  
            String query = "delete from dept where deptno = ? ";  
            System.out.println(query);  
            pstmt = conn.prepareStatement(query);  
            pstmt.setInt(1, 50);  
            // 4.query 실행  
            pstmt.executeUpdate();  
        }  
    }  
}
```

SQL> select * from dept;

DEPTNO	DNAME	LOC
10	ACCOUNTING	NEW YORK
20	RESEARCH	DALLAS
30	SALES	CHICAGO
40	OPERATIONS	BOSTON

5. JDBC – AutoCommit, commit, rollback.

```
public PreparedStatementEx4() {
    try {
        // 1.driver 등록
        Class.forName("oracle.jdbc.driver.OracleDriver");
        // 2.connection 열기 (네트워크 연결)
        conn = DriverManager.getConnection(connect, user, passwd);
        conn.setAutoCommit(false); // 오토 commit 세팅을 false로 둠
        // 3. PreparedStatement 열기
        String query = "delete from dept where deptno = ? ";
        System.out.println(query);
        pstmt = conn.prepareStatement(query);
        pstmt.setInt(1, 50);
        // 4.query 실행
        int i = pstmt.executeUpdate();
        if(i > 0)conn.commit(); // commit
    } catch (Exception e) {
        try {
            conn.rollback(); // rollback
        } catch (SQLException e1) {
            e1.printStackTrace();
        }
        e.printStackTrace();
    }
}
```