14주차 1차시 자바 프로그래밍 2

【학습목표】

1. 자바와 MySQL을 연동하는 방법을 설명할 수 있다.
2. MySQL에 질의하고 결과를 받기 위한 자바 클래스와 메소드를 살펴보고 실습을 통하여 설명할 수 있다.

학습내용1 : 자바와 MySQL의 연동

A. 연동 방법

B. MySQL 서버의 JDBC 드라이버 로드

- 드라이버 클래스 파일을 로드
- Class 클래스의 forName( ) 메소드를 이용하여 특정 클래스 파일을 읽을 수 있다.

```
try       {
     Class.forName( “com.mysql.jdbc.Driver” );
} catch ( ClassNotFoundException e )     {
     e.printStackTrace( );
}
```

- com.mysql.jdbc.Driver 클래스를 로드하여 드라이버 인스턴스를 생성
- DriverManager에 등록
- JDBC 드라이버의 클래스 이름은 DBMS에 따라 다름
- JDBC 드라이버가 없을 경우, ClassNotFoundException 예외가 발생

C. 데이터베이스 연결

- DriverManager.getConnection( ) 메소드를 호출하여 데이터베이스에 연결
- Connection 객체를 반환

```
try       {
     Connection   conn = DriverManager.getConnection(
                                  “jdbc:mysql://localhost:3306/itbank”, “cskim”, “1111” );
} catch ( SQLException e )       {
     e.printStackTrace( );
}
```

- getConnection( )의 매개변수에서 jdbc: 이후의 URL은 DBMS에 따라 다름
- MySQL서버가 같은 컴퓨터에 동작할 경우, 서버 주소로 localhost를 사용
- MySQL은 3306 Port를 사용
- 사용할 데이터베이스 이름을 지정
- 로그인 할 계정과 비밀번호를 지정

[프로그램]

```
import   java.sql.*;

public   class   jdbcEx   {
    public   static   void   main( String[ ]   args )   {
        try   {
            // MySQL 드라이버 로드
            Class.forName( "com.mysql.jdbc.Driver" );

            // JDBC 연결
            Connection   conn = DriverManager.getConnection(
                                "jdbc:mysql://localhost:3306/itbank", "cskim", "1111" );

            System.out.println( "DB 연결" );
        }  catch ( ClassNotFoundException e )   {
            System.out.println( "JDBC 드라이버 로드 오류" );
        }  catch ( SQLException e )  {
            System.out.println( "DB 연결 오류" );
        }
    }
}
```

학습내용2 : 데이터베이스 사용

- 자바에서 데이터베이스 연결 후, MySQL 질의문을 사용하여 데이터베이스에 접근
- Statement 클래스 : SQL문을 실행하기 위한 클래스
- ResultSet 클래스 : SQL문 실행 결과를 받아오기 위해 사용하는 클래스

1. Statement 클래스 메소드

- ResultSet executeQuery(String sql)
  . 데이터 검색을 위한 SQL을 실행
  . 결과를 ResultSet 객체로 반환

- int executeUpdate(String sql)
  . insert, update, delete와 같은 데이터 변경을 위한 sql 문을 실행
  . sql 문 실행으로 영향을 받은 행의 개수나 0을 반환

- void close( )
  . Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환

2. ResultSet 클래스

- 현재 데이터의 형(레코드의 위치)를 가리키는 커서를 관리
- 커서의 초깃 값 : 첫 번째 행 이전을 가리킴
- 커서의 위치와 관련된 메소드와 레코드를 가져오는 메소드를 제공

- ResultSet 클래스 메소드

(1) boolean first( )
  - 커서를 첫 번째 행으로 이동

(2) boolean last( )
  - 커서를 마지막 행으로 이동

(3) boolean next( )
  - 커서를 다음 행으로 이동

(4) boolean previous( )
  - 커서를 이전 행으로 이동

(5) boolean absolute(int row)
  - 커서를 지정된 행 row으로 이동

(6) boolean isFirst( )
  - 첫 번째 행이면 true를 반환

(7) boolean isLast( )
  - 마지막 행이면 true를 반환

(8) void close( )
  - ResultSet 객체의 데이터베이스와 JDBC 리소스를 반환

(9) Xxx getXxx(String columnLable)
  - Xxx는 해당 데이터 타입
  - 현재 행에서 지정된 열이름(columnLable)에 해당하는 데이터를 반환
  - 예: int 형 데이터를 읽는 메소드 - getInt( )

(10) Xxx getXxx(int columnIndex)
  - Xxx는 해당 데이터 타입
  - 현재 행에서 지정된 열 인덱스(columnIndex)에 해당하는 데이터를 반환

3. 데이터 검색 예

- Customer 테이블의 모든 데이터를 검색하시오.

```
Statement stmt = conn.createStatement( );
ResultSet  rs  = stmt.executeQuery( "select * from customer" );
```

- Customer 테이블의 계정과 이름 속성 데이터를 검색하시오.

```
ResultSet  rs  = stmt.executeQuery( "select account, name from customer" );
```

4. 검색된 데이터의 사용

- 검색된 결과는 ResultSet 객체에 저장
- getString( ) : ResultSet 클래스의 메소드를 이용하여 커서가 가리키는 현재 행에 대해 열의 값을 읽는다.

```
String  sAccount = rs.getString( "account" );
String  sName    = rs.getString( "name" );
int     credit   = rs.getInt( "credit" );
```

- Customer 테이블로부터 검색된 모든 데이터를 출력하시오.

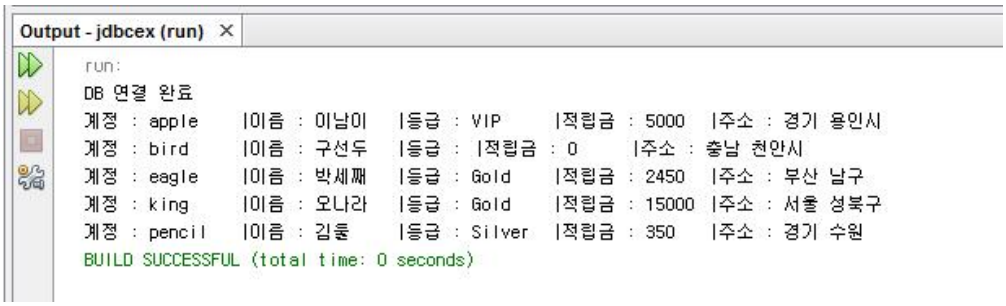
```
while ( rs.next( ) )      {
    System.out.print( "계정:" + rs.getString( "account" ) + "₩t|" );
    System.out.print( "이름:" + rs.getString( "name" ) + "₩t|" );
    System.out.print( "등급:" + rs.getString( "grade" ) + "₩t|" );
    System.out.print( "적립금:" + rs.getInt( "credit" ) + "₩t|" );
    System.out.print( "주소:" + rs.getString( "address" ) );
    System.out.println( );
}
```

. 전체 프로그램

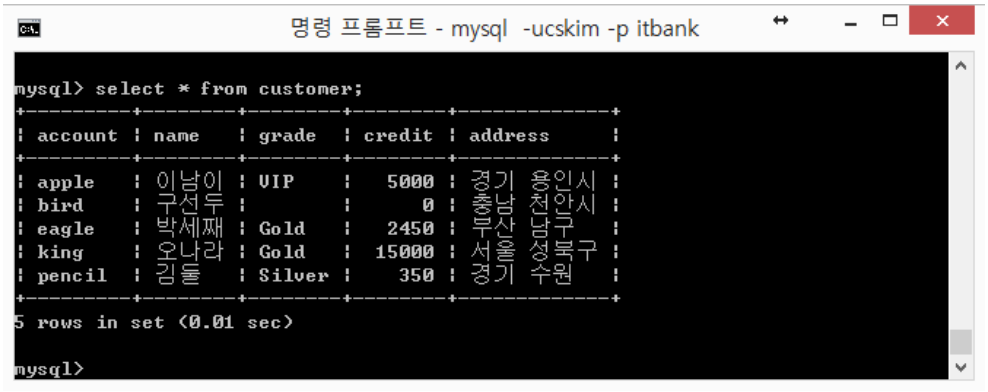
```
import   java.io.*;
import   java.sql.*;

public class Jdbcex    {
    public static void main(String[] args)  {
        Connection conn;
        Statement   stmt = null;

        try       {
            // MySQL 드라이버 로드
            Class.forName( "com.mysql.jdbc.Driver" );

            // JDBC 연결
            conn = DriverManager.getConnection(
                        "jdbc:mysql://localhost:3306/itbank", "cskim", "strong");
            System.out.println( "DB 연결 완료");

            // SQL문 처리용 객체
            stmt = conn.createStatement( );

            // Customer 테이블 검색
            ResultSet res = stmt.executeQuery( "SELECT * FROM customer");
            printData( res );

                } catch ( ClassNotFoundException e )     {
            System.out.println( "JDBC 드라이버 로드 오류.....");
                } catch ( SQLException e )  {
            System.out.println( "DB 연결 오류");
        }
    }

    private static void printData( ResultSet res ) throws SQLException
    {
        while ( res.next( ) )    {
            System.out.print( "계정 : " + res.getString( "account" ) + "₩t|" );
            System.out.print( "이름 : " + res.getString( "name" ) + "₩t|" );
            System.out.print( "등급 : " + res.getString( "grade" ) + "₩t|" );
            System.out.print( "적립금 : " + res.getString( "credit" ) + "₩t|" );
            System.out.print( "주소 : " + res.getString( "address" ) );
            System.out.println( );
        }
    }
```

}

. 실행 결과



. Customer 테이블 데이터



5. 데이터 변경

- 데이터 추가, 수정, 삭제와 같은 데이터 변경은 executeUpdate( ) 메소드를 사용

(1) 레코드 추가

- 다음의 데이터를 Customer 테이블에 추가하시오.

. account : watch, name : 김정확, grade : New, credit : 1000, address = NULL

. Java 코드

```
stmt.executeUpdate( "INSERT INTO customer " +
        "( account, name, grade, credit, address ) " +
        " VALUES ( 'watch', '김정확', 'New', 1000, NULL );" );
```

. 수행 후 결과

```
Output - jdbcex (run)
run:
DB 연결 완료
계정 : apple    |이름 : 이남이   |등급 : VIP     |적립금 : 5000   |주소 : 경기 용인시
계정 : bird     |이름 : 구선두   |등급 :  |적립금 : 0      |주소 : 충남 천안시
계정 : eagle    |이름 : 박세째   |등급 : Gold    |적립금 : 2450   |주소 : 부산 남구
계정 : king     |이름 : 오나라   |등급 : Gold    |적립금 : 15000  |주소 : 서울 성북구
계정 : pencil   |이름 : 김둘     |등급 : Silver  |적립금 : 350    |주소 : 경기 수원
계정 : watch    |이름 : 김정확   |등급 : New     |적립금 : 1000   |주소 : null
BUILD SUCCESSFUL (total time: 0 seconds)
```

. Customer 테이블 데이터

명령 프롬프트 - mysql -ucskim -p itbank

```
mysql> select * from customer;
+---------+--------+--------+--------+-------------+
| account | name   | grade  | credit | address     |
+---------+--------+--------+--------+-------------+
| apple   | 이남이 | VIP    |   5000 | 경기 용인시 |
| bird    | 구선두 |        |      0 | 충남 천안시 |
| eagle   | 박세째 | Gold   |   2450 | 부산 남구   |
| king    | 오나라 | Gold   |  15000 | 서울 성북구 |
| pencil  | 김둘   | Silver |    350 | 경기 수원   |
| watch   | 김정확 | New    |   1000 | NULL        |
+---------+--------+--------+--------+-------------+
6 rows in set (0.01 sec)

mysql>
```

(2) 데이터 수정

- Customer 테이블에서 계정(account)이 'bird'인 고객의 등급(grade)를 'Silver'로 수정하시오.

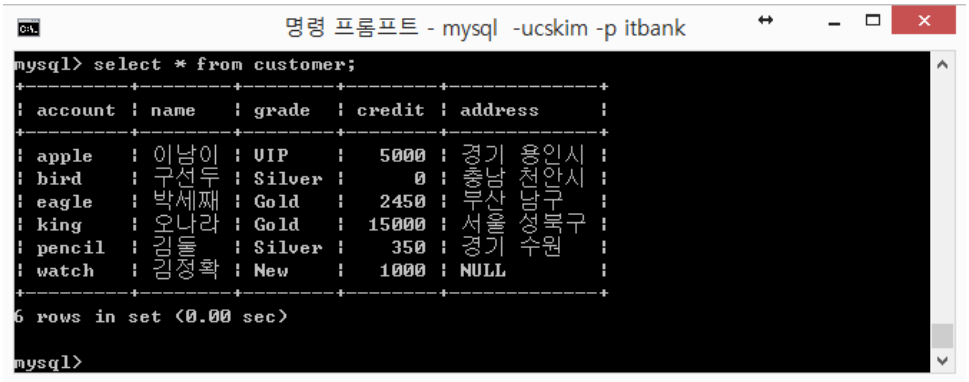
. Java 코드

```
stmt.executeUpdate( "UPDATE customer SET grade='Silver' "
                    + " WHERE account='bird'; " );
```

. 실행 결과

```
Output - jdbcex (run)
run:
DB 연결 완료
계정 : apple    |이름 : 이남이   |등급 : VIP     |적립금 : 5000   |주소 : 경기 용인시
계정 : bird     |이름 : 구선두   |등급 : Silver  |적립금 : 0      |주소 : 충남 천안시
계정 : eagle    |이름 : 박세째   |등급 : Gold    |적립금 : 2450   |주소 : 부산 남구
계정 : king     |이름 : 오나라   |등급 : Gold    |적립금 : 15000  |주소 : 서울 성북구
계정 : pencil   |이름 : 김둘     |등급 : Silver  |적립금 : 350    |주소 : 경기 수원
계정 : watch    |이름 : 김정확   |등급 : New     |적립금 : 1000   |주소 : null
BUILD SUCCESSFUL (total time: 0 seconds)
```

. Customer 테이블 데이터



(3) 레코드 삭제

- Customer 테이블에서 등급(grade)가 'Silver'인 고객의 레코드를 삭제하시오.
- Java 코드

  stmt.executeUpdate("DELETE FROM customer WHERE grade='Silver'; ");

- 실행 결과

- Customer 테이블 데이터

【학습정리】

1. 자바와 MySQL을 연동하는 방법은 JDBC 드라이버를 로드하고, 데이터베이스를 연결, 질의문 작성, MySQL에 질의하고 결과 받기, 결과를 이용, 사용한 리소스 반환하는 절차로 이루어진다.

2. 데이터베이스를 사용하기 위하여 자바의 Statement 클래스와 ResultSet 클래스를 사용한다.