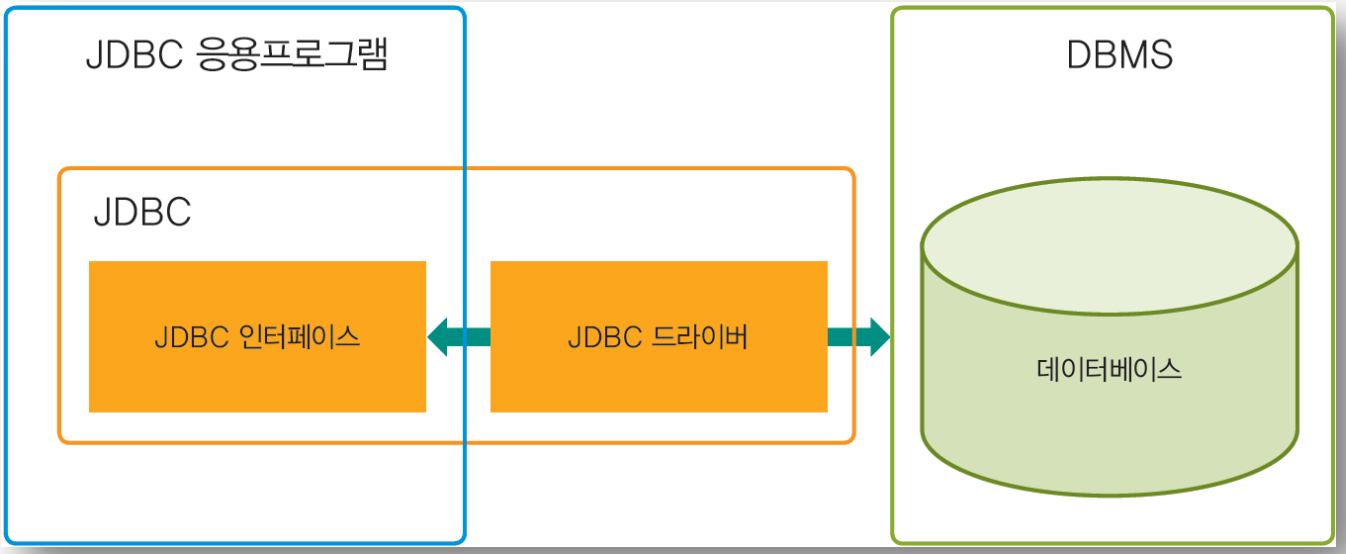
JDBC 프로그래밍 구현

목차

- **JDBC와 드라이버에 대해 이해하고 JDBC 프로그래밍 6 단계를 구현할 수 있다.**
  - • JDBC 의미와 드라이버의 필요성
  - • JDBC 프로그래밍 6단계 구현
  - • JDBC 관련 클래스의 이해
- **JDBC 프로그래밍 방법을 이해하고 구현할 수 있다.**
  - • JDBC 드라이버 로드와 데이터베이스 연결 방법
  - • 원하는 테이블 생성 및 레코드 삽입
  - • 테이블 내용 조회

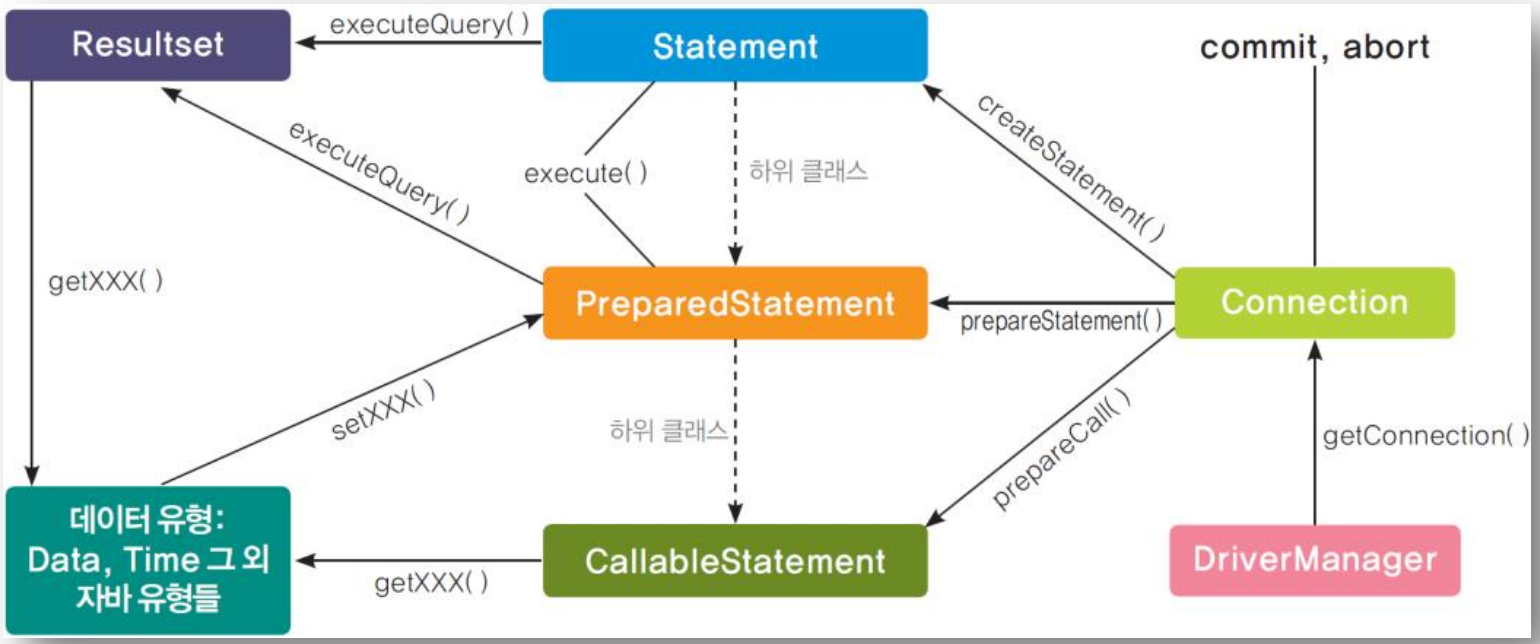
JDBC 이해

- **Java DataBase Connectivity의 첫 자로 구성된 JDBC**
  - 자바 언어로 데이터베이스 프로그래밍을 하기 위한 라이브러리
  - 특정한 DBMS에 종속되지 않는 관련 API(Application Programming Interface)를 제공



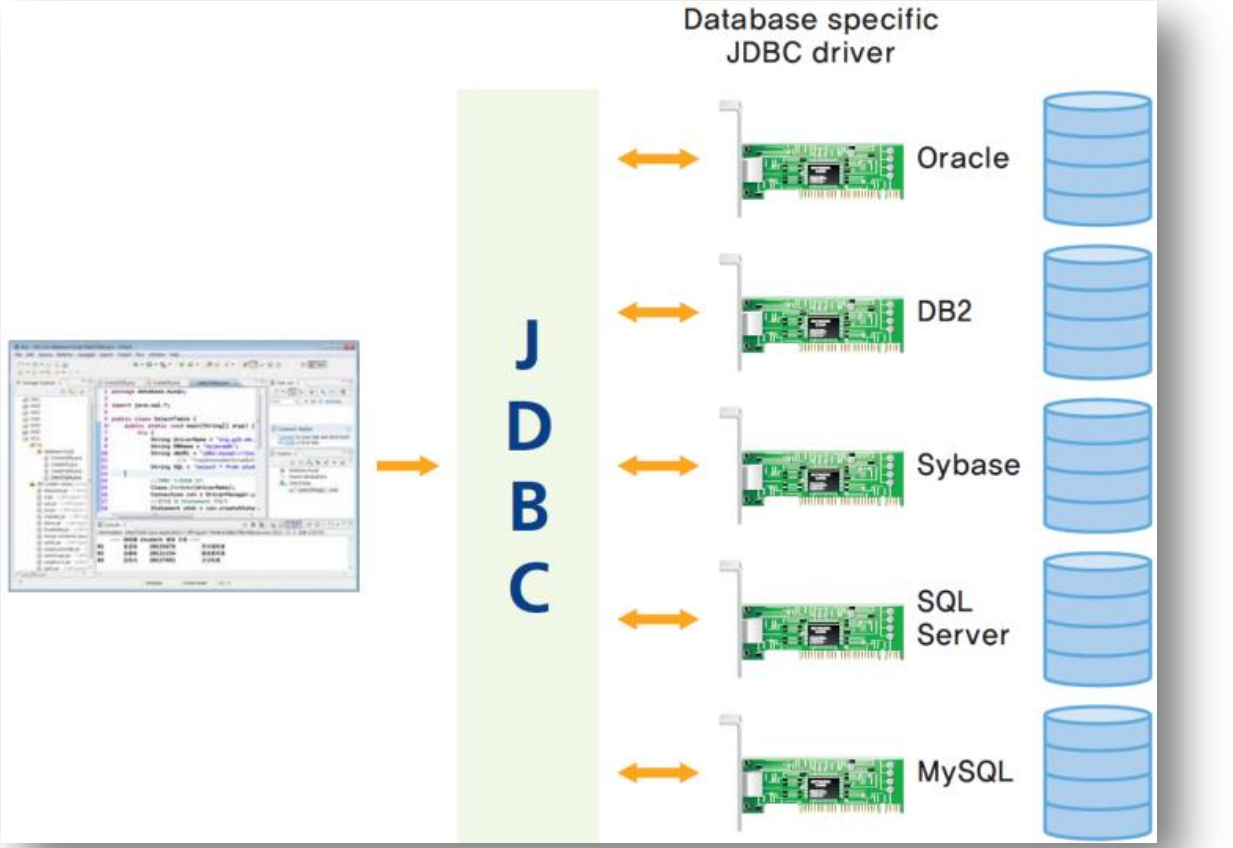
JDBC API 클래스

- **패키지 java.sql와 javax.sql로 구성**
  - Driver, DriverMananager, Connection, Statement, PreparedStatement, CallableStatement, ResultSet, ResultSetMetaData, DatabaseMetaData, DataSource 등
- **데이터베이스 기능을 지원하기 위한 표준 API**
  - 데이터베이스를 연결하여 테이블 형태의 자료를 참조
  - SQL 문을 질의
  - SQL 문의 결과를 처리



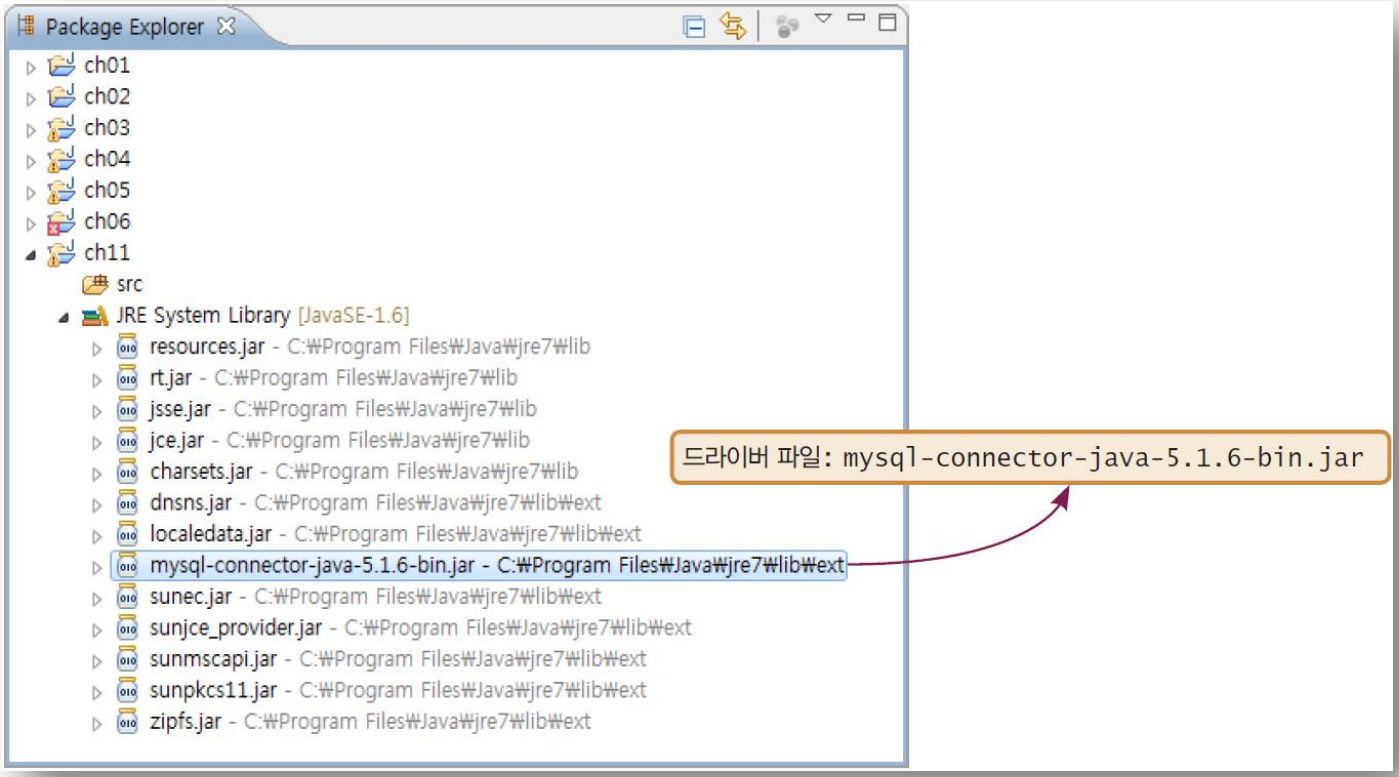
JDBC 역할

- **다양한 DBMS에 독립적으로 데이터베이스 프로그래밍을 가능하도록 하는 API(application programming interfaces) 규격**
  - 오라클(ORACLE), MySQL, SQLServer, DB2 등 어떤 DBMS를 사용하던지 소스의 수정을 최소화하여 바로 실행
  - JDBC와 함께 JDBC 드라이버(JDBC Driver)도 필요



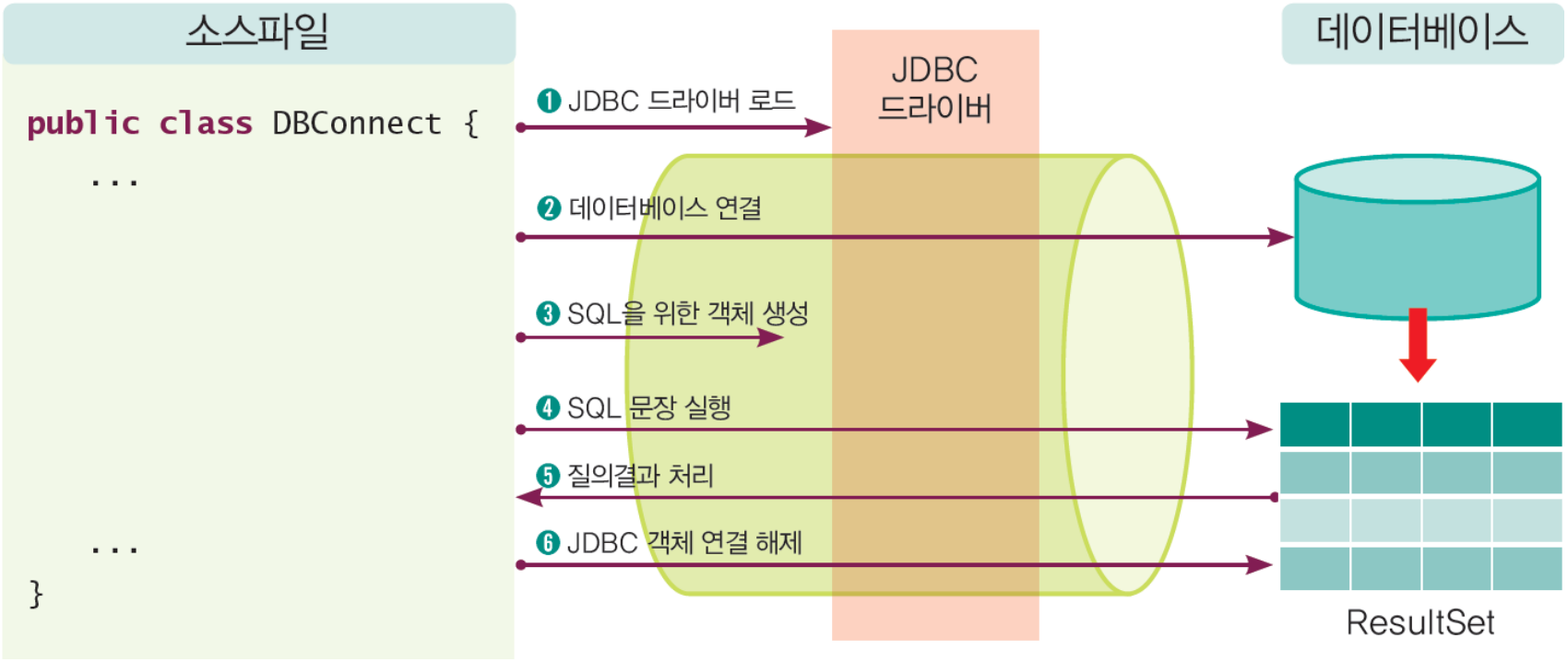
JDBC 드라이버 설치

- **적당한 버전의 Connector/J 커넥터 설치**
  - [jre 설치폴더] 하부 [lib/ext]에 [mysql-connector-java-5.1.6-bin.jar]와 같은 JDBC 드라이버 파일을 복사
    - 이클립스의 [JRE System Library] 하부에서 확인 가능
  - 다른 방법으로는 적당한 폴더에 복사한 후 classpath를 설정하는 방법



JDBC 프로그래밍 과정 6단계

- **select 문으로 간단한 질의를 수행하는 6단계**



JDBC 프로그래밍 과정 6단계-2

select 문으로 간단한 질의를 수행하는 6단계-2

❶ JDBC 드라이버 로드

```
Class.forName("org.gjt.mm.mysql.Driver");
```

❷ 데이터베이스 연결

```
String URL = "jdbc:mysql://localhost:3306/dbname";
Connecton con = DriverManager.getConnection(URL, "user", passwd");
```

❸ SQL을 위한 객체 생성

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

❹ SQL 문장 실행

```
String sql = "select * from student:';
ResultSet result = stmt.executeQuery(sql);
```

❺ 질의결과 처리

```
While(result.next()){
    String col1 = result.getString(1);
    String col2 = result.getString(2);
    int col3 = result.getint(3);
}
```

❻ JDBC 객체 연결 해제

```
result.closr();
stmt.close();
con.close();
```

다양한 JDBC 드라이버 로드

- **Class.forName()**
  - 동적으로 JDBC 드라이브 클래스를 로드
    - 드라이버 클래스가 객체화 되고 객체화와 동시에 자동적으로 DriverManager.registerDriver()를 호출
    - DriverManager에서 관리하는 드라이버 리스트에 드라이버 등록

```
String driverName = "com.mysql.jdbc.Driver";
Class.forName(driverName);
```

```
//JDBC 드라이버의 이름 지정
String driverName = "org.gjt.mm.mysql.Driver";
//JDBC 드라이버 로드
Class.forName(driverName);
```

다양한 JDBC 드라이버 로드

- DBMS MySQL 경우
  - JDBC 드라이버 이름을 [com.mysql.jdbc.Driver]로도 제공
- 다양한 JDBC 드라이버

DBMS 종류	JDBC 드라이버 로드 문장
ORACLE	`Class.forName("oracle.jdbc.driver.OracleDriver");`
MS SQLServer	`Class.forName("com.microsoft.jdbc.sqlserver.SQLServerDriver");`
mSQL	`Class.forName("com.imaginay.sql.msql.MsqlDriver");`
MySQL	`Class.forName("org.gjt.mm.mysql.Driver");` `Class.forName("com.mysql.jdbc.Driver ");`
ODBC	`Class.forName("sun.jdbc.odbc.JdbcOdbcDriver");`

다양한 JDBC 관련 클래스

주요 클래스

패키지	인터페이스(클래스)	클래스 용도
java.lang	클래스 Class	지정된 JDBC 드라이버를 실행시간에 메모리에 로드
java.sql	클래스 DriverManager	여러 JDBC 드라이버를 관리하는 클래스로 데이터베이스를 접속하여 연결 객체 반환
	인터페이스 Connection	특정한 데이터베이스 연결 상태를 표현하는 클래스로 질의할 문장 객체 반환
	인터페이스 Statement	데이터베이스에 SQL 질의 문장을 질의하여 그 결과인 결과 집합 객체를 반환
	인터페이스 ResultSet	질의 결과의 자료를 저장하며 테이블 구조

JDBC의 인터페이스

- 모든 데이터베이스에서 사용할 수 있는 공통적인 데이터베이스 참조 개념을 추상화
- 특정 제품의 JDBC드라이버에서 상속받아 구현됨

데이터베이스 연결 URL 구조

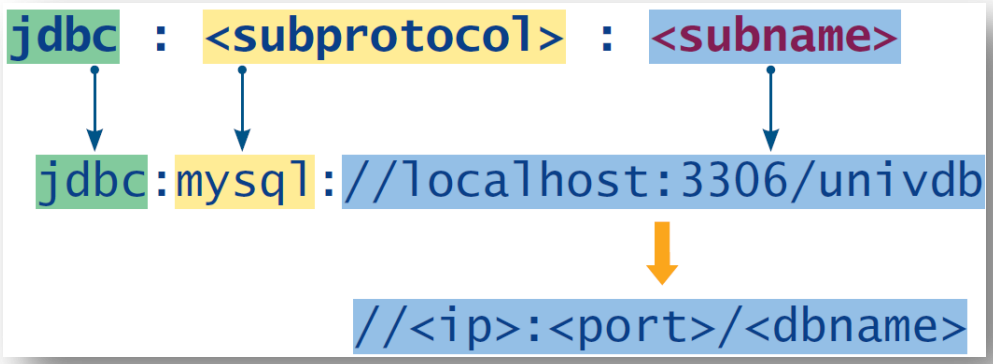
- **DriverManager.getConnection()을 호출**
  - 클래스 DriverManager의 static 메소드인 getConnection()
    - 등록된 드라이버 중에서 주어진 URL로 데이터베이스에 연결할 수 있는 드라이버를 찾아서
    - Driver 클래스의 메소드 connect()를 호출하고
    - 결과인 Connection 객체를 반환

```
//접속할 정보인 URL 지정
String dbURL = "jdbc:mysql://localhost:3306";
//데이터베이스에 연결
Connection con = DriverManager.getConnection(dbURL, "root", " ");
                                              URL   사용자이름  암호
```

- **데이터베이스 URL 정보**
  - JDBC 프로토콜의 의미하는 jdbc로 시작하며
    - 다음에 <subprotocol>, <subname>을 기술
      - 세 부분을 콜론(:)으로 구분

데이터베이스 URL 정보

- JDBC 프로토콜의 의미하는 jdbc로 시작하며
  - 다음에 <subprotocol>, <subname>을 기술
    - 세 부분을 콜론(:)으로 구분



다양한 DBMS 연결 방법

MySQL의 URL에서 <subname> 요소

표현 요소	표현 내용	다른 표현	의미
//<host or ip>	//localhost	//203.214.34.67	MySQL이 실행되는 DBMS 서버를 지정, IP주소 또는 도메인 이름
:<port>	:3306	:3307	DBMS 서비스 포트 번호로, 3306으로 서비스된다면 생략 가능
/<dbname>	/univdb	/mydb	접속할 데이터베이스 이름

DBMS에 따른 데이터베이스 URL

DBMS 종류	JDBC URL
ORACLE	"jdbc:oracle.thin:@localhost:1521:ORA"
MS SQLServer	"jdbc:microsoft:sqlserver://localhost:1433"
mSQL	"jdbc:msql://localhost:1114/univdb"
MySQL	"jdbc:mysql://localhost:3306/univdb"
ODBC	"jdbc:odbc:mydb"

DB 연결 설정

JDBC 드라이버 로드

```
try {
   //Class.forName("com.mysql.jdbc.Driver");
     Class.forName("oracle.jdbc.driver.OracleDriver");
} catch (ClassNotFoundException e) {
   e.printStackTrace();
}
```

- Class.forName()은 동적으로 자바 클래스 로딩
- MySQL의 JDBC 드라이버 클래스인 *com.mysql.jdbc.Driver* 로드
- 드라이버의 클래스 이름은 DB의 드라이버마다 다를 수 있으므로 JDBC 드라이버 문서 참조할 것
- 자동으로 드라이버 인스턴스를 생성하여 DriverManager에 등록
- 해당 드라이버가 없으면 ClassNotFoundException 발생

자바 응용프로그램과 JDBC의 연결

- 연결

```
try {
   // String url="jdbc:mysql://localhost:3306/sampledb"; //MySQL;
   String url="jdbc:oracle.thin:@localhost:1521:XE";
   String user="pgm";
   String password="1234:;
    Connection conn = DriverManager.getConnection(url ,user, password");
} catch (SQLException e) {
   e.printStackTrace();
}
```

  - DriverManager는 자바 어플리케이션을 JDBC 드라이버에 연결시켜 주는 클래스로서 getConnection() 메소드로 DB에 연결하여 Connection 객체 반환
  - getConnection에서 jdbc: 이후에 지정되는 URL의 형식은 DB에 따라 다르므로 JDBC 문서를 참조
    - MySQL 서버가 같은 컴퓨터에서 동작하므로 서버 주소를 localhost로 지정
    - MySQL의 경우 디폴트로 3306 포트를 사용
    - sampledb는 앞서 생성한 DB의 이름
  - User, password는 DB에 로그인할 계정과 패스워드

예제1 : 데이터베이스 연결하는 JDBC 프로그램 작성

JDBC를 이용하여 oracle 데이터베이스에 연결하는 자바 응용프로그램을 작성

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.SQLException;

public class JDBC_Ex1 {
   public static void main (String[] args) {
      String url="jdbc:oracle.thin:@localhost:1521:XE";
      String user="pgm";
      String password="1234:;

       try {
          Class.forName("com.mysql.jdbc.Driver");
          Connection conn = DriverManager.getConnection(url, user, password);
          System.out.println("DB 연결 완료");
       } catch (ClassNotFoundException e) {
          System.out.println("JDBC 드라이버 로드 에러");
       } catch (SQLException e) {
          System.out.println("DB 연결 오류");
       }
   }
}
```

DB 연결 오류 또는 DB 연결 완료

데이터베이스 사용

Stetement 클래스

- SQL문을 실행하기 위해서는 Statement 클래스를 이용
- 주요 메소드

메소드	설명
ResultSet executeQuery(String sql)	주어진 SQL문을 실행하고 결과는 ResultSet 객체에 반환
int executeUpdate(String sql)	INSERT, UPDATE, 또는 DELETE과 같은 SQL문을 실행하고, SQL문 실행으로 영향을 받은 행의 개수나 0을 반환
void close()	Statement 객체의 데이터베이스와 JDBC 리소스를 즉시 반환

- 데이터 검색을 위해 executeQuery() 메소드 사용
- 추가, 수정, 삭제와 같은 데이터 변경은 executeUpdate() 메소드 사용

데이터베이스 사용

- **ResultSet 클래스**
  - SQL문 실행 결과를 얻어오기 위해서는 ResultSet 클래스를 이용
  - 현재 데이터의 행(레코드 위치)을 가리키는 커서(cursor)를 관리
  - 커서의 초기 값은 첫 번째 행 이전을 가리킴
  - 주요 메소드

메소드	설명
boolean first()	커서를 첫 번째 행으로 이동
boolean last()	커서를 마지막 행으로 이동
boolean next()	커서를 다음 행으로 이동
boolean previous()	커서를 이전 행으로 이동
boolean absolute(int row)	커서를 지정된 행으로 이동
boolean isFirst()	첫 번째 행이면 true 반환
boolean isLast()	마지막 행이면 true 반환
void close()	ResultSet 객체의 데이터베이스와 JDBC 리소스를 즉시 반환
Xxx getXxx(String columnLable)	Xxx는 해당 데이터 타입, 현재 행에서 지정된 열 이름에 해당하는 데이터를 반환. 예: int형 데이터를 읽는 메소드는 getInt()
Xxx getXxx(int columnIndex)	Xxx는 해당 데이터 타입, 현재 행에서 지정된 열 인덱스에 해당하는 데이터를 반환. 예:, int형 데이터를 읽는 메소드는 getInt()

데이터베이스 사용

- **테이블의 모든 데이터 검색**

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

  - Statement의 executeQuery()는 SQL문의 실행하여 결과를 넘겨줌
  - 위의 SQL문의 student 테이블에서 모든 행의 모든 열을 읽어 결과를 rs에 저장

- **특정 열만 검색**

```
ResultSet rs = stmt.executeQuery("select name, id from student");
```

  - 특정 열만 읽을 경우는 select문을 이용하여 특정 열의 이름 지정

- **조건 검색**

```
rs = stmt.executeQuery("select name, id, dept from student where id='0494013'");
```

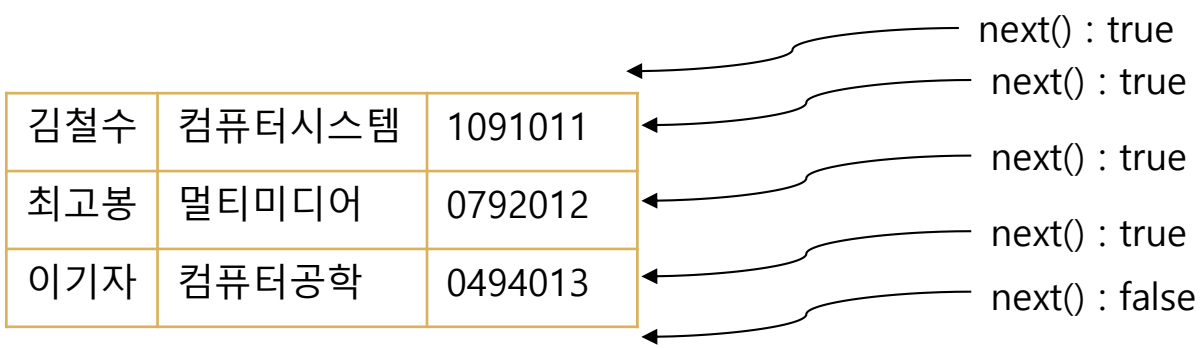
  - select문에서 where절을 이용하여 조건에 맞는 데이터 검색

데이터베이스 사용

검색된 데이터의 사용

```
while (rs.next()) {
    System.out.println(rs.getString("name"));
    System.out.println(rs.getString("id"));
    System.out.println(rs.getString("dept"));
}
rs.close();
```

- Statement객체의 executeQuery() 메소드
  - ResultSet 객체 반환
- ResultSet 인터페이스
  - DB에서 읽어온 데이터를 추출 및 조작할 수 있는 방법 제공
- next() 메소드
  - 다음 행으로 이동



데이터베이스 사용

- ResultSet의 getXXX() 메소드
  - 해당 데이터 타입으로 열 값을 읽어옴
  - 인자로 열의 이름이나 인덱스를 줄 수 있음
  - DB의 데이터 타입에 해당하는 자바 데이터 타입으로 데이터를 읽어야 함.
  - 모든 데이터 타입에 대해 getString() 메소드로 읽을 수 있으나 사용할 때는 해당 데이터 타입으로 변환해서 사용

- ResultSet에서 모든 데이터를 다 읽어들인 후에는 close()를 호출하여 자원 해제

예제 2 : 데이터 검색과 출력

student 테이블의 모든 데이터를 출력하는 프로그램과 이름이 “이기자”인 학생의 데이터를 출력하는 프로그램을 작성하시오.

```
import java.io.UnsupportedEncodingException;
import java.sql.*;
public class JDBC_Ex2 {
   public static void main (String[] args) {
      Connection conn;
      Statement stmt = null;
      try {
         Class.forName(“oracle.jdbc.driver.OracleDriver");
         conn = DriverManager.getConnection(“oracle.thin:@localhost:1521:xe", “pgm”,”1234");
         System.out.println("DB 연결 완료");
         stmt = conn.createStatement();
         ResultSet srs = stmt.executeQuery("select * from student");
         printData(srs, "name", "id", "dept");
         srs = stmt.executeQuery("select name, id, dept from student where name='이기자'");
         printData(srs, "name", "id", "dept");
      } catch (ClassNotFoundException e) {
         System.out.println("JDBC 드라이버 로드 에러");
      } catch (SQLException e) {
         System.out.println("SQL 실행 에러");
      } catch (UnsupportedEncodingException e) {
         System.out.println("지원되지 않는 인코딩 타입");
      }
   }
```

예제 17-2 : 데이터 검색과 출력(소스 계속)


```
    private static void printData(ResultSet srs, String col1, String col2, String col3)
            throws UnsupportedEncodingException, SQLException {
        while (srs.next()) {
            if (!col1.equals(""))
                System.out.print(new String(srs.getString("name").getBytes("ISO-8859-1")));
            if ((!col2.equals(""))
                System.out.print("₩t|₩t" + srs.getString("id"));
            if ((!col3.equals(""))
                System.out.println("₩t|₩t" + new String(srs.getString("dept").getBytes("ISO-8859-1")));
            else
                System.out.println();
        }
    }
}
```

```
DB 연결 완료
이기자  |  0494013  |  컴퓨터공학
김철수  |  1091011  |  컴퓨터시스템
이기자  |  0494013  |  컴퓨터공학
```

데이터의 변경

레코드 추가

```
stmt.executeUpdate("insert into student (name, id, dept) values('" +
                new String("아무개".getBytes(), "ISO-8859-1") +
                "', '0893012', '" +
                new String("컴퓨터공학".getBytes(),"ISO-8859-1") +"');");
```

- DB에 변경을 가하는 조작은 executeUpdate() 메소드 사용
- SQL문 수행으로 영향을 받은 행의 개수 반환

데이터 수정

```
stmt.executeUpdate("update student set id='0189011' where name='"+
                new String("아무개".getBytes(), "ISO-8859-1") +"'");
```

- where문의 MySQL에서 처리되므로 문자열을 Unicode에서 ISO-8859-1로 변환에 주의

데이터 삭제

```
stmt.executeUpdate("delete from student where name='"+
                new String("아무개".getBytes(), "ISO-8859-1") +"'");
```

예제 3 : 데이터의 변경

student 테이블에 새로운 학생 정보를 추가하고, 새로 생성된 학생의 정보를 수정한 후에 다시 삭제하는 코드를 작성하시오. 데이터가 변경될 때마다 모든 테이블의 내용을 출력하도록 하시오.

```
import java.io.*;
import java.sql.*;
public class JDBC_Ex3 {
   public static void main (String[] args) {
      Connection conn;
      Statement stmt = null;
      try {
         Class.forName("oracle.jdbc.driver.OracleDriver");
         conn = DriverManager.getConnection("oracle.thin:@localhost:1521:xe", "pgm","1234");
         System.out.println("DB 연결 완료");
         stmt = conn.createStatement();
         stmt.executeUpdate("insert into student (name, id, dept) values( ' 아무개','0893012','컴퓨터공학'");
         printTable(stmt);
         stmt.executeUpdate("update student set id='0189011' where name='아무개'");
         printTable(stmt);
         stmt.executeUpdate("delete from student where name='아무개'");
         printTable(stmt);
      } catch (ClassNotFoundException e) {
         System.out.println("JDBC 드라이버 로드 에러");
      } catch (SQLException e) {
         System.out.println("SQL 실행 에러");
      } catch (UnsupportedEncodingException e) {
         System.out.println("지원되지 않는 인코딩 타입");
      }
   }
```

예제 17-3 : 데이터의 변경(소스 계속)


```
private static void printTable(Statement stmt) throws SQLException,
UnsupportedEncodingException {
      ResultSet srs = stmt.executeQuery("select * from student");
      while (srs.next()) {
         System.out.print(new String(srs.getString("name")));
         System.out.print("₩t|₩t" + srs.getString("id"));
         System.out.println("₩t|₩t" + new String(srs.getString("dept")));
      }
   }
}
```

```
DB 연결 완료
이기자 | 0494013 | 컴퓨터공학
아무개 | 0893012 | 컴퓨터공학
김철수 | 1091011 | 컴퓨터시스템
아무개 | 0189011 | 컴퓨터공학
이기자 | 0494013 | 컴퓨터공학
김철수 | 1091011 | 컴퓨터시스템
이기자 | 0494013 | 컴퓨터공학
김철수 | 1091011 | 컴퓨터시스템
```