JDBC

JDBC

- 프로그래밍 언어와 관계형 데이터베이스 연동
  - ✓ 프로그래밍 언어 와 데이터베이스 사이의 드라이버를 이용해서 프로그래밍 언어 코드 안에 SQL을 삽입해서 사용하는 방식
  - ✓ 프레임워크를 이용하는 방법
    - SQL Mapper 방식: SQL을 별도로 만들고 프로그래밍 언어의 코드가 호출해서 사용하는 방식으로 별도의 파일에 만들기도 하고 코드에 추가해서 사용하는 방식도 있음
    - ORM(Object Relation Mapping): 테이블의 하나의 행 과 프로그래밍 언어의 인스턴스와 1:1로 매핑해서 사용하는 방식

JDBC

- JDBC(Java Database Connectivity)
  - ✓ 자바 프로그램과 데이터베이스에 대한 인터페이스로 데이터베이스에 접근하는 방법과 SQL 명령들을 전달해서 수행할 수 있도록 해주는 방법을 제공하는 라이브러리
  - ✓ JDBC 인터페이스와 JDBC 드라이버로 구성
  - ✓ 응용프로그램에서는 SQL문 만들어 JDBC Interface를 통해 전송하고 실제 구현 클래스인 JDBC 드라이버에서 DBMS에 접속을 시도하여 SQL문을 전송
  - ✓ DBMS는 SQL 구문을 실행 한 후 결과를 JDBC Driver와 JDBC Interface에게 전달하고 이를 다시 응용프로그램으로 전달해서 SQL문의 결과를 사용
  - ✓ JDBC의 역할은 Application과 DBMS의 Bridge 역할
  - ✓ JDBC 프로그래밍 방법
    - 데이터베이스와 연결하는 드라이버 파일을 찾아서 로드
    - 연결을 관리하는 Connection 인스턴스 생성
    - sql 구문을 처리할 Statement, PreparedStatement, CallableStatement 인스턴스 생성
    - 구문 실행 – select 구문 인 경우 ResultSet 인스턴스를 통한 sql 결과 처리
    - 접속 종료

JDBC

- 드라이버 로드 및 데이터베이스 접속과 접속 종료
  - ✓ Class.forName(드라이버 이름); => 드라이버 로드(ClassNotFoundException이 발생할 수 있는데 이 경우는 jdbc 연결 파일을 추가하지 않았거나 클래스 이름이 잘못된 경우)
  - ✓ Connection Connection변수명= DriverManager.getConnection(데이터베이스이름, "아이디","비밀번호"); => 접속(SQLException 이 발생할 수 있는데 이 때는 데이터베이스 이름이나 아이디 및 비밀번호 와 데이터베이스 활성화 여부 확인)
  - ✓ Connection변수명.close(); => 접속 종료

JDBC

- 접속 에러
  - ✓ No suitable driver found for jdbc:oracle:thin:@localhost:1521:xe: OJDBC드라이버에 해당하는 jar 파일이 Build Path에 제대로 추가됐는지 확인
  - ✓ IO 오류: The Network Adapter could not establish the connection : 데이터베이스 접속 url을 확인하고 데이터베이스가 서비스 중인지 확인
  - ✓ ORA-01017: invalid username/password; logon denied : 계정 정보를 확인

JDBC

- Connection 인스턴스의 트랜잭션 관리 메서드
  - ✓ setAutoCommit(true or false)
  - ✓ rollback()
  - ✓ commit()
  - ✓ 기본은 autoCommit 이며 직접 관리하기 위해서 setAutoCommit에 false를 전달 한 후 사용

JDBC

- SQL 실행 클래스
  - ✓ Statement: 데이터베이스에 쿼리를 보내기 위한 클래스
    - Connection 인스턴스의 createStatement()로 생성
    - 인스턴스는 Connection 인스턴스의 연결 정보를 가져와서 DB에 접근하므로 이 인스턴스를 사용하기 위해서는 Connection 인스턴스가 먼저 존재해야 함
    - insert, update, delete 구문실행: 영향 받은 행의 개수나 0을 리턴
      - 정수변수 = statement변수.executeUpdate(dml문장) ;
      - 리턴 값이 0이면 쿼리가 문법에는 맞지만 쿼리의 결과로 영향 받은 행이 없는 것이고 양수 값이면 영향 받은 행의 개수가 있는 것이므로 쿼리를 수행하고 결과로 변경된 행이 있는 것
    - select 구문 실행
      - ResultSet 변수명 = statement변수.executeQuery(select문장) ;
    - 종료: statement변수.close();

JDBC

- SQL 실행 클래스
  - ✓ PreparedStatement
    - SQL의 틀을 미리 정해 놓고, 나중에 값을 지정하는 방식
    - PreparedStatement 의 일반적 사용 - 값을 대입하는 곳에 ?를 설정하고 나중에 대입

```
PreparedStatement pstmt = conn.prepareStatement(
   "insert into MEMBER (MEMBERID, NAME, EMAIL) values (?, ?, ?)");
pstmt.setString(1, “ggangpae1"); // 첫번째 물음표의 값 지정
pstmt.setString(2, " 박문석");  // 두번째 물음표의 값 지정
pstmt.executeUpdate();
```

    - 쿼리 실행 관련 메서드
      - ResultSet executeQuery() - SELECT 쿼리를 실행할 때 사용되며 ResultSet을 결과값으로 리턴
      - int executeUpdate() - INSERT, UPDATE, DELETE 쿼리를 실행할 때 사용되며, 실행 결과 변경된 레코드의 개수를 리턴
    - PreparedStatement 의 장점
      - DBMS가 PreparedStatement와 관련된 쿼리 파싱 회수 감소
      - 작은 따옴표 등 값에 포함된 특수 문자의 처리가 수월
      - 문자열 연결에 따른 코드의 복잡함 감소

JDBC

- SQL 실행 클래스
  - ✓ PreparedStatement
    - 파라미터 설정 메서드

?의 값 바인딩 메서드	설명
setString(int index, String x)	지정한 인덱스의 파라미터 값을 문자열 x로 지정
setInt(int index, int x)	지정한 인덱스의 파라미터 값을 int 값 x로 지정
setLong(int index, long x)	지정한 인덱스의 파라미터 값을 long 값 x로 지정
setDouble(int index, double x)	지정한 인덱스의 파라미터 값을 double 값 x로 지정
setFloat(int index, float x)	지정한 인덱스의 파라미터 값을 float 값 x로 지정
setTimestamp(int index, Timesetamp x)	지정한 인덱스의 값을 SQL TIMESTMAP 타입을 나타내는 java.sql.Timestamp 타입으로 지정
setDate(int index, Date x)	지정한 인덱스의 값을 SQL DATE 타입을 나타내는 java.sql.Date 타입으로 지정
setTime(int index, Time x)	지정한 인덱스의 값을 SQL TIME 타입을 나타내는 java.sql.Time 타입으로 지정

JDBC

- ❖ SQL 실행 클래스
  - ✓ PreparedStatement
    - ● 날짜 와 시간
      - ◆ 날짜만 저장할 때는 java.sql.Date 타입으로 setDate 함수를 이용해서 바인딩 할 수 있고 가져올 때는 getDate로 가져올 수 있음
      - ◆ 시간만 저장할 때도 방법은 동일한데 java.sql.Time 클래스를 이용
      - ◆ 날짜와 시간을 저장할 때 java.sql.TimeStamp 클래스를 이용할 수 있음
      - ◆ 웹 프로그래밍이나 스마트 폰 프로그래밍이 아니라면 날짜를 입력받을 수 있는 방법이 없기 때문에 날짜 데이터를 만들려면 문자열로 입력받은 후 날짜 데이터로 변경해서 작업
      - ◆ 날짜 나 시간 데이터를 문자열로 저장하기도 함

JDBC

- ❖ SQL 실행 클래스
  - ✓ PreparedStatement
    - ● BLOB
      - ◆ 파일의 내용을 저장하기 위해서 사용하는 데이터 타입
      - ◆ 파일을 생성해두고 이름을 저장해서 사용해도 되지만 파일 이름을 저장하게 되면 파일을 이동하거나 파일의 이름을 마음대로 변경할 수 없음
      - ◆ BLOB는 스트림을 이용해서 저장하고 읽어옴
      - ◆ 데이터베이스에 저장하기 위해서 BLOB를 바인딩 하기 위해서는 setBinaryStream(인덱스, FileInputStream fis, int length) 메서드를 이용
      - ◆ 데이터베이스에 저장된 blob 테이터를 읽어오기 위해서는 InputStream getBinaryStream(인덱스 또는 컬럼이름) 메서드를 이용해서 스트림을 받아서 처리

JDBC

- SQL 실행 클래스
  - ✓ CallableStatement
    - 프로시저를 실행할 수 있는 클래스
    - 인스턴스를 생성할 때 connection인스턴스.getCall(call 프로시저이름(매개변수));
      또는 conn.prepareCall(call 프로시저이름(?))의 형태로 생성
    - executeQuery 메서드로 프로시저를 실행

JDBC

- ResultSet
  - ✓ close()
  - ✓ getXXX(컬럼 인덱스):XXX 타입으로 인덱스에 해당하는 데이터 가져오기
  - ✓ getXXX(컬럼 명): XXX 타입으로 컬럼 명에 해당하는 데이터 가져오기
  - ✓ next(): 다음 행으로 진행(없으면 false를 리턴)
  - ✓ getColumnCount()
  - ✓ absolute(int row)
  - ✓ beforeFirst()
  - ✓ afterLast()
  - ✓ first()
  - ✓ last()
  - ✓ previous()
  - ✓ ResultSetMetaData getMetaData()

JDBC

- ResultSet
  - ✓ ResultSetMetaData
    - 열의 개수: int getColumnCount()
    - 열에 대한 정보: String getColumnName(int column)
    - 열의 제목을 출력하는 데 적절한 제목: String getColumnLabel(int column)
    - 열을 문자열로 출력하기 위한 최대 문자 수: int getColumnDisplaySize(int column)
    - 열의 SQL Type을 돌려줌: int getColumnType(int column)
    - 열의 SQL TypeName을 돌려줌: String getColumnTypeName(int column)

JDBC

- ResultSet
  - ✓ ResultSet의 getXXX() 메서드
    - 해당 데이터 타입으로 열 값을 읽어옴
    - 매개변수로 열의 이름(문자열)이나 인덱스(정수)를 줄 수 있음
    - DB의 데이터 타입에 해당하는 자바 데이터 타입으로 데이터를 읽어야 함.
    - 모든 데이터 타입에 대해 getString() 메서드로 읽을 수 있음
  - ✓ ResultSet에서 모든 데이터를 다 읽어들인 후에는 close()를 호출하여 자원 해제
  - ✓ 1개 데이터 읽기

```
if(rs.next()){
 데이터 읽기
}
```

  - ✓ 여러 개의 데이터 읽기

```
if(rs.next()){
 do{
            데이터 읽기
 }while(rs.next());
}
```

JDBC

- DTO & DAO Pattern
  - ✓ DTO(Data Transfer Object) 패턴: 데이터베이스 와의 연동에 필요한 데이터를 나타내는 클래스를 별도로 생성해서 사용하는 패턴 – Variable Object(VO) 또는 Domain Class 라고도 함
    - 생성방법
      - 테이블에서 저장할 컬럼들을 인스턴스 변수로 선언
      - 필요에 따라 생성자 재정의
      - Getter & Setter 를 생성
      - 디버깅을 위해서 toString()을 재정의
      - 인스턴스 단위 전송이 필요한 경우 Serializable 인터페이스 implements
  - ✓ DAO(Data Access Object) 패턴: 데이터베이스와의 연동에 필요한 코드를 묶어서 별도의 클래스로 생성해서 사용하는 패턴으로 Singleton 패턴과 템플릿 메서드 패턴과 함께 사용하는 것을 권장

Maria DB

- Maria DB
  - ✓ 드라이버 다운로드: https://downloads.mariadb.com/Connectors/java/connector-java-2.4.0/
  - ✓ 드라이버 이름: org.mariadb.jdbc.Driver
  - ✓ 데이터베이스 이름: jdbc:mariadb://ip:포트번호/데이터베이스이름
  - ✓ 데이터에 한글이 있는 경우 데이터베이스 이름 에 ?useUnicode=true&characterEncoding=utf8을 추가
  - ✓ 타임 존 에러가 발생하면 &serverTimezone=UTC 도 추가
  - ✓ ip는 로컬 컴퓨터이면 localhost
  - ✓ 포트 번호는 기본적으로 3306
  - ✓ Maria DB의 JDBC 드라이버를 다운로드 받아서 자바가 설치된 경로의 자바 설치폴더/ jre버전/lib/ext에 복사하거나 프로젝트의 BuildPath에 추가

Maria DB

- 데이터베이스 접속 확인
  - ✓ Java Application 프로젝트 생성
  - ✓ 프로젝트에 MySQL Driver를 복사하고 Build Path에 추가

Maria DB

❖ 데이터베이스 접속 확인

✓ 프로젝트에 main 메서드를 소유한 클래스를 생성하고 작성

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

public class JDBCMariaDB {
    static final String db = "adam";
    static final String id = "root";
    static final String pass = "wnddkd";
    static String url = "jdbc:mariadb://localhost:3306/" + db;
    public static void main(String[] args) {
        System.out.println("Maria DB JDBC 드라이버 로딩중...");
        try {
            // 1. Maria DB 데이타베이스 드라이버 로딩
            Class.forName("org.mariadb.jdbc.Driver");
            System.out.println("Maria DB JDBC 드라이버 로딩 성공...");

        } catch (ClassNotFoundException e) {
            System.out.println("Maria DB JDBC 드라이버 로딩 실패...");
            System.out.println(e.getMessage());
            System.exit(0);
        }
```

Maria DB

- 데이터베이스 접속 확인
  - ✓ 프로젝트에 main 메서드를 소유한 클래스를 생성하고 작성

```
            try(Connection connection = DriverManager.getConnection(url, id, pass)) {
                    System.out.println("접속 / 로그인 성공");
            } catch (Exception e) {
                    System.out.println("접속 / 로그인 실패");
                    System.out.println(e.getMessage());
            }
        }
    }
```

Maria DB

- properties 파일 활용
  - ✓ db.properites 파일을 프로젝트 파일에 생성하고 작성

```
driver=org.mariadb.jdbc.Driver
url=jdbc:mariadb://localhost:3306/adam
id=root
password=wnddkd
```

Maria DB

- properties 파일 활용
  - ✓ db.properites 파일을 프로젝트 파일에 생성하고 작성

```
import java.io.File;
import java.io.FileInputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Properties;

public class PropertyRead {
    public static void main(String[] args) {
            InputStream is = null;
            Connection connection = null;
            try{
                    File file = new File("db.properties");
                    is = new FileInputStream(file);
                    Properties properties = new Properties();
                    properties.load(is);
                    String driver = properties.getProperty("driver");
                    String url = properties.getProperty("url");
                    String id = properties.getProperty("id");
                    String pass = properties.getProperty("password");
                    Class.forName(driver);
                    System.out.println("드라이버 클래스 로드");
```

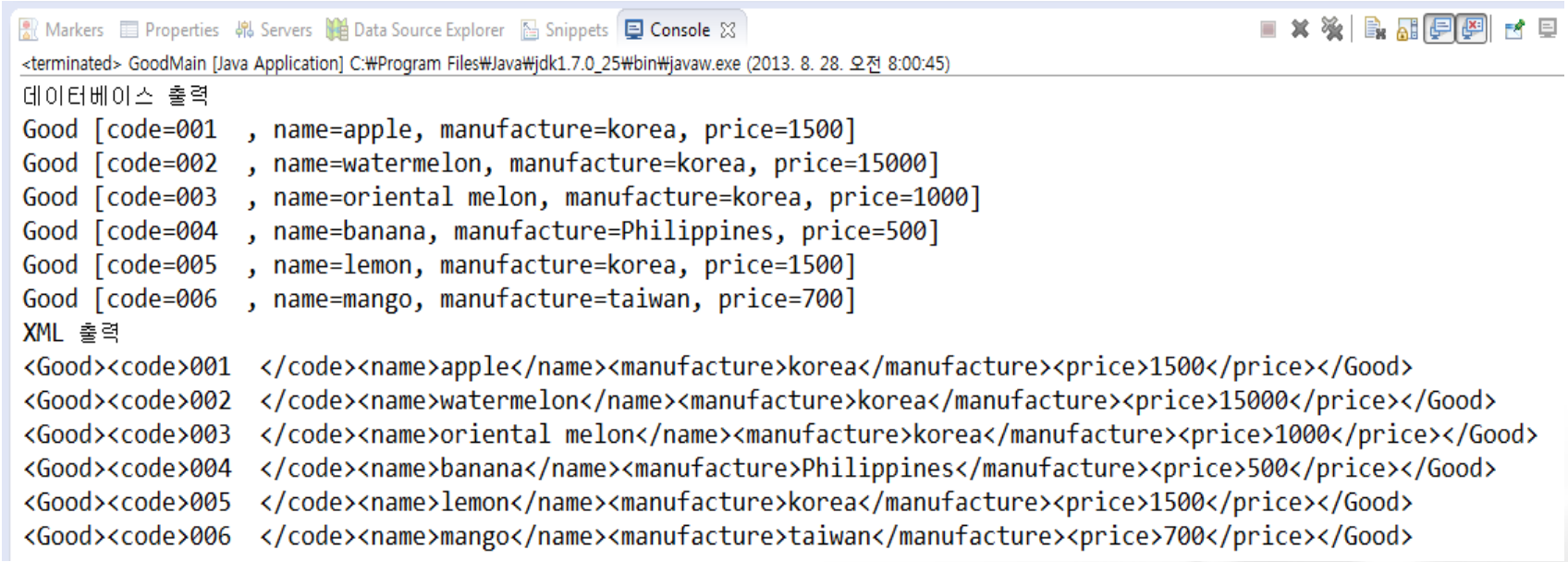
Maria DB

- properties 파일 활용
  - ✓ db.properites 파일을 프로젝트 파일에 생성하고 작성

```
                connection = DriverManager.getConnection(url, id, pass);
                System.out.println("접속 / 로그인 성공");
            } catch (Exception e) {
                System.out.println("데이터베이스 접속 실패");
                System.out.println(e.getMessage());
            }finally {
                try {
                    if(is != null) {
                        is.close();
                    }
                    if(connection != null) {
                        connection.close();
                    }
                }catch (Exception e) {
                    System.out.println(e.getMessage());
                }
            }
        }
    }
```

Maria DB

❖ DAO & DTO Pattern



Maria DB

- DAO & DTO Pattern
  - ✓ 데이터베이스에 접속해서 샘플 데이터 생성

```
create database adam;

use adam;

create table goods (
    code     char(5) not null,
    name     varchar(50) not null,
    manufacture    varchar(20),
    price     int not null,
    primary key(code)
)ENGINE=MyISAM AUTO_INCREMENT=1 DEFAULT CHARSET=utf8;

insert into goods values('001', 'apple', 'korea', 1500);
insert into goods values('002', 'watermelon', 'korea', 15000);
insert into goods values('003', 'oriental melon', 'korea', 1000);
insert into goods values('004', 'banana', 'philippines', 500);
insert into goods values('005', 'lemon', 'korea', 1500);
insert into goods values('006', 'mango', 'taiwan', 700);

select * from goods;

commit;
```

Maria DB

- DAO & DTO Pattern
  - ✓ Goods 테이블의 데이터를 저장하기 위한 DTO 클래스인 Good 클래스를 생성

```
public class Good{
    private String code;
    private String name;
    private String manufacture ;
    private int price;

    public Good() {
            super();
    }
    public Good(String code, String name, String manufacture, int price) {
            super();
            this.code = code;
            this.name = name;
            this.manufacture = manufacture;
            this.price = price;
    }
```

Maria DB

- DAO & DTO Pattern
  - ✓ Goods 테이블의 데이터를 저장하기 위한 DTO 클래스인 Good 클래스를 생성

```
public String getCode() {
        return code;
}
public void setCode(String code) {
        this.code = code;
}
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getManufacture() {
        return manufacture;
}
public void setManufacture(String manufacture) {
        this.manufacture = manufacture;
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ Goods 테이블의 데이터를 저장하기 위한 DTO 클래스인 Good 클래스를 생성

```
    public int getPrice() {
            return price;
     }
     public void setPrice(int price) {
            this.price = price;
     }
     @Override
     public String toString() {
            return "Good [code=" + code + ", name=" + name + ", manufacture="
                                    + manufacture + ", price=" + price + "]";
     }
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 클래스의 메서드의 원형을 가진 GoodDAO 인터페이스를 생성

```
import java.util.List;

public interface GoodDAO {
    //테이블의 모든 데이터를 가져오는 메서드
    public List<Good> allSelect();

    //기본키를 가지고 하나의 데이터를 가져오는 메서드
    public Good getGood(String code);

    //하나의 데이터를 삽입하는 메서드
    public Boolean insertGood(Good good);

    //하나의 데이터를 수정하는 메서드
    public Boolean updateGood(Good good);

    //하나의 데이터를 삭제하는 메서드
    public Boolean deleteGood(String code);
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
public class GoodDAOImpl implements GoodDAO {
    // 싱글톤 클래스를 만드는 코드
    private GoodDAOImpl() {}

    private static GoodDAOImpl obj;

    public static GoodDAOImpl getInstance() {
            if (obj == null)
                    obj = new GoodDAOImpl();
            return obj;
    }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
private static String driver;
private static String url;
private static String id;
private static String pass;
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
static {
        File file = new File("db.properties");
        try {
                FileInputStream is = new FileInputStream(file);
                Properties properties = new Properties();
                properties.load(is);
                driver = properties.getProperty("driver");
                url = properties.getProperty("url");
                id = properties.getProperty("id");
                pass = properties.getProperty("password");

                Class.forName(driver);
        }catch(Exception e) {
                System.out.println(e.getLocalizedMessage());
                System.out.println("시스템을 종료합니다.");
                System.exit(0);
        }
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
// 데이터베이스 연동 메서드에서 사용할 변수 선언
private Connection con;
private PreparedStatement pstmt;

// 데이터베이스 연결 메서드
private boolean connect() {
        boolean result = false;
        try {
                con = DriverManager.getConnection(url, id, pass);
                result = true;
        } catch (Exception e) {
                e.printStackTrace();
        }
        return result;
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
//con과 pstmt 연결 해제하는 메서드
private void close() {
        try {
                if (pstmt != null)
                        pstmt.close();
                if (con != null)
                        con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
// Goods 테이블의 모든 데이터를 읽어서 리턴하는 메서드
public List<Good> allSelect() {
        // 데이터를 저장해서 리턴할 인스턴스 생성
        List<Good> list = new ArrayList<Good>();
        // select 구문의 결과를 저장할 변수 생성
        ResultSet rs = null;
        try {
                // 연결에 성공하면
                if (connect()) {
                        // SQL을 실행할 수 있는 인스턴스 생성
                        pstmt = con.prepareStatement("select * from
goods");
                        // select 구문의 결과를 rs에 저장
                        rs = pstmt.executeQuery();
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
                                        // 검색된 데이터를 읽어서 list에 저장
                                        if (rs.next()) {
                                                do {
                                        Good good = new Good();
                                        good.setCode(rs.getString("code"));
                                        good.setName(rs.getString("name"));
                                        good.setManufacture(rs.getString("manufacture"));
                                        good.setPrice(rs.getInt("price"));
                                        list.add(good);
                                                } while (rs.next());
                                        }
                                }
                        } catch (Exception e) {
                                e.printStackTrace();
                        } finally {
                                try {
                                        if (rs != null) rs.close();
                                        close();
                                } catch (Exception e) {}
                        }
                        return list;
                }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
// code(Primary Key)를 받아서 goods 테이블에서 데이터를 검색한 후 리턴하는 메서드
// 데이터는 없거나 1개만 나올 수 있습니다.
// 없을 때는 null을 리턴하고 있을 때는 1개를 리턴
// 목록에서 제목을 눌렀을 때 상세보기를 하거나 회원정보 수정을 눌렀을 때 회원정보 보여주는 화면을 만들 때 사용
public Good getGood(String code) {
        // 검색된 정보를 저장해서 리턴하기 위한 변수
        Good good = null;
        // 검색된 데이터 정보를 저장할 변수
        ResultSet rs = null;
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
try{
        //데이터베이스 연결
        if(connect()){
                //SQL 실행 인스턴스 생성
                pstmt = con.prepareStatement(
                "select * from goods where trim(code) = ?");
                //물음표에 데이터 바인딩
                pstmt.setString(1, code);
                //select 구문을 실행해서 결과를 rs에 저장
                rs = pstmt.executeQuery();
                //검색된 데이터는 1개가 넘지 않습니다.
                if(rs.next()){
                        good = new Good();
                        good.setCode(rs.getString("code").trim());
                        good.setName(rs.getString("name"));
                good.setManufacture(rs.getString("manufacture"));
                        good.setPrice(rs.getInt("price"));
                }
        }
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
                catch(Exception e){
                        e.printStackTrace();
                }
                finally{
                        try{
                                if(rs != null)
                                        rs.close();
                                close();
                        }catch(Exception e){
                                e.printStackTrace();
                        }
                }
                return good;
        }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
//하나의 데이터를 삽입하는 메서드
public Boolean insertGood(Good good){
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(
                                "insert into goods(" +
                                        "code,
name, manufacture,price)"+
                                " values(?,?,?,?)");
                        pstmt.setString(1, good.getCode());
                        pstmt.setString(2, good.getName());
                        pstmt.setString(3, good.getManufacture());
                        pstmt.setInt(4, good.getPrice());
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
            }catch(Exception e){
                        e.printStackTrace();
            }finally{
                        close();
            }
            return result;
      }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
public Boolean updateGood(Good good) {
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(   "update goods " +
                                        "set name = ?, manufacture =
?, price = ? "+
                                        "where trim(code) = ?");

                        pstmt.setString(1, good.getName());
                        pstmt.setString(2, good.getManufacture());
                        pstmt.setInt(3, good.getPrice());
                        pstmt.setString(4, good.getCode());
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
            }catch(Exception e){
                        e.printStackTrace();
            }finally{
                        close();
            }
            return result;
      }
```

Maria DB

- DAO & DTO Pattern
  - ✓ DAO 메서드를 구현할 GoodDAO 인터페이스를 implements 한 GoodDAOImpl 클래스를 생성

```
@Override
public Boolean deleteGood(String code) {
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(
                                        "delete from goods " +
                                        "where trim(code) = ?");
                        pstmt.setString(1, code);
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
        }catch(Exception e){
                e.printStackTrace();
        }finally{
                close();
        }
        return result;
    }
}
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
import java.util.List;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class GoodMain {

    public static void main(String[] args) {
        //키보드 입력 객체 만들기
        Scanner sc = new Scanner(System.in);

        //각 열의 값을 저장할 변수
        String code = null;
        String name = null;
        String manufacture = null;
        String imsi = null;
        int price = 0;

        //데이터베이스 작업 결과를 저장할 변수
        Boolean result = null;
        Good  good = null;
        List<Good> list = null;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
//무한 루프
MainLoop : while(true) {
        //메뉴 출력과 입력 받기
        System.out.print("1.데이터 전체 가져오기 2.데이터 1개 가져오기
3.데이터 추가 4.데이터 수정 5.데이터 삭제 6.프로그램 종료:");
        String menu = sc.nextLine();
        //데이터베이스 연동 인스턴스 만들기
        GoodDAO goodDao = GoodDAOImpl.getInstance();
        switch(menu) {
        case "1":
                list = goodDao.allSelect();
                for(Good temp : list) {
                        System.out.println(temp);
                }
                break;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "2":
                        System.out.print("조회할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good == null) {
                            System.out.println("존재하지 않는 코드입
니다.");
                        }else {
                            System.out.println(good);
                        }
                        break;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "3":
                        System.out.print("삽입할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good == null) {
                            System.out.println("사용 가능한 코드입니
다.");
                            System.out.print("이름을 입력하세요:");
                            name = sc.nextLine();
                            System.out.print("원산지를 입력하세요:");
                            manufacture = sc.nextLine();
                            System.out.print("가격을 입력하세요:");
                            imsi = sc.nextLine();
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        good = new Good(code,
name, manufacture, price);
                                                        result =
goodDao.insertGood(good);
                                                        if(result == true) {
                                                                System.out.println("
데이터 삽입에 성공하셨습니다.");
                                                        }else {
                                                                System.out.println("
데이터 삽입에 실패하셨습니다.");
                                                        }
                                                }catch(Exception e) {
                                                        System.out.println("잘못된 가격
을 입력하셨습니다.");
                                                }
                                        }else {
                                                System.out.println("사용 불가능한 코드입
니다.");
                                        }
                                        break;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "4":
                        System.out.print("수정할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good != null) {
                            System.out.println("수정 가능한 코드입니
다.");
                            System.out.print("이름을 입력하세요:");
                            name = sc.nextLine();
                            System.out.print("원산지를 입력하세요:");
                            manufacture = sc.nextLine();
                            System.out.print("가격을 입력하세요:");
                            imsi = sc.nextLine();
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        good = new Good(code,
name, manufacture, price);
                                                        result =
goodDao.updateGood(good);
                                                        if(result == true) {
                                                                System.out.println("
데이터 수정에 성공하셨습니다.");
                                                        }else {
                                                                System.out.println("
데이터 수정에 실패하셨습니다.");
                                                        }
                                                }catch(Exception e) {
                                                        System.out.println("잘못된 가격
을 입력하셨습니다.");
                                                }
                                        }else {
                                                System.out.println("존재하지 않는 코드라
서 수정할 수 없습니다.");
                                        }
                                        break;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                        case "5":
                                System.out.print("삭제할 데이터의 코드를 입력하세요
:");
                                code = sc.nextLine();
                                good = goodDao.getGood(code);
                                if(good != null) {
                                        System.out.println("삭제 가능한 코드입니
다.");
                                        int r =
JOptionPane.showConfirmDialog(null, "정말로 삭제하시겠습니까?");
                                        if(r == JOptionPane.OK_OPTION) {
                                                result =
goodDao.deleteGood(code);
                                                if(result == true) {
                                                        System.out.println("

삭제에 성공하셨습니다..");
                                                }else {
                                                        System.out.println("

삭제에 실패하셨습니다..");
                                                }
                                        }
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                        }else {
                                                System.out.println("존재하지 않는 코드라
서 삭제할 수 없습니다.");
                                        }
                                        break;
```

Maria DB

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                        case "6":
                                break MainLoop;
                        default :
                                System.out.println("메뉴를 잘못 선택하셨습니다.");
                                break;
                        }
                        System.out.println("엔터를 누르면 메인 메뉴로 이동합니다.");
                        sc.nextLine();
                }
                System.out.println("프로그램을 종료합니다!!!!");
                sc.close();
        }
}
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
create table Item(
code int primary key auto_increment,
title char(50) not null,
category varchar(50),
description text
)engine=innodb default charset=utf8;
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(1, 'Java', 'Programming Language','오픈소스 라이브러리가 많은 범용 프로그래밍 언어');
insert into Item values(2, 'Eclipse', 'IDE', '프로그래밍을 편리하게 할 수 있도록 해주는 오픈 소스 프로그램');

insert into Item values(3, 'Oracle', 'DataBase', '대기업이나 공공기관이 주로 사용하는 관계형 DBMS');
insert into Item values(4, 'MySQL', 'DataBase', '오픈 소스 기반의 관계형 DBMS - Maria DB와 거의 동일');
insert into Item values(5, 'PostgreSQL', 'DataBase', '오픈 소스 기반의 관계형 DBMS');
insert into Item values(6, 'MongoDB', 'DataBase', '가장 많이 언급되는 NoSQL');
insert into Item values(7, 'DBeaver', 'DB Tool', '데이터베이스 사용을 쉽게 해주는 프로그램');

insert into Item values(8, 'HTML', 'Web FrontEnd', '브라우저에 콘텐츠를 표시하기 위한 마크업 언어');
insert into Item values(9, 'CSS', 'Web FrontEnd', 'HTML에 디자인을 추가하기 위한 언어');
insert into Item values(10, 'JavaScript', 'Web FrontEnd', 'HTML을 동적으로 제어하기 위한 언어');
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(11, 'Servlet & JSP', 'Java Web Programming','URL을 이용해서 호출할 수 있는 Java EE 클래스');

insert into Item values(12, 'Spring', 'Framework', '자바 애플리케이션 개발  프레임워크');
insert into Item values(13, 'MyBatis', 'Framework', 'SQL Mapper 프레임워크');
insert into Item values(14, 'Hibernate', 'Framework', 'ORM 프레임워크');

insert into Item values(15, 'Python', 'Programming Language','진입 장벽이 비교적 낮고 오픈소스 라이브러리가 많은 범용 프로그래밍 언어');
insert into Item values(16, 'R', 'Programming Language','통계 계산과 그래픽을 위한 프로그래밍 언어이자 소프트웨어 환경');

insert into Item values(17, 'Jupyter Notebook','IDE', '모든 프로그래밍 언어에 걸쳐 인터랙티브 데이터 과학과 사이언티픽 컴퓨팅을 지원하는 개발 환경');
insert into Item values(18, 'Pycharm', 'IDE', '파이썬 프로그래밍을 편리하게 할 수 있도록 해주는 프로그램');
insert into Item values(19, 'Google Colab', 'IDE', '구글 코랩은 클라우드 기반으로 주피터 노트북 개발환경');
```

연습문제

- 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(20, 'SW 공학', '소프트웨어 개발 기반 기술', '소프트웨어 개발을 공학적으로 접근하기 위한 학문');
insert into Item values(21, 'DataStructure', '소프트웨어 개발 기반 기술', '데이터를 효율적으로 사용하기 위해서 관리하는 기술');
insert into Item values(22, 'Algoritum', '소프트웨어 개발 기반 기술', '문제 해결 능력');
insert into Item values(23, 'Design Pattern','소프트웨어 개발 기반 기술' '클래스를 목적에 맞게 설계한느 기술');
insert into Item values(24, '개발 방법론', '소프트웨어 개발 기반 기술', '효율적인 개발을 위한 방법론');
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(25, 'Data Mining', 'Contents', '대규모로 저장된 데이터 안에서 체계적이고 자동적으로 통계적 규칙이나 패턴을 분석하여 가치있는 정보를 추출하는 과정으로 KDD 라고도 함');
insert into Item values(26, 'AI', 'Contents', '동적 컴퓨팅 환경에 내장된 알고리즘을 생성하고 적용하여 인간의 지능을 모방하는 기초 지능');
insert into Item values(27, 'Machine Learning', 'Contents', '데이터를 이용해서 알고리즘을 컴퓨터가 학습하는 기술');
insert into Item values(28, 'Deep Learning', 'Contents', '사람의 사고방식을 컴퓨터에게 가르치는 기계학습의 한 분야');
insert into Item values(29, 'Reinforcement Learning', 'Contents', '어떤 환경 안에서 정의된 에이전트가 현재의 상태를 인식하여, 선택 가능한 행동들 중 보상을 최대화하는 행동 혹은 행동 순서를 선택하는 방법');
insert into Item values(30, 'Metaverse', 'Contents', '3차원에서 실제 생활과 법적으로 인정되는 활동인 직업, 금융, 학습 등이 연결된 가상 세계');
insert into Item values(31, 'DevOps', 'Contents', '소프트웨어의 개발과 운영의 합성어로서, 소프트웨어 개발자와 정보기술 전문가 간의 소통, 협업 및 통합을 강조하는 개발 환경이나 문화');
insert into Item values(32, 'MLOps', 'Contents', '프로덕션 환경에서 기계 학습 모델을 안정적이고 효율적으로 배포 및 유지 관리하는 것을 목표로 하는 일련의 관행');
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(33, 'Tomcat', 'Web Application Server','apache web server를 이용하는 오픈 소스 프로그램');
insert into Item values(34, 'Docker', 'Container', '리눅스의 응용 프로그램들을 프로세스 격리 기술들을 사용해 컨테이너로 실행하고 관리');
insert into Item values(35, 'Kubernetes', '소프트웨어 관리 툴', '컨테이너화된 워크로드와 서비스를 관리하기 위한 이식성이 있고, 확장가능한 오픈소스 플랫폼');
insert into Item values(36, 'Git', '버전 관리 툴', '컴퓨터 파일의 변경사항을 추적하고 여러 명의 사용자들 간에 해당 파일들의 작업을 조율하기 위한 분산 버전 관리 시스템');
insert into Item values(37, 'Jenkins', 'CI 툴', '소프트웨어 개발 시 지속적 통합(continuous integration) 서비스를 제공하는 툴');
insert into Item values(38, 'JIRA', '이슈 추적 툴', '버그 추적, 이슈 추적, 프로젝트 관리 기능을 제공하는 소프트웨어');
insert into Item values(39, 'Zeplin', '협업 도구', '디자이너 및 개발자를 위한 공동 작업 응용 프로그램');
insert into Item values(40, 'Slack', '클라우드 기반 팀 협업 도구', '모든 대화와 지식을 위한 검색 가능한 로그"(Searchable Log of All Conversation and Knowledge)의 준말');
```

연습문제

❖ 아래 샘플 데이터를 이용해서 CRUD 작업을 수행하는 프로그램을 작성하시오.

```
insert into Item values(41, 'Linux', '운영체제', '오픈 소스 유닉스 계열 운영 체제');
insert into Item values(42, 'AWS', '클라우드 서비스', '아마존에서 제공하는 클라우드 서비스');
insert into Item values(43, 'GCP', '클라우드 서비스', '구글에서 제공하는 클라우드 서비스');
insert into Item values(44, 'Azure', '클라우드 서비스', 'MS에서 제공하는 클라우드 서비스');
insert into Item values(45, 'Tensorflow', 'AI Framework', '구글이 제공하는 AI 프레임워크');
insert into Item values(46, 'Pytorch', 'AI Framework', '페이북에서 제공하는 AI 프레임워크');
insert into Item values(47, 'Keras', 'AI Framework', 'MXNet, Deeplearning4j, 텐서플로,
Microsoft Cognitive Toolkit 또는 Theano 위에서 수행할 수 있음');

commit;

select * from Item;
```

MySQL

- MySql
  - ✓ 드라이버 이름: com.mysql.jdbc.Driver(com.mysql.cj.jdbc.Driver)
  - ✓ 데이터베이스 이름: jdbc:mysql://ip:포트번호/데이터베이스이름
  - ✓ 데이터에 한글이 있는 경우 데이터베이스 이름 뒤에 ?useUnicode=true&characterEncoding=utf8을 추가
  - ✓ 타임 존 에러가 발생하면 &serverTimezone=UTC 도 추가
  - ✓ ip는 로컬 컴퓨터이면 localhost
  - ✓ 포트번호는 기본적으로 3306
  - ✓ MySql의 JDBC 드라이버를 다운로드 받아서 자바가 설치된 경로의 자바 설치폴더/ jre버전/lib/ext에 복사하거나 프로젝트의 BuildPath에 추가

Mongo DB

- Mongo DB 접속
  - ✓ 드라이버 파일들을 프로젝트의 build path에 추가(mongodb-driver, mongodb-driver-core, bson.jar)

Mongo DB

- Mongo DB 접속
  - ✓ Mongo DB 연결 확인

```
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoDatabase;
public class MongoMain {
    public static void main(String[] args) {
            //Mongo DB 연결
            MongoClient mongoClient =
    MongoClients.create("mongodb://localhost:27017");
            //test 라는 데이터베이스 연결
            MongoDatabase database = mongoClient.getDatabase("adam");
            //소유한 모든 컬렉션 확인
            for (String name : database.listCollectionNames()) {
                    System.out.println(name);
            }
    }
}
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;

import org.bson.Document;

import com.mongodb.BasicDBObject;
import com.mongodb.client.FindIterable;
import com.mongodb.client.MongoClient;
import com.mongodb.client.MongoClients;
import com.mongodb.client.MongoCollection;
import com.mongodb.client.MongoCursor;
import com.mongodb.client.MongoDatabase;
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
public class GoodDAO{
    // 싱글톤 클래스를 만드는 코드
    private GoodDAO() {}

    private static GoodDAO obj;

    public static GoodDAO getInstance() {
            if (obj == null)
                    obj = new GoodDAO();
            return obj;
    }
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
// 데이터베이스 연동 메소드에서 사용할 변수 선언
        MongoClient mongoClient;
        MongoDatabase database;

        // 데이터베이스 연결 메소드
        private void connect() {
                mongoClient =
MongoClients.create("mongodb://localhost:27017");
                database = mongoClient.getDatabase("test");
        }

        // 데이터베이스 연결 해제 메소드
        private void close() {
                mongoClient.close();
        }
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
// Goods 테이블의 모든 데이터를 읽어서 리턴하는 메소드
public List<Document> allGoods() {
        // 데이터를 저장해서 리턴할 인스턴스 생성
        List<Document> list = new ArrayList<Document>();
        connect();
        try {
                MongoCollection<Document> collection =
database.getCollection("goods");
                try (MongoCursor<Document> cur =
collection.find().iterator())                                {
                        while (cur.hasNext()) {
                                Document doc = cur.next();
                                list.add(doc);
                        }
                }
        }catch (Exception e) {
                e.printStackTrace();
        } finally {
                close();
        }
        return list;
}
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
public Document getGood(String code) {
        Document document = null;
        connect();
        try {
                MongoCollection<Document> collection =
database.getCollection("goods");
                Map<String, Object> map = new HashMap<String,
Object>();
                map.put("code", code);
                BasicDBObject object = new BasicDBObject(map);
                FindIterable<Document> fit = collection.find(object);
                ArrayList<Document> docs = new
ArrayList<Document>();
                fit.into(docs);
                if(docs.size() > 0) {
                        document = docs.get(0);
                }
        }
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
                    catch (Exception e) {
                            e.printStackTrace();
                    } finally {
                            close();
                    }
                    return document;
            }
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
// 하나의 데이터를 삽입하는 메소드
public Boolean insertGood(Document document) {
        Boolean result = false;
        connect();
        try {
                MongoCollection<Document> collection =
database.getCollection("goods");
                collection.insertOne(document);
                result = true;
        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                close();
        }
        return result;
}
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
public Boolean updateGood(Document document) {
        Boolean result = false;
        connect();
        try {
                MongoCollection<Document> collection =
database.getCollection("goods");
                collection.updateOne(new Document("code",
document.get("code")),
                                new Document("$set",
document));
                result = true;
        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                close();
        }
        return result;
}
```

Mongo DB

- CRUD 작업
  - ✓ GoodDAO 클래스

```
//데이터를 삭제하는 메소드
public Boolean deleteGood(String code) {
        Boolean result = false;
        connect();
        try {
                MongoCollection<Document> collection =
database.getCollection("goods");
                Document document = new Document();
                document.append("code",code);
                collection.deleteOne(document);
                result = true;
        } catch (Exception e) {
                e.printStackTrace();
        } finally {
                close();
        }
        return result;
}
}
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
import java.util.List;
import java.util.Scanner;
import org.bson.Document;
public class GoodMain {
    public static void main(String[] args) {
            // 키보드 입력 객체 만들기
            Scanner sc = new Scanner(System.in);

            // 각 열의 값을 저장할 변수
            String code = null;
            String name = null;
            String manufacture = null;
            String imsi = null;
            int price = 0;

            // 데이터베이스 작업 결과를 저장할 변수
            Boolean result = null;
            Document document = null;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
MainLoop: while (true) {
                // 메뉴 출력과 입력 받기
                System.out.print("1.데이터 전체 가져오기 2.데이터 1개 가져오기
3.데이터 추가 4.데이터 수정 5.데이터 삭제 6.프로그램 종료:");
                String menu = sc.nextLine();
                // 데이터베이스 연동 인스턴스 만들기
                GoodDAO goodDao = GoodDAO.getInstance();
                switch (menu) {
                case "1":
                        List<Document> list = goodDao.allGoods();
                        for(Document doc : list) {
                                System.out.println(doc.get("code") + ":"
+ doc.get("name")+":" + doc.get("price"));
                        }
                        break;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                    case "2":
                        System.out.print("조회할 데이터의 코드를
입력하세요:");
                        code = sc.nextLine();
                        document = goodDao.getGood(code);
                        if(document == null) {
                            System.out.println("조회할려고 하는
코드가 없습니다.");
                        }else {
                            System.out.println(document);
                        }
                        break;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                    case "3":
                                System.out.print("삽입할 데이터의 코드를
        입력하세요:");
                                code = sc.nextLine();
                                document = goodDao.getGood(code);
                                if (document == null) {
                                        System.out.println("사용 가능한
        코드입니다.");
                                        System.out.print("이름을 입력하세요:");
                                        name = sc.nextLine();
                                        System.out.print("원산지를 입력하세요:");
                                        manufacture = sc.nextLine();
                                        System.out.print("가격을 입력하세요:");
                                        imsi = sc.nextLine();
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        document = new Document();
                                                        document.append("code",
        code);
                                                        document.append("name",
        name);

                document.append("manufacture", manufacture);
                                                        document.append("price",
        price);
                                                        result =
        goodDao.insertGood(document);
                                                        if (result == true) {

                System.out.println("데이터 삽입에 성공하셨습니다.");
                                                        } else {

                System.out.println("데이터 삽입에 실패하셨습니다.");
                        }
                                                }
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                                    catch (Exception e) {
                                                            System.out.println("잘못된
        가격을 입력하셨습니다.");
                                                    }
                                            } else {
                                                    System.out.println("사용 불가능한
        코드입니다.");
                                            }
                                            break;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                    case "4":
                                System.out.print("수정할 데이터의 코드를
        입력하세요:");
                                code = sc.nextLine();
                                document = goodDao.getGood(code);
                                if (document != null) {
                                        System.out.println("수정 가능한
        코드입니다.");
                                        System.out.print("이름을 입력하세요:");
                                        name = sc.nextLine();
                                        System.out.print("원산지를 입력하세요:");
                                        manufacture = sc.nextLine();
                                        System.out.print("가격을 입력하세요:");
                                        imsi = sc.nextLine();
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        document.append("code",
code);
                                                        document.append("name",
name);

        document.append("manufacture", manufacture);
                                                        document.append("price",
price);
                                                        result =
goodDao.updateGood(document);
                                                        if (result == true) {

        System.out.println("데이터 수정에 성공하셨습니다.");
                                                        } else {

        System.out.println("데이터 수정에 실패하셨습니다.");
                                                        }
                                                }
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                                    catch (Exception e) {
                                                            System.out.println("잘못된
가격을 입력하셨습니다.");
                                                    }
                                            } else {
                                                    System.out.println("존재하지 않는
코드라서 수정할 수 없습니다.");
                                            }
                                            break;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                case "5":
                                        System.out.print("삭제할 데이터의 코드를
        입력하세요:");
                                        code = sc.nextLine();
                                        document = goodDao.getGood(code);
                                        if (document != null) {
                                                System.out.println("삭제 가능한
        코드입니다.");
                                                System.out.print("정말로
        삭제하시겠습니까?(삭제:Y)");
                                                String r = sc.nextLine();
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                                                    if (r.toUpperCase().equals("Y")) {
                                                            result =
goodDao.deleteGood(code);
                                                            if (result == true) {

        System.out.println("삭제에 성공하셨습니다..");
                                                            } else {

        System.out.println("삭제에 실패하셨습니다..");
                                                            }
                                                    }
                                            }
                                            else {
                                                    System.out.println("존재하지 않는
코드라서 삭제할 수 없습니다.");
                                            }
                                            break;
```

Mongo DB

- CRUD 작업
  - ✓ GoodMain 클래스

```
                        case "6":
                                break MainLoop;
                        default:
                                System.out.println("메뉴를 잘못 선택하셨습니다.");
                                break;
                        }
                        System.out.println("엔터를 누르면 메인 메뉴로 이동합니다.");
                        sc.nextLine();
                }
                System.out.println("프로그램을 종료합니다!!!!");
                sc.close();
        }
}
```

Oracle

- Oracle
  - ✓ 드라이버 이름: oracle.jdbc.driver.OracleDriver
  - ✓ 데이터베이스 이름: jdbc:oracle:thin:@ip:포트번호:데이터베이스이름
    - sid 대신에 ServiceName 으로 접속하는 경우에는 :데이터베이스이름 대신에 /서비스이름 으로 작성
    - ip는 로컬 컴퓨터이면 localhost
    - 포트번호는 기본적으로 1521
    - 데이터베이스 이름은 기본적으로 orcl(Standard Edition 이나 Enterprise Edition) 이나 xe(Express Edition)
  - ✓ Oracle 홈페이지에서 JDBC 드라이버를 다운로드 받거나 설치된 폴더에서 찾아서 ojdbc6.jar 파일을 자바가 설치된 경로의 자바 설치폴더/ jre버전/lib/ext에 복사하거나 프로젝트의 build path에 드라이버 파일을 추가
    - 오라클이 설치된 경우 드라이버는 C:₩app₩데이터베이스이름 ₩product₩11.2.0₩dbhome_1₩jdbc₩lib
    - 오라클 홈페이지: http://www.oracle.com/technetwork/database/enterprise-edition/jdbc-112010-090769.html

Oracle

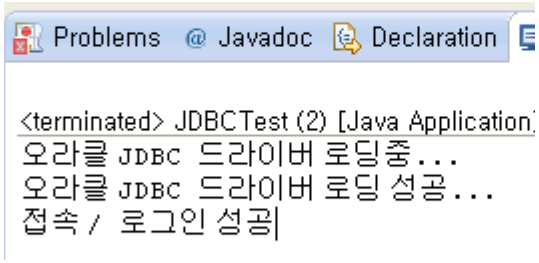
❖ Oracle 접속

```
import java.sql.Connection;
import java.sql.DriverManager;
public class OracleConnect {
    static final String sid = "xe";
    static final String id = "system";
    static final String pw = "oracle";

    public static void main(String[] args) {

      System.out.println("오라클 JDBC 드라이버 로딩중...");
      //오라클 데이타베이스 서버 접속
      Connection connection = null;
      try {
                  // 접속할 데이타베이스의 URL을 만든다.
                  String url = "jdbc:oracle:thin:@localhost:1521:" + sid;

                  // 접속한다(Login)
                  connection = DriverManager.getConnection(url, id, pw);
                  System.out.println("접속 / 로그인 성공");
      }
```

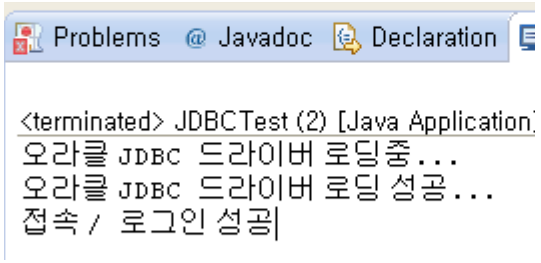


Oracle

❖ Oracle 접속

```
            catch (Exception e) {
                        System.out.println("접속 / 로그인 실패");
                        System.out.println(e.getMessage());
            }

            //연결을 종료한다.
            try {
                        connection.close();
            } catch (Exception e) {
                        System.out.println(e.getMessage());
            }
        }
    }
```



Oracle

- Oracle에 접속해서 작업
  - ✓ 시퀀스 생성

```
CREATE SEQUENCE id_sequence
INCREMENT BY 1
START WITH 1
NOCYCLE;
```

  - ✓ 테이블 생성

```
CREATE TABLE SAMPLE(
        NUM NUMBER PRIMARY KEY,
        NAME VARCHAR2(20));
```

  - ✓ 데이터 추가

```
INSERT INTO SAMPLE VALUES(id_sequence.nextval, 'IRIN');
INSERT INTO SAMPLE VALUES(id_sequence.nextval, 'YEZI');
INSERT INTO SAMPLE VALUES(id_sequence.nextval, 'SUZI');
INSERT INTO SAMPLE VALUES(id_sequence.nextval, 'JENNIE');
```

  - ✓ 데이터 확인: SELECT * FROM SAMPLE;
  - ✓ 작업 완료: COMMIT;

Oracle

❖ Oracle CRUD – select

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.ResultSet;
import java.sql.Statement;

public class SelectMain {

    static final String sid = "xe";
    static final String id = "system";
    static final String pass = "oracle";
```

```
1:IRIN
2:YEZI
3:SUZI
4:JENNIE
```

Oracle

❖ Oracle CRUD – select

```
public static void main(String[] args) {

    Connection connection = null;
    Statement stmt = null;
    ResultSet rs = null;

    String query = "SELECT * FROM SAMPLE";
    try {
            // 접속할 데이타베이스의 URL을 생성
            String url = "jdbc:oracle:thin:@localhost:1521:" + sid;
            // 접속
            connection = DriverManager.getConnection(url, id, pass);
            // sql을 실행 시킬 수 있는 Statement 인스턴스 생성
            stmt = connection.createStatement();
            // select 구문을 실행시키고 그 결과를 rs에 저장
            rs = stmt.executeQuery(query);
```

Oracle

❖ Oracle CRUD – select

```
                    // 데이터가 있다면
                    if (rs.next()) {
                            do {
                                    // 첫번째 컬럼의 값은 정수로 가져오고 두번째 컬
    럼의 값은 문자열로 가져와서 출력
                                    System.out.println(rs.getInt(1) + ":" +
    rs.getString(2));
                            } while (rs.next());
                    }

                    // 읽어온 데이터가 없다면
                    else {
                            System.out.println("읽은 데이터가 없습니다.");
                    }
                    rs.close();
                    stmt.close();
                    connection.close();
      } catch (Exception e) {
                    System.out.println("에러:" + e.getMessage());
      }
    }
}
```

Oracle

- Oracle CRUD – insert

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

public class InsertMain {

    static final String sid = "xe";
    static final String id = "system";
    static final String pass = "oracle";
```

Oracle

- Oracle CRUD – insert

```
public static void main(String[] args) {
   Connection connection = null;
   Statement stmt = null;

   try {
                // 접속할 데이타베이스의 URL을 생성
                String url = "jdbc:oracle:thin:@localhost:1521:" + sid;
                // 접속
                connection = DriverManager.getConnection(url, id, pass);
                // sql을 실행 시킬 수 있는 Statement 인스턴스 생성
                stmt = connection.createStatement();

                String query =
                                "INSERT INTO SAMPLE VALUES
(id_sequence.nextval, 'TZUYU')";
                int result = stmt.executeUpdate(query);
                if(result != 0)
                                System.out.println("삽입성공");
                else
                                System.out.println("삽입실패");
                stmt.close();
                connection.close();
   }
```

Oracle

- Oracle CRUD – insert

```
            catch (Exception e) {
                        System.out.println("에러:" + e.getMessage());
            }
        }
    }
```

Oracle

- Oracle CRUD - PreparedStatement

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

public class PreparedStatementMain {

    static final String sid = "xe";
    static final String id = "system";
    static final String pass = "oracle";
```

Oracle

❖ Oracle CRUD – PreparedStatement

```
public static void main(String[] args) {

  Connection connection = null;
  PreparedStatement stmt = null;
  try {
            // 접속할 데이타베이스의 URL을 생성
            String url = "jdbc:oracle:thin:@localhost:1521:" + sid;
            // 접속
            connection = DriverManager.getConnection(url, id, pass);

            // sql을 실행 시킬 수 있는 Statement 인스턴스 생성
            stmt = connection.prepareStatement("INSERT INTO SAMPLE
VALUES(id_sequence.nextval,?)");
            stmt.setString(1, "TIFFANY");
            int result = stmt.executeUpdate();
            if (result != 0)
                        System.out.println("삽입성공");
            else
                        System.out.println("삽입실패");
            stmt.close();
            connection.close();

  }
```

Oracle

❖ Oracle CRUD – PreparedStatement

```
            catch (Exception e) {
                        System.out.println("에러:" + e.getMessage());
            }
        }
    }
```

Oracle

- BLOB 저장 과 읽기
  - ✓ Oracle에서 실습 테이블 생성

```
CREATE TABLE FILESAMPLE(
        NUM NUMBER(10) PRIMARY KEY,
        FILENAME VARCHAR2(50) NOT NULL,
        FILECONTENT BLOB NOT NULL,
        UPLOADDATE DATE);
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 저장

```
import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.Timestamp;
import java.util.Calendar;

import javax.swing.JFileChooser;
import javax.swing.filechooser.FileNameExtensionFilter;

public class FileUploadMain {
    public static void main(String[] args) {
            Connection con = null;
            PreparedStatement pstmt = null;
            int rownum = -1;
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 저장

```
try {
        con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"oracle");
        // 파일을 선택하기 위한 대화상자
        JFileChooser fc = new JFileChooser();
        FileNameExtensionFilter filter = new FileNameExtensionFilter("
이미지", "jpg","jpeg","gif","png");
        fc.setFileFilter(filter);
        // 대화상자를 화면에 출력하고 누른 버튼의
        // 값을 result에 저장
        int result = fc.showOpenDialog(null);
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 저장

```
// 누른 버튼이 열기버튼이면
if (result == JFileChooser.APPROVE_OPTION) {
        // 선택한 파일을 f에 저장
        File f = fc.getSelectedFile();
        String filename = f.getName();
        FileInputStream fis = new FileInputStream(f);
        // 현재 날짜 및 시간을 Date 인스턴스로 만들기
        Calendar cal = Calendar.getInstance();
        Timestamp today = new
Timestamp(cal.getTimeInMillis());
        pstmt = con.prepareStatement("INSERT INTO
FILESAMPLE VALUES(" + "id_sequence.nextval, ?,?,?)");
        pstmt.setString(1, filename);
        // blob 바인딩
        pstmt.setBinaryStream(2, fis, (int) f.length());
        // 날짜 및 시간 바인딩
        pstmt.setTimestamp(3, today);
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 저장

```
                                        rownum = pstmt.executeUpdate();
                                        if (rownum > 0)
                                                System.out.println("삽입 성공");
                                        pstmt.close();
                                        con.close();
                                }
                        } catch (Exception e) {
                                System.out.println(e.getMessage());
                                e.printStackTrace();
                        }
                }
        }
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 읽기

```
import java.io.FileOutputStream;
import java.io.InputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.sql.Timestamp;

public class FileReadMain {
    public static void main(String[] args) {
            Connection con = null;
            PreparedStatement pstmt = null;
            String sql = null;
            ResultSet rs = null;
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 읽기

```
try {
        con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"oracle");
        // test2 테이블의 모든 데이터를 읽어오는 sql
        sql = "SELECT * FROM FILESAMPLE";
        pstmt = con.prepareStatement(sql);
        rs = pstmt.executeQuery();
        if (rs.next()) {
                do {
                        // 숫자, 문자열, 날짜 읽어오기
                        int num = rs.getInt("NUM");
                        String filename =
rs.getString("FILENAME");
                        Timestamp uploadDate =
rs.getTimestamp("UPLOADDATE");
                        System.out.println(num + ":" + filename
+ ":" + uploadDate);
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 읽기

```
                                                        // BLOB 가져오기
                                                        InputStream is =
            rs.getBinaryStream("filecontent");

                                                        // 가져온 데이터를 filename에 기록하기
                                                        FileOutputStream fos = new
            FileOutputStream("./" + filename);

                                                        while (true) {
                                                                byte[] b = new byte[1024];
                                                                int len = is.read(b);
                                                                if (len <= 0)
                                                                        break;
                                                                fos.write(b, 0, len);
                                                        }
                                                        fos.close();

                                                } while (rs.next());
                                        } else {
                                                System.out.println("데이터가 없습니다.");
                                        }
                                }
```

Oracle

- BLOB 저장 과 읽기
  - ✓ 읽기

```
                catch (Exception e) {
                        System.out.println(e.getMessage());
                        e.printStackTrace();
                }
            }
        }
```

Oracle

- Procedure
  - ✓ 데이터베이스에서 자주 사용하는 sql을 미리 작성해두고 호출하는 개체
  - ✓ 프로시저를 이용하면 코딩이나 성능 및 효율 면에서 이득을 취할 수 도 있음
  - ✓ Oracle에서 작성

```
DROP SEQUENCE id_sequence;

CREATE SEQUENCE id_sequence INCREMENT BY 1 START WITH 1 NOCYCLE;

CREATE TABLE PROCSAMPLE(
NUM NUMBER(10) PRIMARY KEY,
MESSAGE VARCHAR2(200),
WRITEDATE DATE);

CREATE OR REPLACE PROCEDURE MyProc
(P_MESSAGE IN PROCSAMPLE.MESSAGE%TYPE)
  IS
  BEGIN
        INSERT INTO PROCSAMPLE
        VALUES(id_sequence.nextval,P_MESSAGE, SYSDATE);
  END;
/
```

Oracle

- Procedure
  - ✓ Procedure 호출

```
import java.sql.CallableStatement;
import java.sql.Connection;
import java.sql.DriverManager;
import java.util.Scanner;

public class ProcedureMain {
    public static void main(String[] args) {
            Scanner in = new Scanner(System.in);
            System.out.print("메시지를 입력:");
            String message = "";
            try {
                    message = in.nextLine();
            } catch (Exception e) {
                    System.out.println(e.getMessage());
            }
            Connection conn = null;
            String url = "jdbc:oracle:thin:@localhost:1521:xe";
            String id = "system";
            String pass = "oracle";
            String query1 = "{call MyProc(?)}";
```

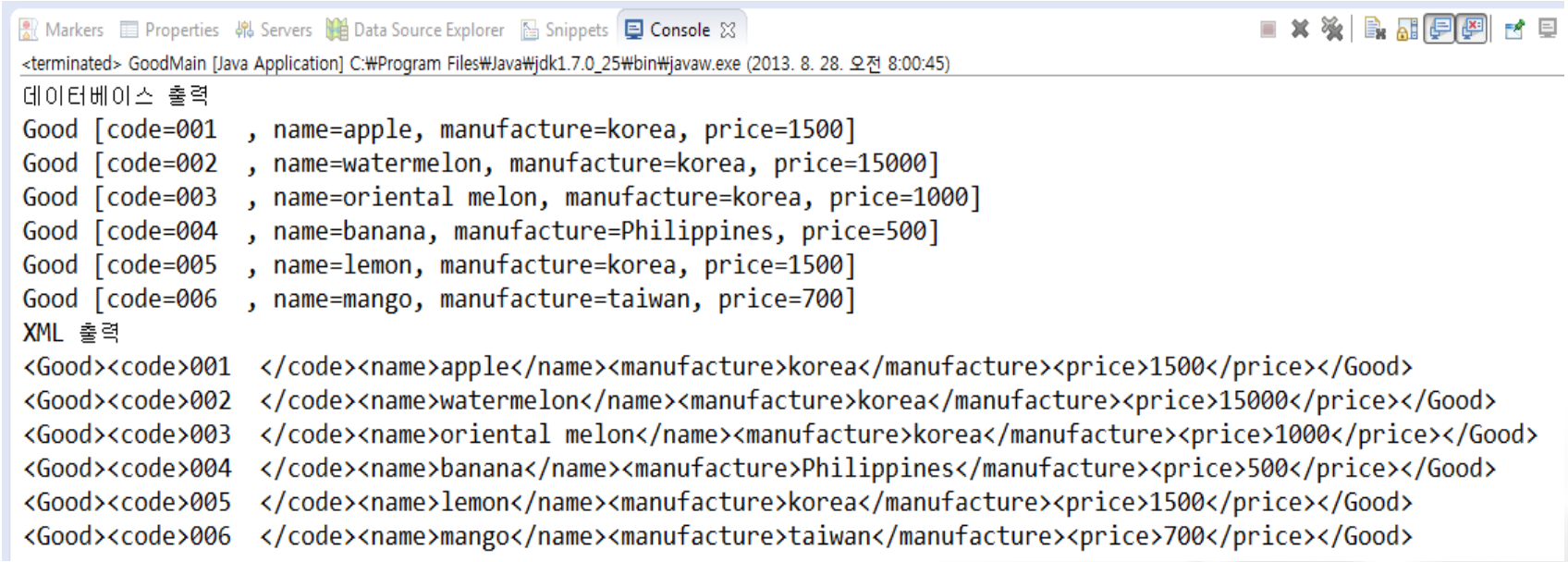
Oracle

- Procedure
  - ✓ Procedure 호출

```
                try {
                        conn = DriverManager.getConnection(url, id, pass);
                        CallableStatement call = conn.prepareCall(query1);
                        call.setString(1, message);
                        call.executeQuery();
                        call.close();
                        conn.close();
                        in.close();
                        System.out.println("프로시저 실행 성공");
                } catch (Exception e) {
                        System.out.println("실패:" + e.getMessage());
                }
        }
}
```

Oracle

❖ DTO & DAO Pattern



Oracle

- DTO & DAO Pattern
  - ✓ 데이터베이스 작업
    - 데이터베이스 테이블 작성

```
create table Goods (
    code      char(5) not null,
    name      varchar2(50) not null,
    manufacture    varchar2(20),
    price     number(10) not null,
    primary key(code)
);
```

    - 샘플 데이터 입력

```
insert into Goods values('001', 'apple', 'korea', 1500);
insert into Goods values('002', 'watermelon', 'korea', 15000);
insert into Goods values('003', 'oriental melon', 'korea', 1000);
insert into Goods values('004', 'banana', 'philippines', 500);
insert into Goods values('005', 'lemon', 'korea', 1500);
insert into Goods values('006', 'mango', 'taiwan', 700);

commit;

select * from Goods;
```

Oracle

- DTO & DAO Pattern
  - ✓ DTO 클래스 생성

```
public class Good{
    private String code;
    private String name;
    private String manufacture ;
    private int price;

    public Good() {
            super();
    }
    public Good(String code, String name, String manufacture, int price) {
            super();
            this.code = code;
            this.name = name;
            this.manufacture = manufacture;
            this.price = price;
    }
```

Oracle

- DTO & DAO Pattern
  - ✓ DTO 클래스 생성

```
public String getCode() {
        return code;
}
public void setCode(String code) {
        this.code = code;
}
public String getName() {
        return name;
}
public void setName(String name) {
        this.name = name;
}
public String getManufacture() {
        return manufacture;
}
public void setManufacture(String manufacture) {
        this.manufacture = manufacture;
}
```

Oracle

- DTO & DAO Pattern
  - ✓ DTO 클래스 생성

```
        public int getPrice() {
                return price;
        }
        public void setPrice(int price) {
                this.price = price;
        }
        @Override
        public String toString() {
                return "Good [code=" + code + ", name=" + name + ", manufacture="
                                + manufacture + ", price=" + price + "]";
        }
}
```

Oracle

- DTO & DAO Pattern
  - ✓ DAO 인터페이스 생성

```
import java.util.List;

public interface GoodDAO {
    //테이블의 모든 데이터를 가져오는 메소드
    public List<Good> allSelect();

    //기본키를 가지고 하나의 데이터를 가져오는 메소드
    public Good getGood(String code);

    //하나의 데이터를 삽입하는 메소드
    public Boolean insertGood(Good good);

    //하나의 데이터를 수정하는 메소드
    public Boolean updateGood(Good good);

    //하나의 데이터를 삭제하는 메소드
    public Boolean deleteGood(String code);
}
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
import java.io.File;
import java.io.FileInputStream;
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;
import java.util.Properties;

public class GoodDAOImpl implements GoodDAO {
    // 싱글톤 클래스를 만드는 코드
    private GoodDAOImpl() {}

    private static GoodDAOImpl obj;

    public static GoodDAOImpl getInstance() {
            if (obj == null)
                    obj = new GoodDAOImpl();
            return obj;
    }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
// 데이터베이스 연동 메서드에서 사용할 변수 선언
private Connection con;
private PreparedStatement pstmt;

// 데이터베이스 연결 메서드
private boolean connect() {
        boolean result = false;
        try {
                con =
DriverManager.getConnection("jdbc:oracle:thin:@localhost:1521:xe", "system",
"oracle");
                result = true;
        } catch (Exception e) {
                e.printStackTrace();
        }
        return result;
}
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
//con과 pstmt 연결 해제하는 메서드
private void close() {
        try {
                if (pstmt != null)
                        pstmt.close();
                if (con != null)
                        con.close();
        } catch (Exception e) {
                e.printStackTrace();
        }
}
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
// Goods 테이블의 모든 데이터를 읽어서 리턴하는 메서드
public List<Good> allSelect() {
        // 데이터를 저장해서 리턴할 인스턴스 생성
        List<Good> list = new ArrayList<Good>();
        // select 구문의 결과를 저장할 변수 생성
        ResultSet rs = null;
        try {
                // 연결에 성공하면
                if (connect()) {
                        // SQL을 실행할 수 있는 인스턴스 생성
                        pstmt = con.prepareStatement("select * from
goods");
                        // select 구문의 결과를 rs에 저장
                        rs = pstmt.executeQuery();
                        // 검색된 데이터를 읽어서 list에 저장
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
                                        if (rs.next()) {
                                                do {
                                                        Good good = new Good();

                good.setCode(rs.getString("code"));

                good.setName(rs.getString("name"));

                good.setManufacture(rs.getString("manufacture"));

                good.setPrice(rs.getInt("price"));
                                                        list.add(good);
                                                } while (rs.next());
                                        }
                        }
                } catch (Exception e) {
                        e.printStackTrace();
                }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
            finally {
                    try {
                            if (rs != null) rs.close();
                            close();
                    } catch (Exception e) {}
            }
            return list;
    }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
// code(Primary Key)를 받아서 goods 테이블에서 데이터를 검색한 후 리턴하는 메서드
// 데이터는 없거나 1개만 나올 수 있습니다.
// 없을 때는 null을 리턴하고 있을 때는 1개를 리턴
// 목록에서 제목을 눌렀을 때 상세보기를 하거나 회원정보 수정을 눌렀을 때 회원정보 보여주는 화면을 만들 때 사용
public Good getGood(String code) {
        // 검색된 정보를 저장해서 리턴하기 위한 변수
        Good good = null;
        // 검색된 데이터 정보를 저장할 변수
        ResultSet rs = null;
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
try{
        //데이터베이스 연결
        if(connect()){
                //SQL 실행 인스턴스 생성
                pstmt = con.prepareStatement(
                        "select * from goods where
trim(code) = ?");

                //물음표에 데이터 바인딩
                pstmt.setString(1, code);
                //select 구문을 실행해서 결과를 rs에 저장
                rs = pstmt.executeQuery();
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
                                        //검색된 데이터는 1개가 넘지 않습니다.
                                        if(rs.next()){
                                                good = new Good();

        good.setCode(rs.getString("code").trim());
                                                good.setName(rs.getString("name"));

        good.setManufacture(rs.getString("manufacture"));
                                                good.setPrice(rs.getInt("price"));
                                        }
                        }
        }
        catch(Exception e){
                        e.printStackTrace();
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
                finally{
                        try{
                                if(rs != null)
                                        rs.close();
                                close();
                        }catch(Exception e){
                                e.printStackTrace();
                        }
                }
                return good;
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
//하나의 데이터를 삽입하는 메서드
public Boolean insertGood(Good good){
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(
                                        "insert into goods(" +
                                                        "code,
name, manufacture,price)"+
                                        " values(?,?,?,?)");
                        pstmt.setString(1, good.getCode());
                        pstmt.setString(2, good.getName());
                        pstmt.setString(3, good.getManufacture());
                        pstmt.setInt(4, good.getPrice());
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
                catch(Exception e){
                        e.printStackTrace();
                }finally{
                        close();
                }
                return result;
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
public Boolean updateGood(Good good) {
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(   "update goods " +
                                        "set name = ?, manufacture
= ?, price = ? "+
                                        "where trim(code) = ?");

                        pstmt.setString(1, good.getName());
                        pstmt.setString(2, good.getManufacture());
                        pstmt.setInt(3, good.getPrice());
                        pstmt.setString(4, good.getCode());
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
                catch(Exception e){
                        e.printStackTrace();
                }finally{
                        close();
                }
                return result;
        }
```

Oracle

- DTO & DAO Pattern
  - ✓ DAOImpl 클래스 생성

```
@Override
public Boolean deleteGood(String code) {
        Boolean result = false;
        try{
                if(connect()){
                        pstmt = con.prepareStatement(
                                        "delete from goods " +
                                        "where trim(code) = ?");
                        pstmt.setString(1, code);
                        int rownum = pstmt.executeUpdate();
                        if(rownum > 0)
                                result = true;
                }
        }catch(Exception e){
                e.printStackTrace();
        }finally{
                close();
        }
        return result;
    }
}
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
import java.util.List;
import java.util.Scanner;
import javax.swing.JOptionPane;

public class GoodMain {

    public static void main(String[] args) {
        //키보드 입력 객체 만들기
        Scanner sc = new Scanner(System.in);

        //각 열의 값을 저장할 변수
        String code = null;
        String name = null;
        String manufacture = null;
        String imsi = null;
        int price = 0;

        //데이터베이스 작업 결과를 저장할 변수
        Boolean result = null;
        Good  good = null;
        List<Good> list = null;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
//무한 루프
MainLoop : while(true) {
        //메뉴 출력과 입력 받기
        System.out.print("1.데이터 전체 가져오기 2.데이터 1개 가져오기
3.데이터 추가 4.데이터 수정 5.데이터 삭제 6.프로그램 종료:");
        String menu = sc.nextLine();
        //데이터베이스 연동 인스턴스 만들기
        GoodDAO goodDao = GoodDAOImpl.getInstance();
        switch(menu) {
        case "1":
                list = goodDao.allSelect();
                for(Good temp : list) {
                        System.out.println(temp);
                }
                break;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "2":
                        System.out.print("조회할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good == null) {
                            System.out.println("존재하지 않는 코드입
니다.");
                        }else {
                            System.out.println(good);
                        }
                        break;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "3":
                        System.out.print("삽입할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good == null) {
                            System.out.println("사용 가능한 코드입니
다.");
                            System.out.print("이름을 입력하세요:");
                            name = sc.nextLine();
                            System.out.print("원산지를 입력하세요:");
                            manufacture = sc.nextLine();
                            System.out.print("가격을 입력하세요:");
                            imsi = sc.nextLine();
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        good = new Good(code,
name, manufacture, price);
                                                        result =
goodDao.insertGood(good);
                                                        if(result == true) {
                                                                System.out.println("
데이터 삽입에 성공하셨습니다.");
                                                        }else {
                                                                System.out.println("
데이터 삽입에 실패하셨습니다.");
                                                        }
                                                }catch(Exception e) {
                                                        System.out.println("잘못된 가격
을 입력하셨습니다.");
                                                }
                                        }else {
                                                System.out.println("사용 불가능한 코드입
니다.");
                                        }
                                        break;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "4":
                        System.out.print("수정할 데이터의 코드를 입력하세요
:");
                        code = sc.nextLine();
                        good = goodDao.getGood(code);
                        if(good != null) {
                            System.out.println("수정 가능한 코드입니
다.");
                            System.out.print("이름을 입력하세요:");
                            name = sc.nextLine();
                            System.out.print("원산지를 입력하세요:");
                            manufacture = sc.nextLine();
                            System.out.print("가격을 입력하세요:");
                            imsi = sc.nextLine();
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                                try {
                                                        price = Integer.parseInt(imsi);
                                                        good = new Good(code,
name, manufacture, price);
                                                        result =
goodDao.updateGood(good);
                                                        if(result == true) {
                                                                System.out.println("
데이터 수정에 성공하셨습니다.");
                                                        }else {
                                                                System.out.println("
데이터 수정에 실패하셨습니다.");
                                                        }
                                                }catch(Exception e) {
                                                        System.out.println("잘못된 가격
을 입력하셨습니다.");
                                                }
                                        }else {
                                                System.out.println("존재하지 않는 코드라
서 수정할 수 없습니다.");
                                        }
                                        break;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                        case "5":
                                System.out.print("삭제할 데이터의 코드를 입력하세요
:");
                                code = sc.nextLine();
                                good = goodDao.getGood(code);
                                if(good != null) {
                                        System.out.println("삭제 가능한 코드입니
다.");
                                        int r =
JOptionPane.showConfirmDialog(null, "정말로 삭제하시겠습니까?");
                                        if(r == JOptionPane.OK_OPTION) {
                                                result =
goodDao.deleteGood(code);
                                                if(result == true) {
                                                        System.out.println("

삭제에 성공하셨습니다..");
                                                }else {
                                                        System.out.println("

삭제에 실패하셨습니다..");
                                                }
                                        }
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                                        }else {
                                                System.out.println("존재하지 않는 코드라
서 삭제할 수 없습니다.");
                                        }
                                        break;
```

Oracle

- DAO & DTO Pattern
  - ✓ 실행을 위한 GoodMain 클래스를 생성

```
                    case "6":
                        break MainLoop;
                    default :
                        System.out.println("메뉴를 잘못 선택하셨습니다.");
                        break;
                    }
                    System.out.println("엔터를 누르면 메인 메뉴로 이동합니다.");
                    sc.nextLine();
                }
                System.out.println("프로그램을 종료합니다!!!!");
                sc.close();
            }
        }
```