MySQL 연동

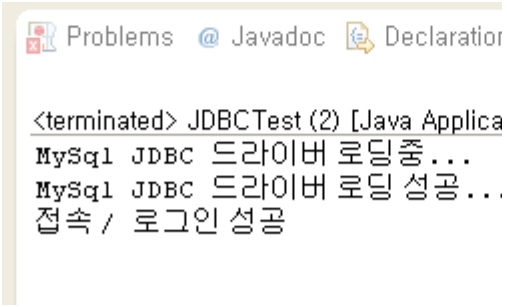
1. JDBC

❖MySql

- ✓ 드라이버 이름: com.mysql.jdbc.Driver
- ✓ 데이터베이스 이름: jdbc:mysql://ip:포트번호/데이터베이스이름
- ✓ 데이터에 한글이 있는 경우 데이터베이스 이름 뒤에 ?useUnicode=true&characterEncoding=utf8을 추가
- ✓ ip는 로컬 컴퓨터이면 localhost
- ✓ 포트번호는 기본적으로 3306
- ✓ MySql의 JDBC 드라이버를 다운로드 받아서 자바가 설치된 경로의 자바 설치폴더/ jre 버전/lib/ext에 복사하거나 프로젝트의 BuildPath에 추가

실습(MySql 접속)

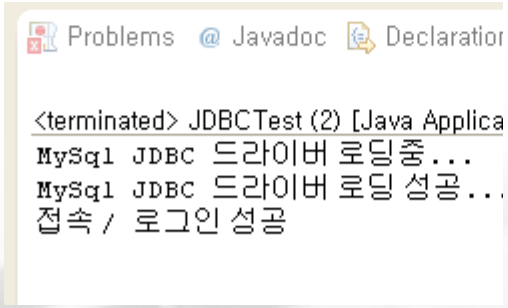
- 프로젝트 생성
- 프로젝트에 MySQL Driver를 복사하고 Build Path에 추가
- 프로젝트에 main 메소드를 소유한 클래스를 생성하고 작성



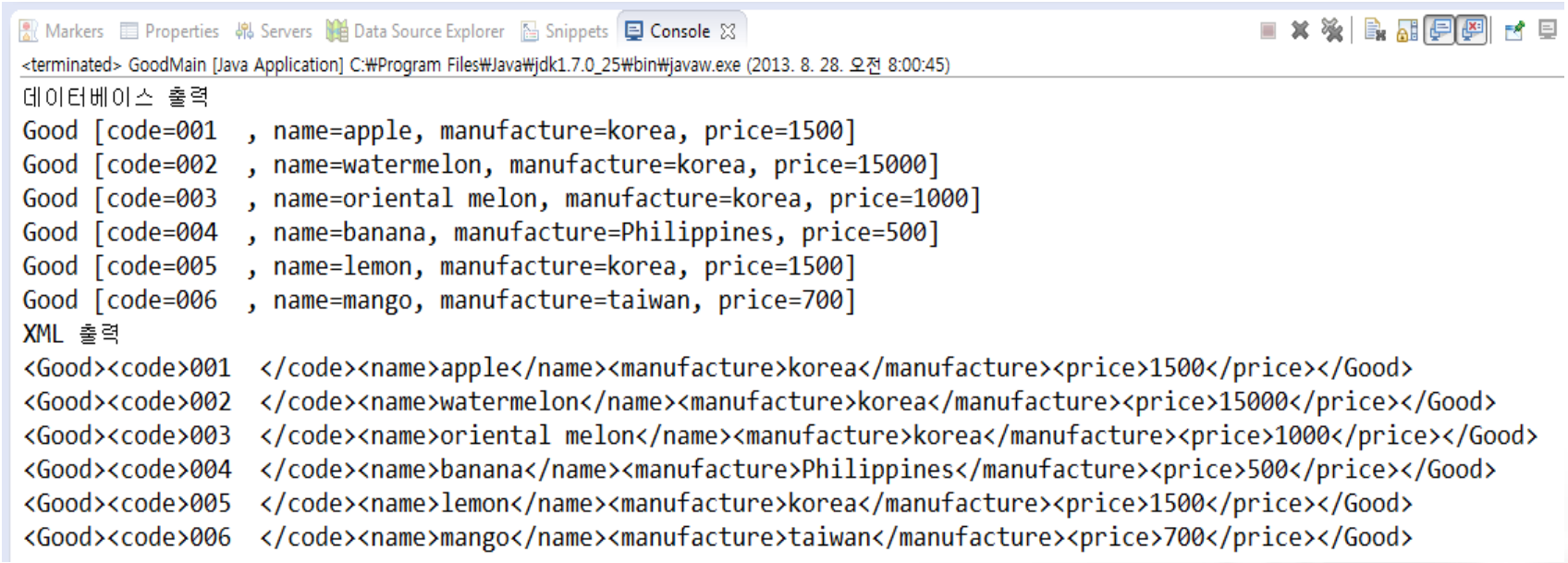

```
import java.sql.Connection;
import java.sql.DriverManager;

public class JDBCMySQL {
        static final String sid = "sample";
        static final String id = "root";
        static final String pass = "wnddkd";

        public static void main(String[] args) {
                // 1. MySql 데이타베이스 드라이버 로딩
                System.out.println("MySql JDBC 드라이버 로딩중...");
                try {
                        Class.forName("com.mysql.jdbc.Driver");
                        System.out.println("MySql JDBC 드라이버 로딩 성공...");

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

실습(MySql 접속)

```
// 2. MySql 데이타베이스 서버 접속
Connection connection = null;
try {
        // 접속할 데이타베이스의 URL을 만든다.
        String url = "jdbc:mysql://localhost:3306/" + sid;
        // 접속한다(Login)
        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());
}

// 3. 연결을 종료한다.
try {
        connection.close();
} catch (Exception e) {
        System.out.println(e.getMessage());
        e.printStackTrace();
}
    }
}
```



DTO & DAO 패턴


Markers
Properties
Servers
Data Source Explorer
Snippets
Console
<terminated> GoodMain [Java Application] C:₩Program Files₩Java₩jdk1.7.0_25₩bin₩javaw.exe (2013. 8. 28. 오전 8:00:45)
데이터베이스 출력
Good [code=001 , name=apple, manufacture=korea, price=1500]
Good [code=002 , name=watermelon, manufacture=korea, price=15000]
Good [code=003 , name=oriental melon, manufacture=korea, price=1000]
Good [code=004 , name=banana, manufacture=Philippines, price=500]
Good [code=005 , name=lemon, manufacture=korea, price=1500]
Good [code=006 , name=mango, manufacture=taiwan, price=700]
XML 출력
<Good><code>001 </code><name>apple</name><manufacture>korea</manufacture><price>1500</price></Good>
<Good><code>002 </code><name>watermelon</name><manufacture>korea</manufacture><price>15000</price></Good>
<Good><code>003 </code><name>oriental melon</name><manufacture>korea</manufacture><price>1000</price></Good>
<Good><code>004 </code><name>banana</name><manufacture>Philippines</manufacture><price>500</price></Good>
<Good><code>005 </code><name>lemon</name><manufacture>korea</manufacture><price>1500</price></Good>
<Good><code>006 </code><name>mango</name><manufacture>taiwan</manufacture><price>700</price></Good>

DTO & DAO 패턴

❖ 데이터베이스 테이블 작성

```
use sample;

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;
```

❖ 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;
```

DTO & DAO 패턴

❖ 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;
	}
```

DTO & DAO 패턴

```
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;
 }
```

DTO & DAO 패턴

```
    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 + "]";
     }
}
```

DTO & DAO 패턴

❖ DAO 클래스의 메소드의 원형을 가진 GoodDAO 인터페이스를 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);
}
```

DTO & DAO 패턴

❖ DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

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;
	}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

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

// 데이터베이스 연결 메소드
private boolean connect() {
	boolean result = false;
	try {
		con =
DriverManager.getConnection("jdbc:mysql://localhost:3306/sample?useUnicode=true&characterE
ncoding=utf8", "root", "wnddkd");
		result = true;
	} catch (Exception e) {
		e.printStackTrace();
	}
	return result;
}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

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

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

```
// Goods 테이블의 모든 데이터를 읽어서 리턴하는 메소드
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();
```

DTO & DAO 패턴

- DAO 메소드를 구현할 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;
}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

```
// code(Primary Key)를 받아서 goods 테이블에서 데이터를 검색한 후 리턴하는 메소드
// 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;

        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();
```

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 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();
        }
        finally{
                try{
                        if(rs != null)
                                rs.close();
                        close();
                }catch(Exception e){
                        e.printStackTrace();
                }
        }
        return good;
}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 GoodDAOImpl 클래스를 생성

```
//하나의 데이터를 삽입하는 메소드
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;
                }
        }catch(Exception e){
                e.printStackTrace();
        }finally{
                close();
        }
        return result;
}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 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;
                }
        }catch(Exception e){
                e.printStackTrace();
        }finally{
                close();
        }
        return result;
}
```

DTO & DAO 패턴

- DAO 메소드를 구현할 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;
    }
}
```

DTO & DAO 패턴

❖ Main 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;
```

DTO & DAO 패턴

- Main 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;
```

DTO & DAO 패턴

- Main 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 코드입니다.");
        }else {
                System.out.println(good);
        }
        break;
```

DTO & DAO 패턴

- Main 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 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();
                            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.");
                                }
                            }
```

DTO & DAO 패턴

- Main 클래스를 생성

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

DTO & DAO 패턴

- Main 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 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();
                                    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.");
                                            }
                                    }
```

DTO & DAO 패턴

- Main 클래스를 생성

```
                                        catch(Exception e) {
                                                System.out.println("잘못된 가격을 입력하셨습니
다.");
                                        }
                                }else {
                                        System.out.println("존재하지 않는 코드라서 수정할 수 없습
니다.");
                                }
                                break;
```

DTO & DAO 패턴

- Main 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 코드입니다.");
                            int r = JOptionPane.showConfirmDialog(null, "정말로 삭제
하시겠습니까?");
                            if(r == JOptionPane.OK_OPTION) {
                                result = goodDao.deleteGood(code);
                                if(result == true) {
                                    System.out.println("삭제에 성공하셨
습니다..");
                                }else {
                                    System.out.println("삭제에 실패하셨
습니다..");
                                }
                            }
                        }else {
                            System.out.println("존재하지 않는 코드라서 삭제할 수 없습
니다.");
                        }
                        break;
```

DTO & DAO 패턴

- Main 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();
    }
}
```

COVID19

❖ 데이터베이스 테이블 작성

```
use sample;

create table covid19(
      num int primary key,
      region varchar(30) not null,
      nation varchar(30) not null,
      pop int not null,
      confirmcount int not null,
      deathcount int not null)engine=innodb default charset=utf8;

insert into covid19(num, region, nation, pop, confirmcount, deathcount)
      values(1, '아메리카', '미국', 331002647, 1486376, 89550);

insert into covid19(num, region, nation, pop, confirmcount, deathcount)
      values(2, '유럽', '러시아', 145934460, 281752, 9709);

commit;

select * from covid19;
```

COVID19

❖ DTO 클래스인 COVID 클래스를 생성

```
public class Covid {
	private int num;
	private String region;
	private String nation;
	private int pop;
	private int confirmcount;
	private int deathcount;

	public int getNum() {
		return num;
	}
	public void setNum(int num) {
		this.num = num;
	}
	public String getRegion() {
		return region;
	}
	public void setRegion(String region) {
		this.region = region;
	}
```

COVID19

- DTO 클래스인 COVID 클래스를 생성

```
public String getNation() {
        return nation;
}
public void setNation(String nation) {
        this.nation = nation;
}
public int getPop() {
        return pop;
}
public void setPop(int pop) {
        this.pop = pop;
}
public int getConfirmcount() {
        return confirmcount;
}
public void setConfirmcount(int confirmcount) {
        this.confirmcount = confirmcount;
}
public int getDeathcount() {
        return deathcount;
}
public void setDeathcount(int deathcount) {
        this.deathcount = deathcount;
}
```

COVID19

- DTO 클래스인 COVID 클래스를 생성

```
    @Override
    public String toString() {
            return "Covid [num=" + num + ", region=" + region + ", nation=" + nation + ", pop=" + pop
    + ", confirmcount="
                                    + confirmcount + ", deathcount=" + deathcount + "]";
    }
}
```

COVID19

❖ DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
import java.sql.Connection;
import java.sql.DriverManager;
import java.sql.PreparedStatement;
import java.sql.ResultSet;
import java.util.ArrayList;
import java.util.List;

public class CovidDAO {

	//클래스가 로드될 때 1번만 수행되는 코드
	//제일 먼저 실행되는 코드
	static {
		//MySQL 드라이버 클래스 로드
		try {
			Class.forName("com.mysql.jdbc.Driver");

		} catch (Exception e) {
			System.err.println("드라이버 클래스 로드 실패");
			System.err.println(e.getMessage());
			e.printStackTrace();
		}
	}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//싱글톤 패턴(인스턴스를 1개만 생성할 수 있도록 하는 패턴)을 위한 코드
//Back-End Programmer를 주력으로 하고자 하는 경우에는
//객체지향 디자인 패턴을 학습

//생성자가 private 이므로 외부에서 인스턴스 생성을 못함
private CovidDAO() {}

//변수를 1개만 생성할 수 있도록 선언
private static CovidDAO covidDAO;

//외부에서 생성된 인스턴스를 사용할 수 있도록 리턴해주는 메소드
public static CovidDAO sharedInstance() {
        //static 변수이므로 null을 대입하지 않는 이상
        //맨 처음에만 null이고 이후에는 null이 될 수 없음
        if(covidDAO == null) {
                covidDAO = new CovidDAO();
        }
        return covidDAO;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//여러 메소드에서 공통으로 사용할 변수
//java.sql 패키지의 클래스를 import
private Connection con;
private PreparedStatement pstmt;

//데이터베이스 접속을 위한 메소드
private void connect() {
	try {
		//데이터베이스 연결
		con = DriverManager.getConnection(
			"jdbc:mysql://192.168.0.200:3306/sample"
			+ "?useUnicode=true&characterEncoding=utf8",
			"root","*******");
		//System.out.println("데이터베이스 접속 성공");
	}catch(Exception e) {
		System.err.println("데이터베이스 연결 실패");
		System.err.println(e.getMessage());
		e.printStackTrace();
	}
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//데이터베이스 연결 객체를 정리해주는 메소드
private void close() {
	try {
		if(pstmt != null) {
			pstmt.close();
		}
		if(con != null) {
			con.close();
		}
	}catch(Exception e) {
		System.err.println("데이터베이스 해제 실패");
		System.err.println(e.getMessage());
		e.printStackTrace();
	}
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//1.전체보기를 위한 메소드
//조회는 몇 열의 데이터를 몇 행으로 조회하는지 : 리턴타입
//where 절이 있는지 확인 : 파라미터
//select * from covid19;
public List<Covid> allCovid() {
        //List를 리턴할 때는 인스턴스를 만들고 리턴
        List<Covid> list = new ArrayList<Covid>();
        //데이터베이스 연결
        connect();
        try {
                //SQL 실행 객체 생성
                pstmt = con.prepareStatement(
                        "select * from covid19");
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
                    //SQL 실행
                    ResultSet rs = pstmt.executeQuery();
                    while(rs.next()) {
                            //List 개별 요소 인스턴스를 생성
                            Covid covid = new Covid();
                            //인스턴스의 내부 요소를 채우기
                            covid.setNum(rs.getInt("num"));
                            covid.setNation(rs.getString("nation"));
                            covid.setConfirmcount(rs.getInt("confirmcount"));
                            //List에 추가
                            list.add(covid);
                    }
                    rs.close();
            }catch(Exception e) {
                    System.err.println("전체 데이터 가져오기 실패");
                    System.err.println(e.getMessage());
                    e.printStackTrace();
            }
            //데이터베이스 연결 해제
            close();
            //데이터가 없을 때는 List의 size가 0
            return list;
    }
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//2.상세보기를 위한 메소드
//select * from 테이블이름 where 기본키=?
public Covid getCovid(int num){
        //하나의 행이 리턴되는 경우는 인스턴스를 데이터를 가져왔을 때 생성
        Covid covid = null;
        connect();

        try {
                //select 구문의 경우 where절이 있으면
                //데이터를 매개변수로 받아서 바인딩을 해야 합니다.
                pstmt = con.prepareStatement(
                        "select * from covid19 where num = ?");
                pstmt.setInt(1, num);

                //SQL 실행
                ResultSet rs = pstmt.executeQuery();
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
                    //리턴되는 데이터가 1개 이하인 경우
                    if(rs.next()) {
                                //리턴할 데이터의 인스턴스를 생성
                                covid = new Covid();
                                covid.setNum(rs.getInt("num"));
                                covid.setRegion(rs.getString("region"));
                                covid.setNation(rs.getString("nation"));
                                covid.setPop(rs.getInt("pop"));
                                covid.setConfirmcount(
                                            rs.getInt("confirmcount"));
                                covid.setDeathcount(
                                            rs.getInt("deathcount"));
                    }

        }catch(Exception e) {
                    System.out.println("상세보기 실패");
                    System.out.println(e.getMessage());
                    e.printStackTrace();
        }

        close();
        return covid;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//covid19테이블에서 가장 큰 num을 찾아오는 메소드
//select max(num) from covid19;
public int maxNum() {
	//데이터가 없을 때는 번호가 0이 있다고 가정
	int result = 0;
	connect();
	try {
		pstmt = con.prepareStatement(
			"select max(num) from covid19");
		ResultSet rs = pstmt.executeQuery();
		if(rs.next()) {
			//select의 첫번째 컬럼의 값을 정수로 result에 저장
			result = rs.getInt(1);
		}

	}catch(Exception e) {
		System.out.println("가장 큰 번호 가져오기 실패");
		System.out.println(e.getMessage());
		e.printStackTrace();
	}
	close();
	return result;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//3.데이터를 삽입하는 메소드
//insert into 테이블이름(컬럼이름 나열)
//values(값을 나열)
//특별한 경우가 아니면 컬럼은 2개 이상
//select를 제외한 모든 SQL의 실행은 영향받은 행의 개수를 리턴
public int insertCovid(Covid covid){
        //여기서 -1은 의미없는 값으로 삽입 실패를 의미하는 값
        //어떤 음수라도 가능 - 0은 조심
        int result = -1;
        //가장 큰 num을 찾아서 +1을 한 후 num에 대입
        int num = this.maxNum() + 1;
        //위의 문장이 connect() 뒤에 있으면 데이터베이스 연결을 해제해서
        //예외가 발생
        connect();
        try {
                //SQL 실행 객체 생성 - SQL 생성
                //값을 대입하는 곳은 ?로 설정
                //값을 대입하는 곳 중에서 고정된 값이면 고정된 값을 이용
                pstmt = con.prepareStatement(
                        "insert into covid19("
                        + "num, region, nation, pop, "
                        + "confirmcount, deathcount) "
                        + "values(?,?,?,?,?,?)");
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
                    //비어 있는 곳에 값을 채움(바인딩 - Binding)
                    //번호를 입력받는 경우
                    //pstmt.setInt(1, covid.getNum());
                    //가장 큰 번호 + 1
                    pstmt.setInt(1, num);
                    pstmt.setString(2, covid.getRegion());
                    pstmt.setString(3, covid.getNation());
                    pstmt.setInt(4, covid.getPop());
                    pstmt.setInt(5, covid.getConfirmcount());
                    pstmt.setInt(6, covid.getDeathcount());
                    //SQL 실행
                    result = pstmt.executeUpdate();
            }catch(Exception e) {
                    System.out.println("데이터 삽입 실패");
                    System.out.println(e.getMessage());
                    //위의 2개 작업은 파일이나 데이터베이스에 기록하고 주석 처리
                    //예외 발생 지점을 찾기 위한 작업
                    e.printStackTrace();
            }
            //데이터베이스 연결 해제
            close();
            return result;
    }
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//데이터를 삭제하는 메소드
//delete from 테이블이름 where 기본키 = ?
//리턴 타입은 정수
//매개변수는 기본키
public int deleteCovid(int num){
	int result = -1;
	connect();
	//삽입, 삭제, 갱신은 sql을 생성 부분과 바인딩 하는 부분만 변경
	try {
		pstmt = con.prepareStatement(
			"delete from covid19 where num = ?");
		//데이터 바인딩
		pstmt.setInt(1, num);
		//실행
		result = pstmt.executeUpdate();
	}catch(Exception e) {
		System.out.println("데이터 삭제 실패");
		System.out.println(e.getMessage());
		e.printStackTrace();
	}
	close();
	return result;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//데이터 수정을 위한 메소드
public int updateCovid(Covid covid) {
        int result = -1;
        connect();
        //삽입, 삭제, 갱신은 sql을 생성 부분과 바인딩 하는 부분만 변경
        try {
                pstmt = con.prepareStatement(
                        "update covid19 "
                        + "set region=?,nation=?,pop=?,"
                        + "confirmcount=?, deathcount=? "
                        + "where num=?");
                pstmt.setString(1, covid.getRegion());
                pstmt.setString(2, covid.getNation());
                pstmt.setInt(3, covid.getPop());
                pstmt.setInt(4, covid.getConfirmcount());
                pstmt.setInt(5, covid.getDeathcount());
                pstmt.setInt(6, covid.getNum());

                result = pstmt.executeUpdate();

        }
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
        catch(Exception e) {
                System.out.println("데이터 수정 실패");
                System.out.println(e.getMessage());
                e.printStackTrace();
        }
        close();
        return result;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
        //대륙이름이나 국가에 포함된 데이터 찾아오기
//검색어를 매개변수로 받아서 검색어가 포함된 데이터를 찾아오는 메소드
public List<Covid> searchCovid(String word){
        List<Covid> list = new ArrayList<Covid>();
        connect();
        try {
                        pstmt = con.prepareStatement(
                                "select * from covid19 "
                                + "where region like ? or nation like ?");
                        pstmt.setString(1, "%" + word + "%");
                        pstmt.setString(2, "%" + word + "%");

                        ResultSet rs = pstmt.executeQuery();
                        while(rs.next()) {
                                Covid covid = new Covid();

                                covid.setNum(rs.getInt("num"));
                                covid.setRegion(rs.getString("region"));
                                covid.setNation(rs.getString("nation"));
                                covid.setConfirmcount(
                                rs.getInt("confirmcount"));
                                list.add(covid);
                        }
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
                rs.close();

        }catch(Exception e) {
                System.out.println("데이터 검색 실패");
                System.out.println(e.getMessage());
                e.printStackTrace();
        }
        close();
        return list;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//전체 데이터 개수를 구하는 메소드
//select count(*) from covid19;
public int getCount(){
        int result = -1;
        connect();
        try {
                pstmt = con.prepareStatement(
                        "select count(*) from covid19");
                ResultSet rs = pstmt.executeQuery();
                if(rs.next()) {
                        result = rs.getInt("count(*)");
                }
                rs.close();
        }catch(Exception e) {
                System.out.println("데이터 개수 세기 실패");
                System.out.println(e.getMessage());
                e.printStackTrace();
        }
        close();
        return result;
}
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
//페이지 번호와 페이지 당 데이터 개수를 받아서 페이지 번호에 해당하는 데이터를
//조회하는 메소드
//실제 업무인 경우는 매개변수로 검색어가 포함됩니다.
public List<Covid> pageCovid(int pageno, int pagecnt){
        List<Covid> list = new ArrayList<Covid>();
        connect();
        try {
                pstmt = con.prepareStatement("select * "
                                + "from covid19 "
                                + "limit ?, ?");
                pstmt.setInt(1, pagecnt *(pageno-1));
                pstmt.setInt(2, pagecnt);

                ResultSet rs = pstmt.executeQuery();
                while(rs.next()) {
                        Covid covid = new Covid();
                        covid.setNum(rs.getInt("num"));
                        covid.setNation(rs.getString("nation"));
                        covid.setDeathcount(rs.getInt("deathcount"));
                        list.add(covid);
                }
                rs.close();
```

COVID19

- DAO 역할을 수행할 클래스인 CovidDAO 클래스를 생성

```
            }catch(Exception e) {
                        System.out.println("데이터 검색 실패");
                        System.out.println(e.getMessage());
                        e.printStackTrace();
            }
            close();
            return list;
    }
}
```

COVID19

❖ CovidMain 클래스를 생성

```
import java.util.List;
import java.util.Scanner;
public class CovidMain {
	public static void main(String[] args) {
		// 데이터베이스 사용 객체를 생성
		CovidDAO dao = CovidDAO.sharedInstance();
		// 키보드 입력 객체를 생성
		Scanner sc = new Scanner(System.in);
		// 여러 개의 데이터를 저장하기 위한 변수
		List<Covid> list = null;
		// 하나의 데이터를 저장하기 위한 변수
		Covid covid = null;
		// 삽입, 삭제, 갱신의 결과를 저장하기 위한 변수
		int result = -1;
		// Covid 각각을 위한 변수
		int num = -1;
		String region = null;
		String nation = null;
		int pop = -1;
		int confirmcount = -1;
		int deathcount = -1;
		// 입력을 받기 위한 임시변수
		String temp = null;
```

COVID19

- CovidMain 클래스를 생성

```
// mainloop 라고 이름을 붙인 이유는
// 내부에서 switch 를 사용할 것이고
// 7번을 누를 때 반복문을 한번에 빠져나오기 위해서입니다.
mainloop: while (true) {
        System.out
                .println("1.전체보기  2.2개씩 보기 3.상세보기 " + "4.대륙이나 국가로 검색₩n" + "5.데이터삽입 6.데이터수정 7.데이터삭제 " + "8.프로그램 종료");
        System.out.print("메뉴 입력:");
        String menu = sc.nextLine();
        switch (menu) {
        case "1":
                // 전체 가져오기를 호출
                list = dao.allCovid();
                // 데이터 출력
                System.out.printf("%5s%15s%10s₩n", "번호", "국가", "사망자수");
                for (Covid imsi : list) {
                        System.out.printf("%5d₩t%15s₩t%10d₩n", imsi.getNum(), imsi.getNation(), imsi.getConfirmcount());
                }

                break;
```

COVID19

- CovidMain 클래스를 생성

```
case "2":
//전체 데이터 개수 가져오기
int cnt = dao.getCount();
//System.out.println("데이터개수:" + cnt);
//페이지 수 만들기 - 페이지당 데이터 개수는 2
int pagesu =
            (int)((double)cnt/2 + (double)(2-1)/2);
//System.out.println("페이지개수:" + pagesu);

//첫번째 페이지의 데이터 가져오기
list = dao.pageCovid(1, 2);
for(Covid imsi : list) {
            System.out.println(imsi);
}
//현재 페이지 번호 저장
int pageno = 1;
```

COVID19

- CovidMain 클래스를 생성

```
while(true) {
        //아무키나 누르고 Enter 치면 종료
        //그냥 Enter 치면 다음 페이지의 데이터도 가져와서 출력하기
        System.out.println("아무키나 누르면 종료");
        System.out.print("Enter만 누르면 다음페이지 데이터 가져오기");
        temp = sc.nextLine();
        if(temp.trim().length() == 0) {
                pageno = pageno + 1;
                if(pageno > pagesu) {
                        System.out.println("더이상 가져올 데이터 없음");
                }else {
                        //pageno 에 해당하는 데이터 가져오기
                        List<Covid>currentData = dao.pageCovid(pageno, 2);
                        list.addAll(currentData);
                        for(Covid imsi : list) {
                                System.out.println(imsi);
                        }
                }
        }else {
                break;
        }
}
break;
```

COVID19

- CovidMain 클래스를 생성

```
                case "3":
                // 조회할 번호를 입력받기
                System.out.print("조회할 번호:");
                temp = sc.nextLine();
                try {
                        num = Integer.parseInt(temp);
                        // 데이터 가져오기
                        covid = dao.getCovid(num);
                        if (covid == null) {
                                System.out.println("해당하는 번호의 데이터가
없습니다.");
                        } else {
                                System.out.println(covid);
                                //System.out.printf("%5d₩t%15s₩n",
covid.getNum(), covid.getNation());
                        }
                } catch (Exception e) {
                        System.out.println("정수를 입력하세요!!");
                        break;
                }

                break;
```

COVID19

- CovidMain 클래스를 생성

```
case "4":
        System.out.print("조회할 대륙이나 국가:");
        temp = sc.nextLine();
        list = dao.searchCovid(temp);
        if(list.size() == 0) {
                System.out.println("조회된 데이터가 없습니다.");
        }else {
                for(Covid imsi : list) {
                System.out.printf("%5d₩t%10s₩t%10s₩t%10d₩n",
                        imsi.getNum(), imsi.getRegion(),
                        imsi.getNation(), imsi.getConfirmcount());
                }
        }

        break;
```

COVID19

- CovidMain 클래스를 생성

```
case "5":
// 대륙과 국가를 입력받기
System.out.print("대륙:");
region = sc.nextLine();
System.out.print("국가:");
nation = sc.nextLine();

// 인구수, 확진자수, 사망자수 입력받기
System.out.print("인구 수:");
temp = sc.nextLine();
pop = Integer.parseInt(temp);

System.out.print("확진자 수:");
temp = sc.nextLine();
confirmcount = Integer.parseInt(temp);

System.out.print("사망자 수:");
temp = sc.nextLine();
deathcount = Integer.parseInt(temp);
```

COVID19

❖ CovidMain 클래스를 생성

```
// 데이터베이스 메소드에 넘겨주기 위해서 입력받은 데이터를
// 1개로 만들기
covid = new Covid();
covid.setNum(num);
covid.setRegion(region);
covid.setNation(nation);
covid.setConfirmcount(confirmcount);
covid.setPop(pop);
covid.setDeathcount(deathcount);

// 데이터베이스 메소드 호출
result = dao.insertCovid(covid);
// 결과 사용
if (result > 0) {
		System.out.println("데이터 삽입 성공");
} else {
		System.out.println("데이터 삽입 실패");
}

break;
```

COVID19

- CovidMain 클래스를 생성

```
case "6":
//기본키 값 입력받기
System.out.print("수정할 번호:");
num = sc.nextInt();
sc.nextLine(); //Enter를 가져가기 위해서 추가
//데이터 찾아오기
covid = dao.getCovid(num);
if(covid == null) {
		System.out.println("없는 번호입니다.");
}else {
		//대륙입력받기
		System.out.println("수정하지 않으려면 Enter");
		System.out.print("대륙 입력(이전-" +
			covid.getRegion() + "):");
		region = sc.nextLine();
		//글자 수가 0이면 이전 값 갖기
		if(region.trim().length() == 0) {
			region = covid.getRegion();
		}
```

COVID19

- CovidMain 클래스를 생성

```
//국가 입력받기
System.out.println("수정하지 않으려면 Enter");
System.out.print("국가 입력(이전-" +
            covid.getNation() + "):");
nation = sc.nextLine();
//글자 수가 0이면 이전 값 갖기
if(nation.trim().length() == 0) {
            nation = covid.getNation();
}
//인구 입력받기
System.out.println("수정하지 않으려면 Enter");
System.out.print("인구 입력(이전-" +
            covid.getPop() + "):");
temp = sc.nextLine();
if(temp.trim().length() == 0) {
            pop = covid.getPop();
}else {
            pop = Integer.parseInt(temp);
}
```

COVID19

- CovidMain 클래스를 생성

```
//확진자 수 입력받기
System.out.println("수정하지 않으려면 Enter");
System.out.print("확진자 수 입력(이전-" +
        covid.getConfirmcount() + "):");
temp = sc.nextLine();
if(temp.trim().length() == 0) {
        confirmcount = covid.getConfirmcount();
}else {
        confirmcount = Integer.parseInt(temp);
}
//사망자 수 입력받기
System.out.println("수정하지 않으려면 Enter");
System.out.print("사망자 수 입력(이전-" +
        covid.getDeathcount() + "):");
temp = sc.nextLine();
if(temp.trim().length() == 0) {
        deathcount = covid.getDeathcount();
}else {
        deathcount = Integer.parseInt(temp);
}
```

COVID19

- CovidMain 클래스를 생성

```
                    //입력받은 데이터를 하나로 만들기
                    covid.setRegion(region);
                    covid.setNation(nation);
                    covid.setPop(pop);
                    covid.setConfirmcount(confirmcount);
                    covid.setDeathcount(deathcount);

                    result = dao.updateCovid(covid);

                    if(result > 0) {
                              System.out.println("수정 성공");
                    }else {
                              System.out.println("수정 실패");
                    }
          }

          break;
```

COVID19

- CovidMain 클래스를 생성

```
case "7":
//삭제할 번호 입력
System.out.print("삭제할 번호:");
temp = sc.nextLine();
num = Integer.parseInt(temp);
//데이터의 존재 여부를 확인
covid = dao.getCovid(num);
if(covid == null) {
		System.out.println("삭제할 수 없는 번호입니다.");
}else {
		//정말로 삭제할 것인지 확인
		System.out.print("정말로 삭제(Y):");
		temp = sc.nextLine();
```

COVID19

- CovidMain 클래스를 생성

```
                    //영문을 입력받아서 비교
                    //trim은 좌우 공백 제거
                    //toUpperCase는 모두 대문자로 변경
                    if(temp.trim().toUpperCase().equals("Y")) {
                                //삭제
                                result = dao.deleteCovid(num);
                                //삭제 결과 사용
                                if(result > 0) {
                                            System.out.println("삭제 성공");
                                }else {
                                            System.out.println("삭제 실패");
                                }
                    }
        }

        break;
```

COVID19

❖ CovidMain 클래스를 생성

```
                    case "8":
                    System.out.println("프로그램 종료");
                    break mainloop;
                default:
                    System.out.println("잘못된 메뉴 선택 !!!");
                    break;
                }
            }
            // 키보드 입력 객체 닫기
            sc.close();
    }
}
```