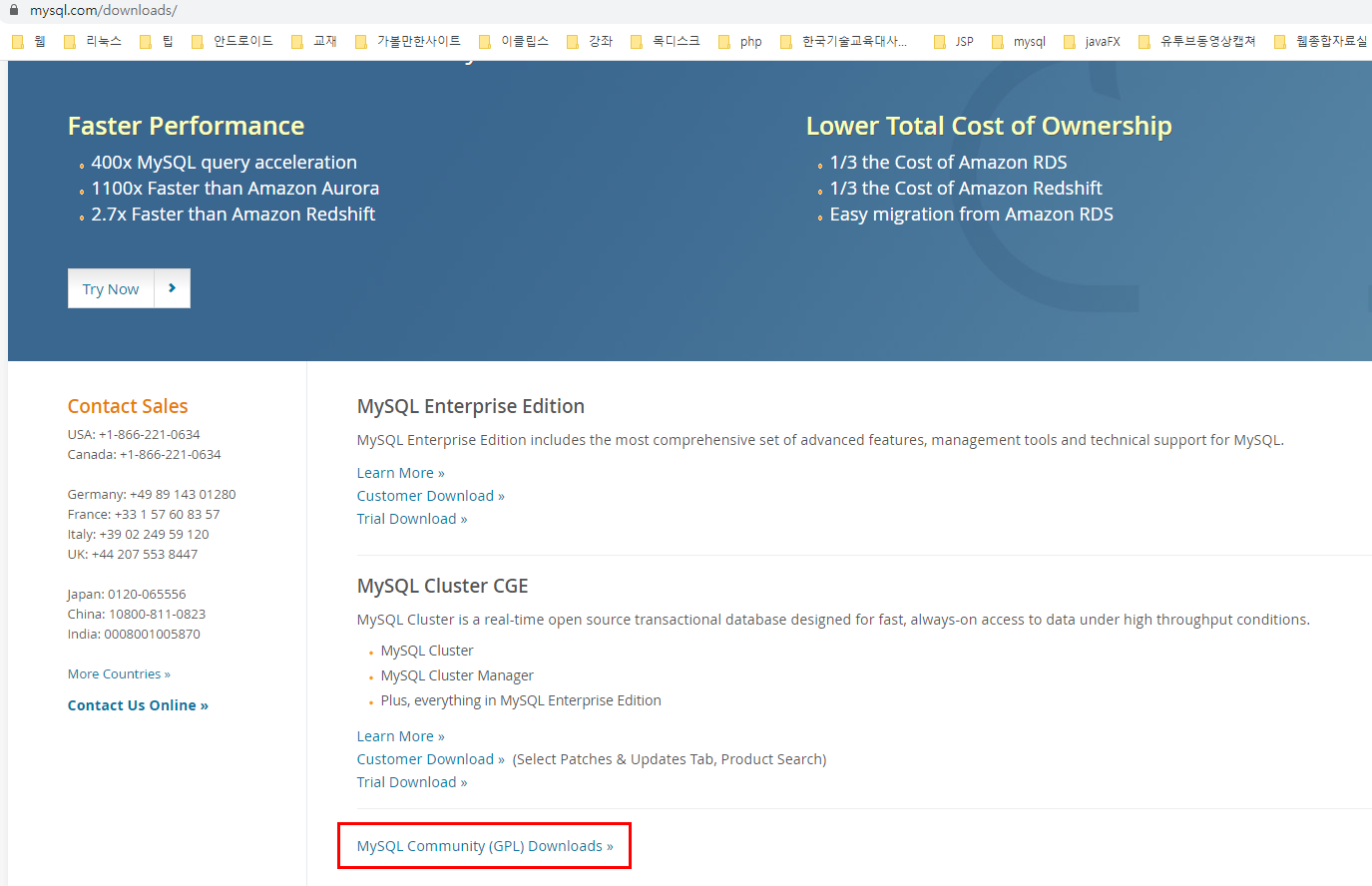
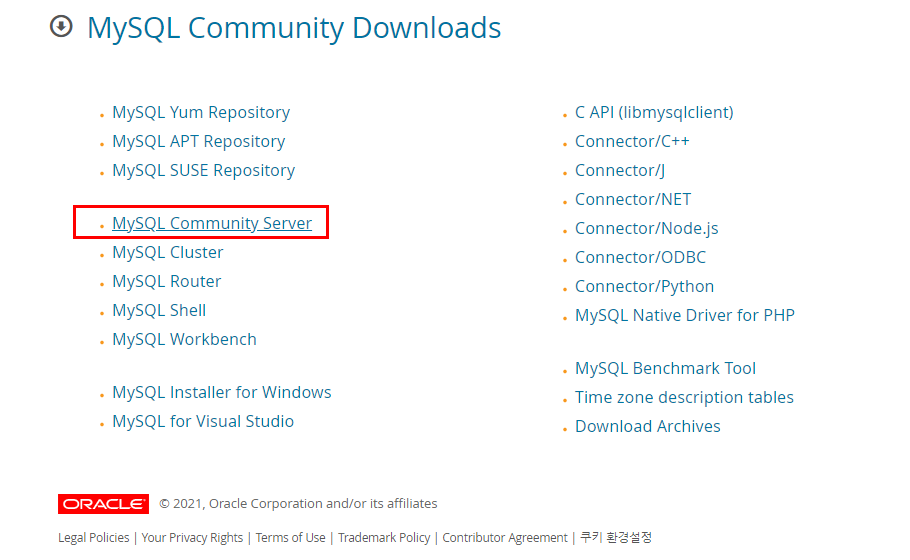
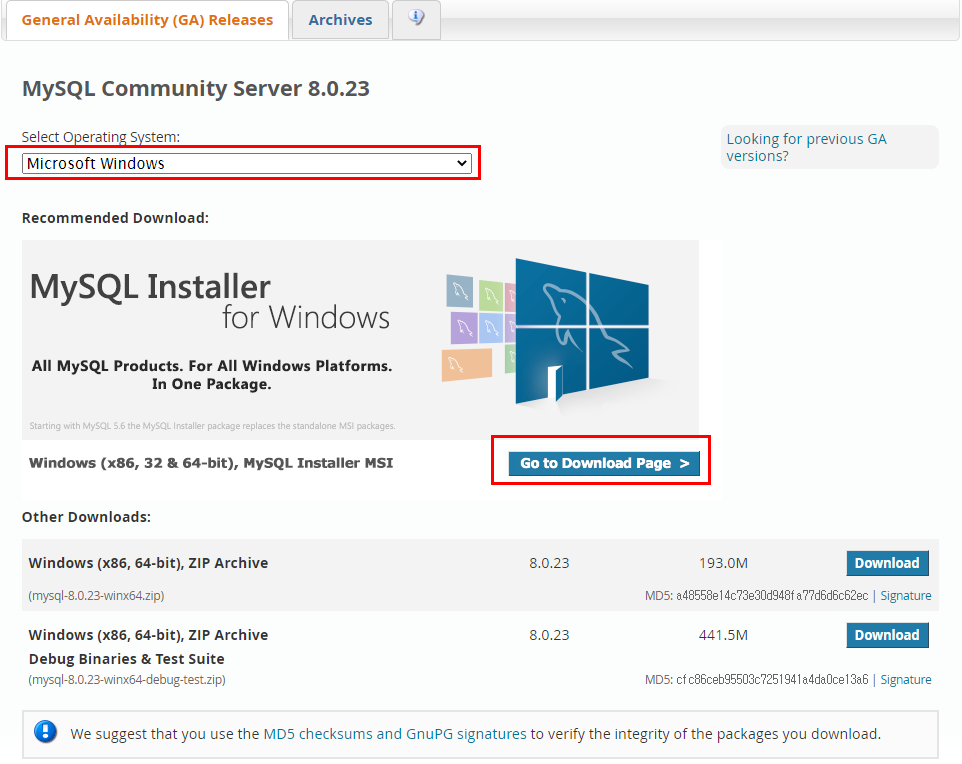
성적관리프로그램 프로젝트

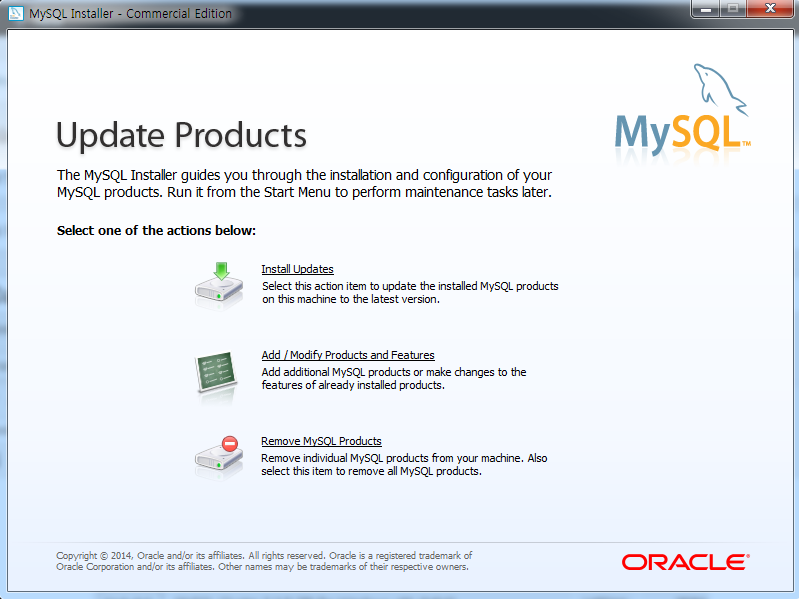
MySQL 설치 ① <http://www.mysql.com/downloads/>

다운 받으실 링크는 아래와 같습니다.

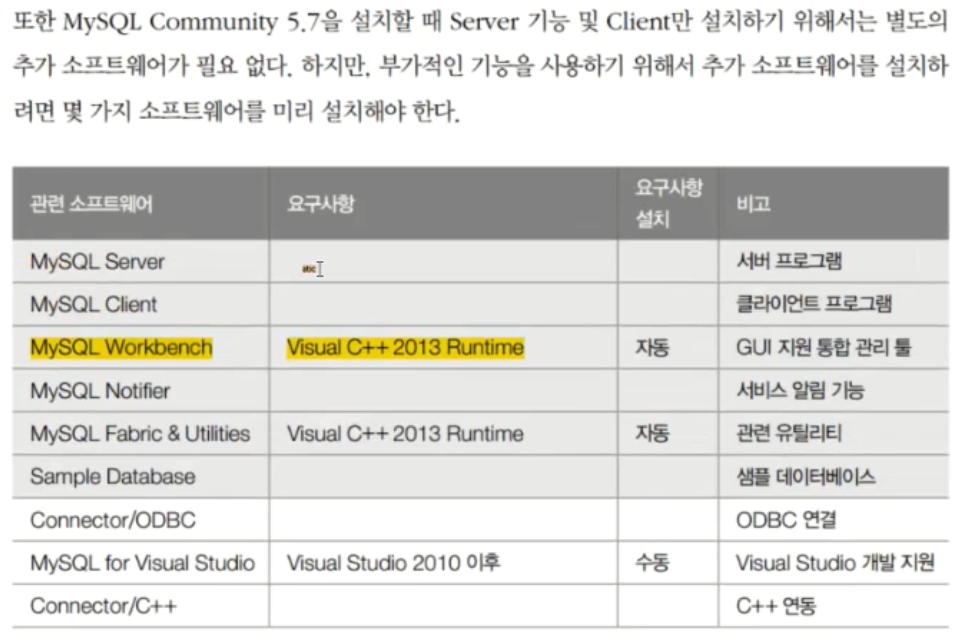




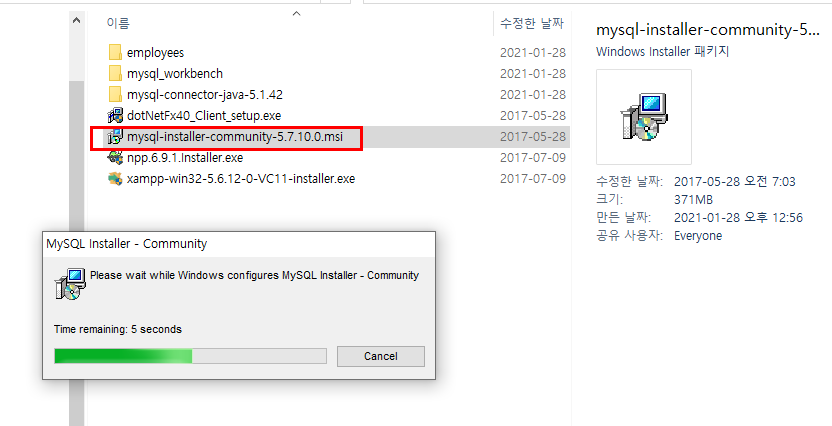


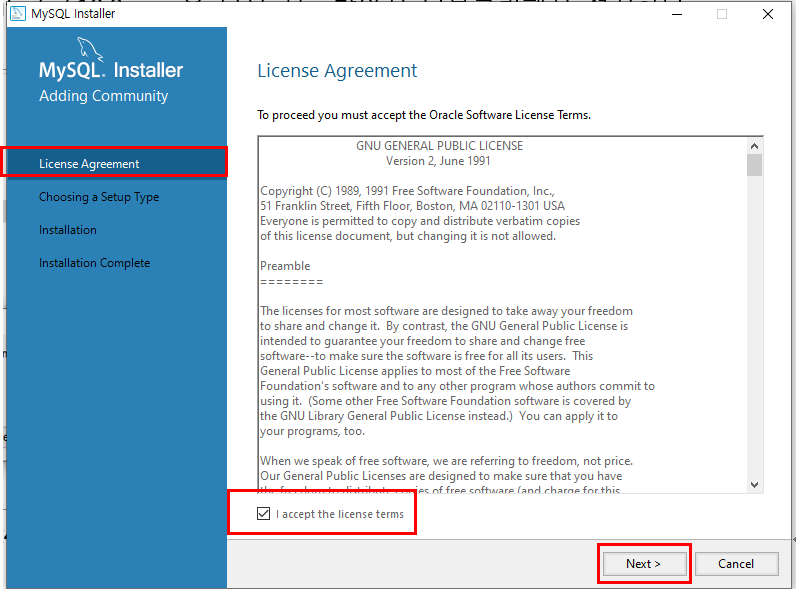


* MYSQL COMMUNITY 5.7 설치할때 SERVER 기능 및 CLIENT만 설치하기 위해서는 별도의 추가 소프트웨어가 필요없다.



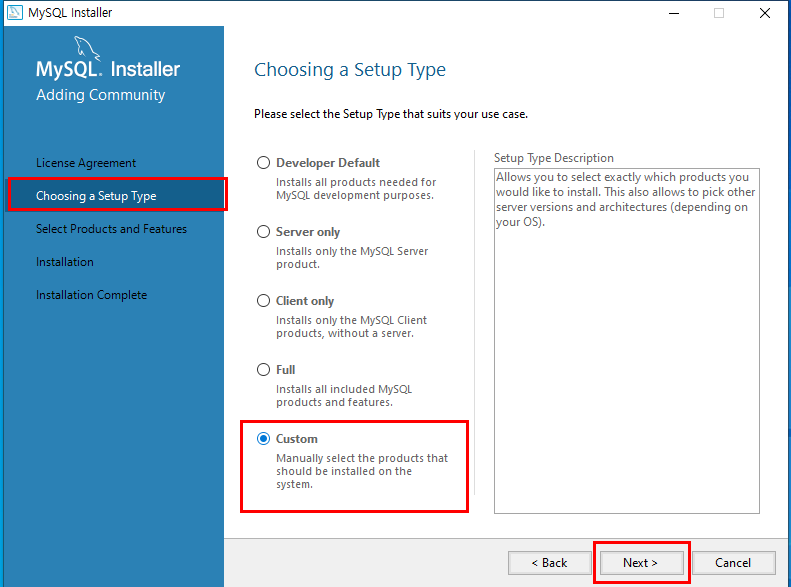
1. 공유폴더에 mysql-installer-community-5.7.10.0.msi 을 다운로드 받아서 더블클릭해서 설치한다.



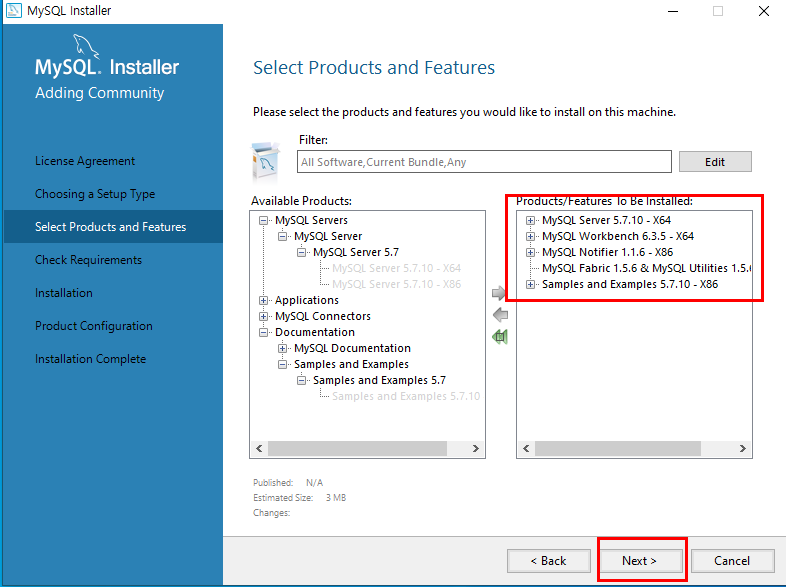


2. 닷넷 프레임워크가 없다면 dotNetFx40\_Client\_setup.exe 파일을 다운로드 받아서 설치하고 재부팅한다.

3. 설치진행하면 동의한후 다음아래와 같이 custom으로 설치한다.



4. 다음아래와 같이 5개를 각각 선택해서 설치한다.



1)MySQL server 5.7.10 x64(64bit)

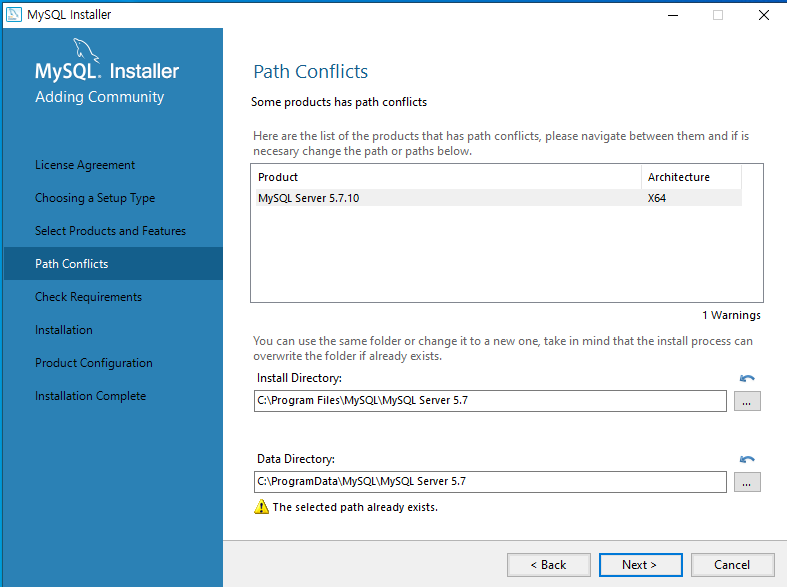
2)MySQL Workbench 6.3.5 x64(64bit) –GUI 통합관리툴 Visual C++ 2013 Runtime 요구되어짐

3)MySQL Notifier 1.1.6 x86(64bit가 없으면 32bit를 설치해도 된다.)- 서비스 알림기능

4)MySQL Utilities -> MySQL Fabric & Utilities 1.5,6 -관련유틸리티 Visual C++ 2013 Runtime 요구되어짐

5)Documentation->Samples and Examples 5.7.10 –x86 -샘플데이타베이스

5. Path Conflicts 가 발생하면 기존의 파일에 적용시키고 next를 선택한다.



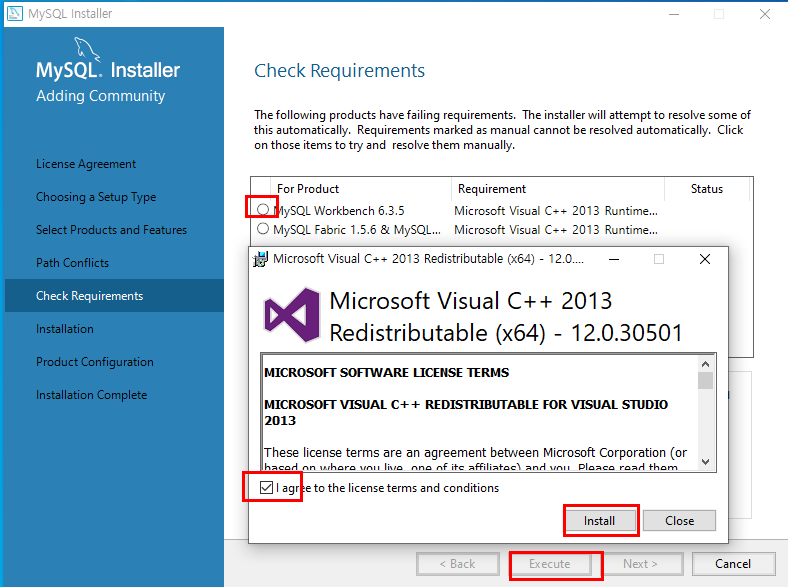
6. 다음 MySql WorkBench를 선택해서 execute 버튼을 선택해서 다음 아래와 같이 진행한다. (인터넷연결되어있어된다. 아니면 Visual C++ 2013 Runtime 다운로드 받아야됨)

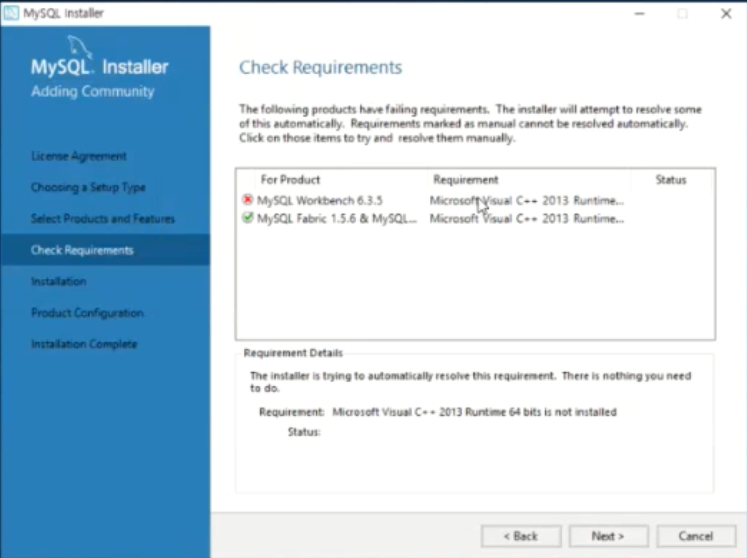
-MySQL Workbench 6.3.5 x64(64bit) –Visual C++ 2013 Runtime 요구되어짐

-MySQL Fabric & Utilities 1.5, Visual C++ 2013 Runtime 요구되어지면

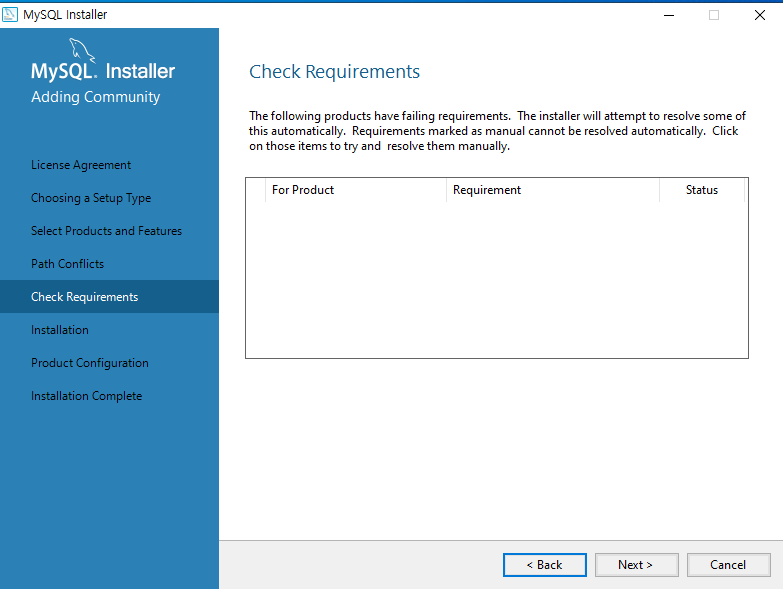
-Modify Setup MySQL Fabric & Utilities 1.5, Visual C++ 2013 Runtime 요구되어지면 (Repair, Uninstall, close)

선택메시지가 나오면 Repair 선택을 진행할 것

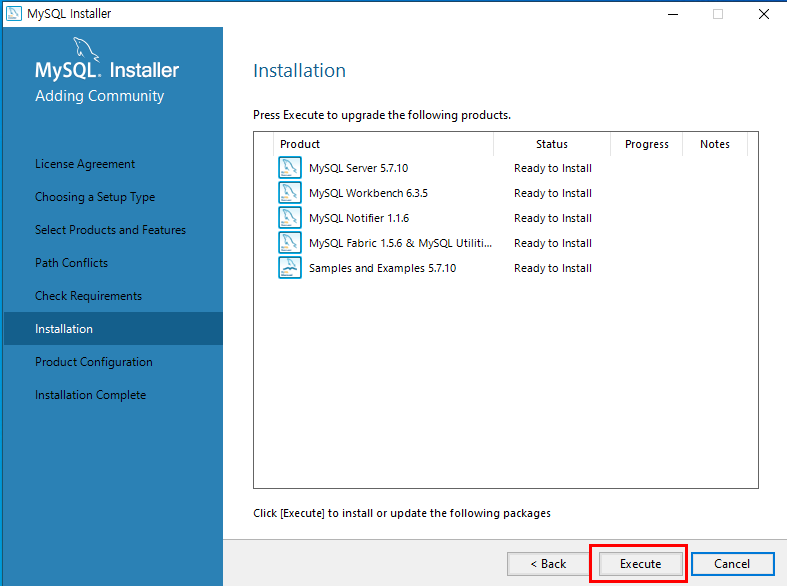




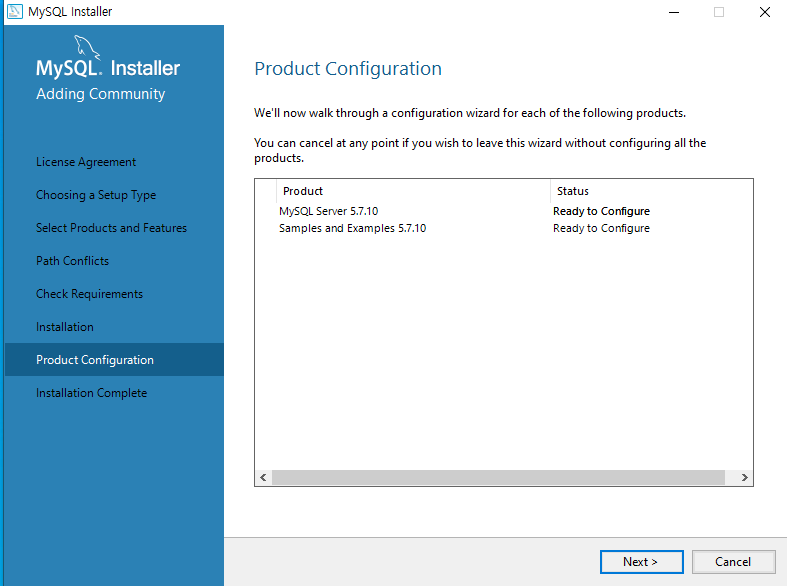
혹시 빨간색 X가 나오면 back을 해서 next를 누르면 설치되서 두 개 파일이 나오지 않음.



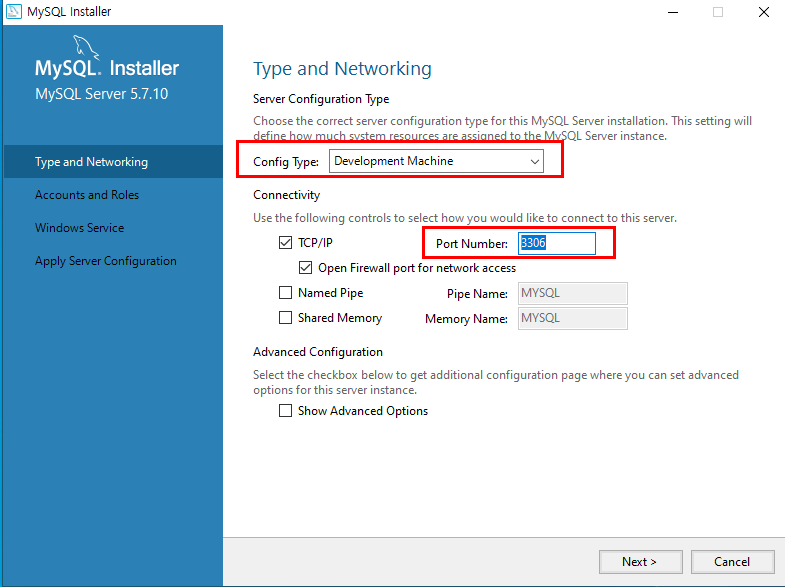
7. 다음 5개 파일을 설치하기 위해서 execute 버튼을 누르면 다음 아래와 같이 설치된다.



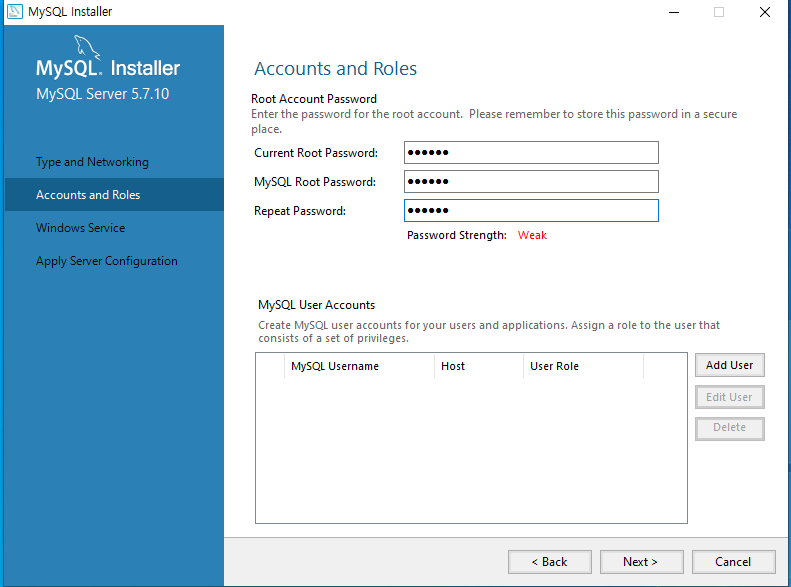
8. Product configuration에서 next버튼을 누른다.



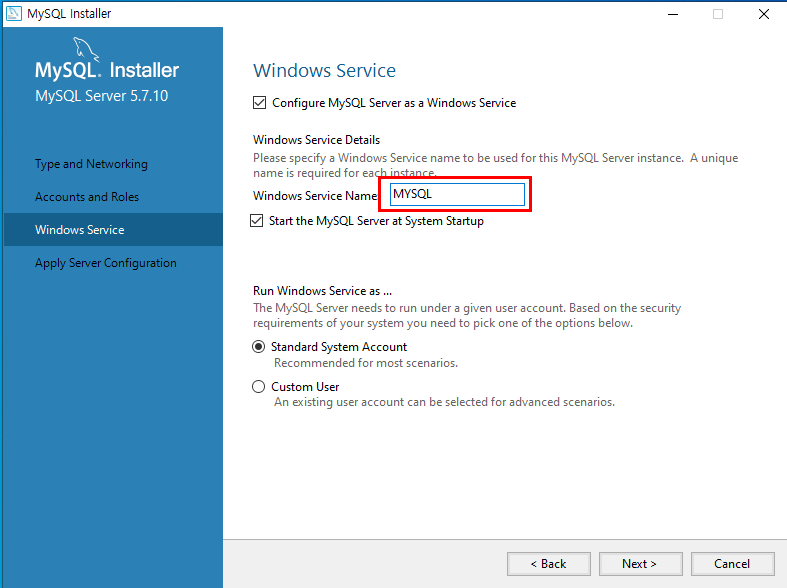
9. 다음 config Type : Development Machine을 선택하고 포트번호 3306을 잘 기억하고 나머지는 아래와 같이 셋팅한다.



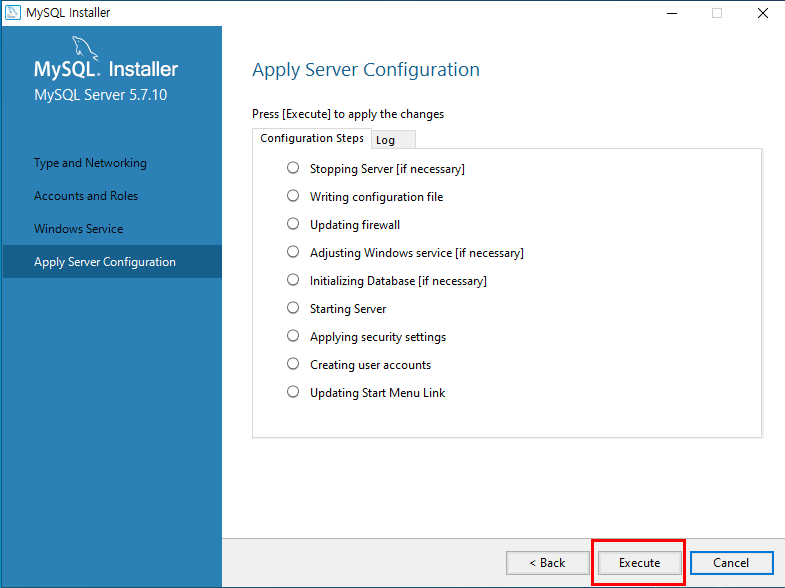
10. 비밀번호를 입력한다. 당연히 어렵게 써야한다. (123456)

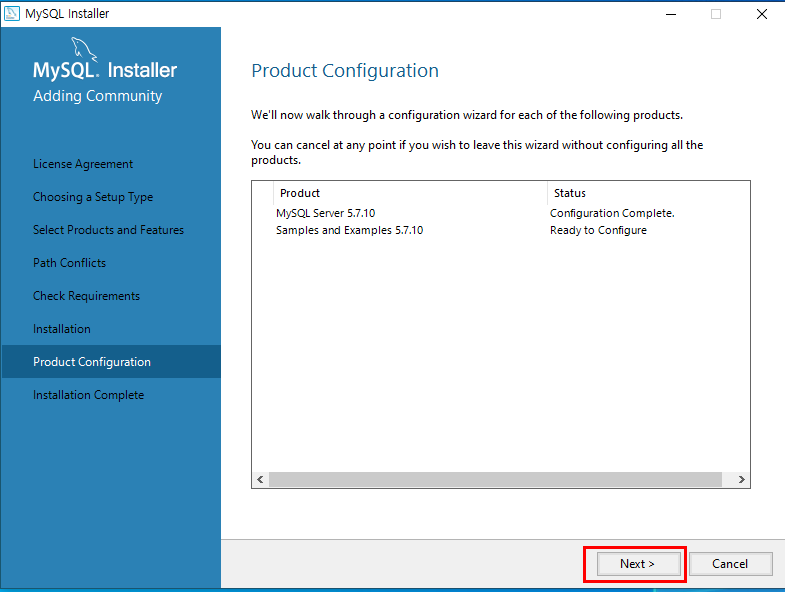


11. 이름을 MySQL57를 MySQL로 수정하고 나머지는 그대로 두고 다음을 누른다.

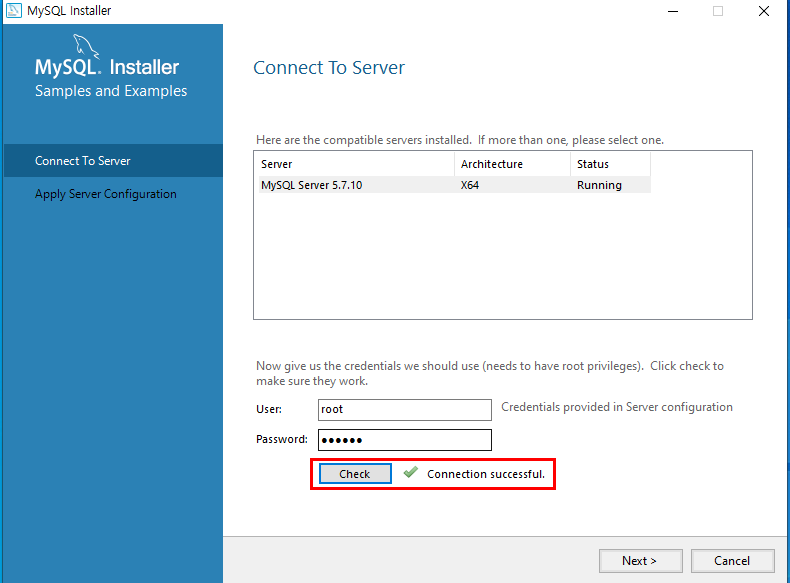


12. 다음 9개를 설치하기 위해서 execute 버튼을 클릭한다.

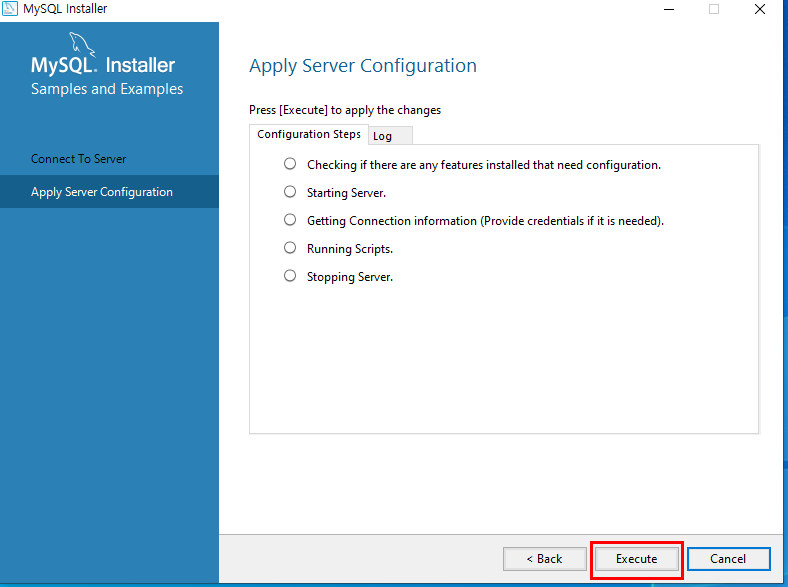


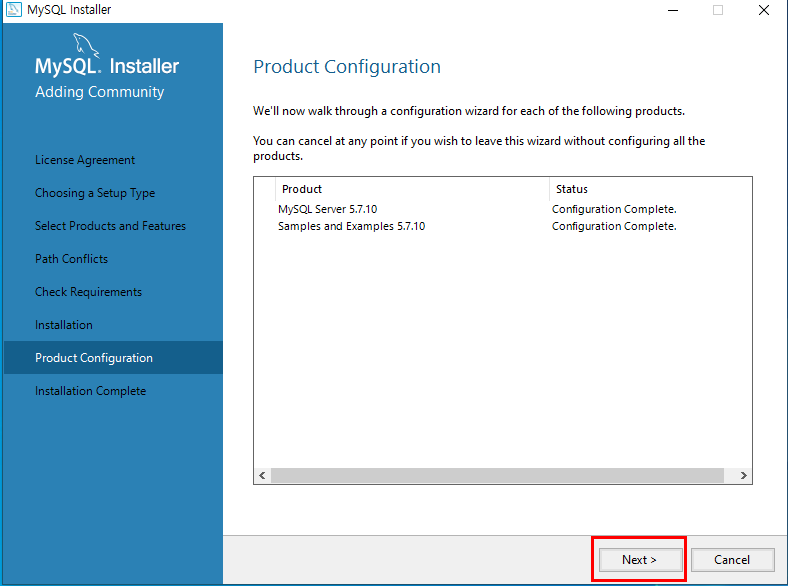


13.. 다음은 비밀번호를 check를해서 확인하고 next 버튼을 클릭한다.

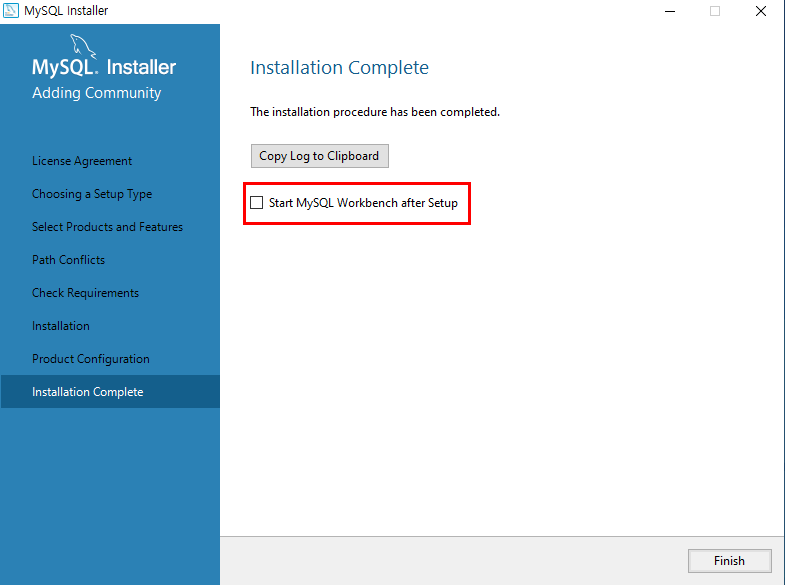


14. 다음 서버환경설정을 execute -> finish -> next 버튼을 눌러서 실행한다.



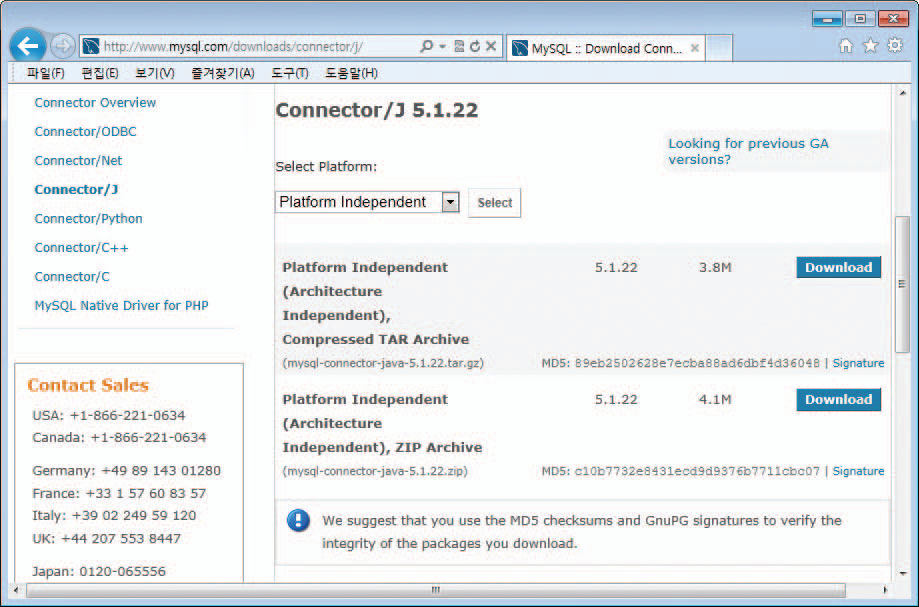


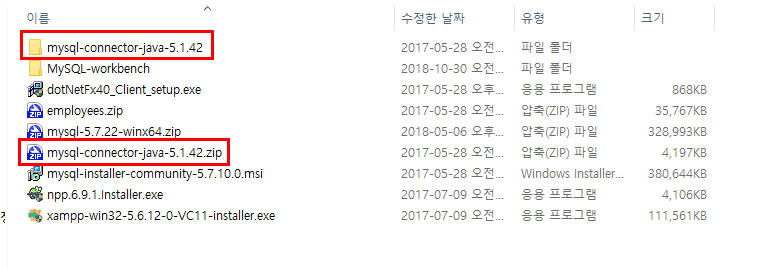
15. 다음 workBanch를 체크를 해제하고 finish 누른다.



16. JDBC 설치한다. (여기서부터 우리가 해야된다)

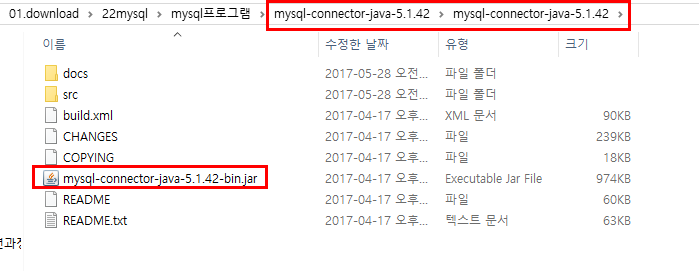
: 6반 공유방에서 다운받는다. 그리고 압축을푼다.

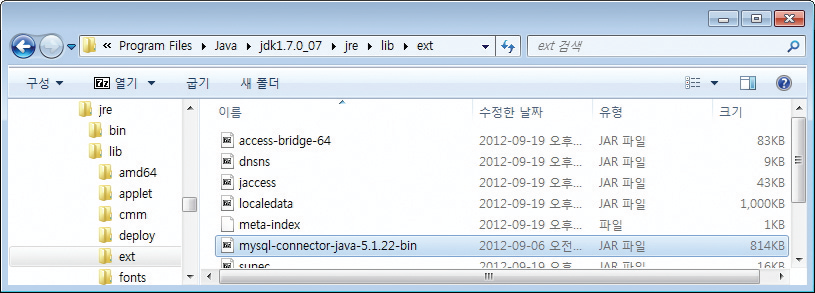




▶Java용 드라이버를 복사한다.

* mysql-connector-java-5.1.42-bin.jar 파일을 C 드라이브 Progema Files -> java -> jdk1.7.0 ->JRE ->LIB ->EXT 디렉터리에 붙여넣는다
* 주의 : JRE만 설치한 경우는 JRE 설치 디렉터리 밑의 LIB\EXT 디렉터리에 복사도 해도 된다.

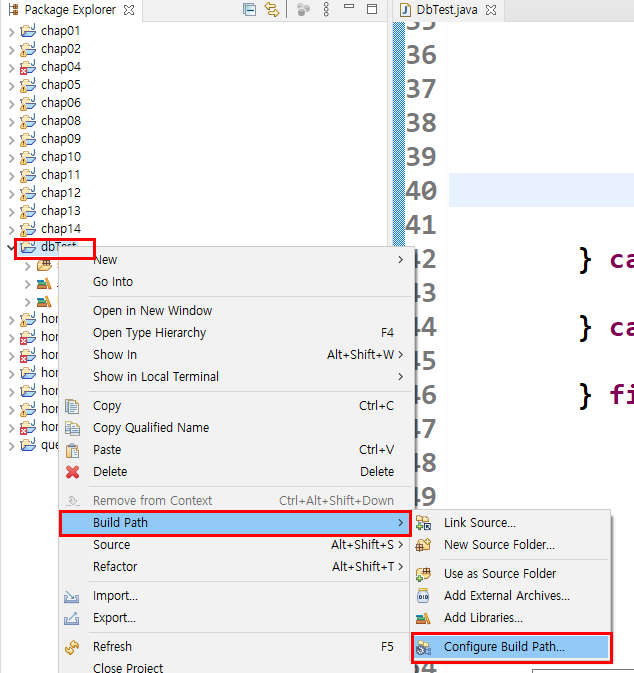




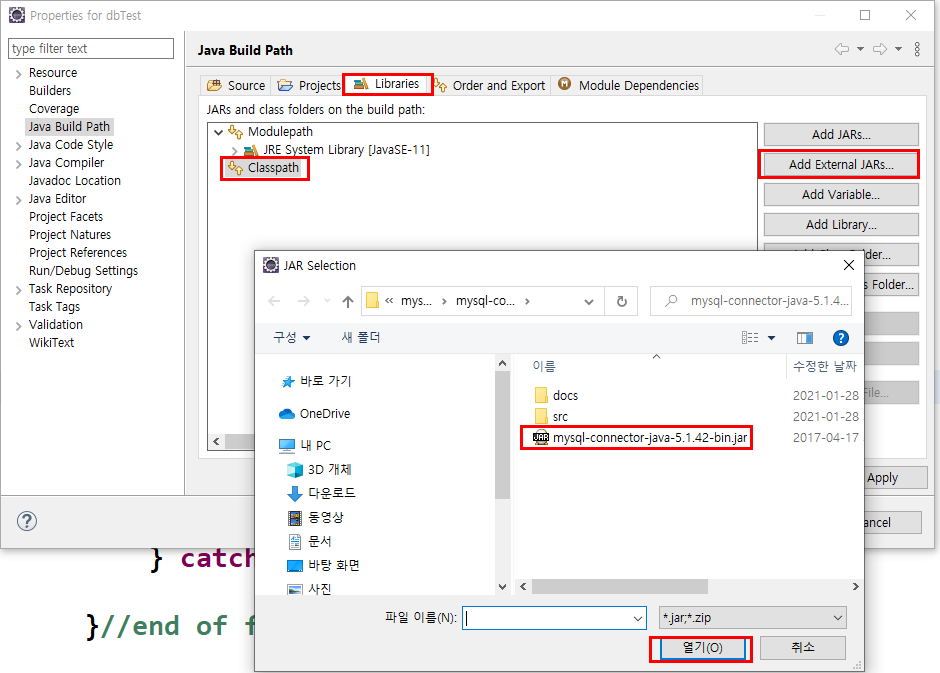
* 만약, jdk11 버전부터 jre 폴더가 없다면 다음아래와 같이 진행한다.

이클립스에서 해당프로젝트를 만들고 에서 우클릭을 통해서 Build Path -> configure Build Path를 선택한다.

1)

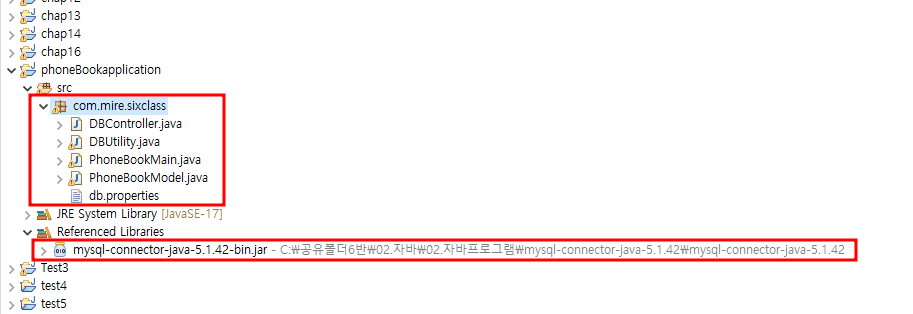


2)다음아래와 같이 mysql-connector-java-5.1.42-bin.jar 파일을 가져온다.



3)다음아래소스를 실행해서 성공하면 큰 문제없이 진행이 된다.

☞학생 데이터베이스 연결해서 프로젝트 진행하기



**데이터베이스 가서 다음과 같은 내용으로 모델링한다.**

**create database student\_db;**

**use student\_db;**

**create table student\_tbl(**

**id int not null,**

**name CHAR(20) not null,**

**gender CHAR(10) not null,**

**kor int not null,**

**eng int not null,**

**math int not null,**

**total int not null,**

**avr double not null,**

**grade char(1) not null,**

**PRIMARY KEY(id)**

**);**

1. db.properties

DRIVER=com.mysql.jdbc.Driver

URL=jdbc:mysql://localhost/phoneBookdb

userID=root

userPassword=123456

주의: URL=jdbc:mysql://localhost/phoneBookDB?autoReconnect=true&verifyServerCertificate=false&useSSL=true 이렇게 처리하는 것은 결과값에 빨간줄이 나온 것을 방지하기 위함.

1. DBUtil.java

**package** studentProgram;

**import** java.io.FileNotFoundException;

**import** java.io.FileReader;

**import** java.io.IOException;

**import** java.sql.DriverManager;

**import** java.sql.SQLException;

**import** java.util.Properties;

**import** com.mysql.jdbc.Connection;

**public** **class** DBUtility {

**public** **static** Connection getConnection() {

Connection con=**null**;

FileReader fr=**null**;

**try** {

fr=**new** FileReader("src\\studentProgram\\db.properties");

Properties properties=**new** Properties();

properties.load(fr);

String Driver=properties.getProperty("Driver");

String URL=properties.getProperty("URL");

String userID=properties.getProperty("userID");

String userPW=properties.getProperty("userPW");

//드라이버 적재

Class.*forName*(Driver);

//데이터베이스연결

con=(Connection) DriverManager.*getConnection*(URL, userID, userPW);

} **catch** (ClassNotFoundException e) {

System.***out***.println("mysql database connection fail");

e.printStackTrace();

} **catch** (SQLException e) {

System.***out***.println("mysql database connection fail");

e.printStackTrace();

} **catch** (FileNotFoundException e) {

System.***out***.println("file db.properties connection fail");

e.printStackTrace();

} **catch** (IOException e) {

System.***out***.println("file db.properties connection fail");

e.printStackTrace();

}**finally** {

**try** {

fr.close();

} **catch** (IOException e) {

e.printStackTrace();

}

}

**return** con;

}

}

1. DBController.java

package studentProgram;

import java.sql.SQLException;

import java.util.ArrayList;

import java.util.List;

import com.mysql.jdbc.Connection;

import com.mysql.jdbc.PreparedStatement;

import java.sql.ResultSet;

public class DBController {

//데이터 입력 : insert 쿼리문

public static int dataInsert(StudentModel sd) {

//데이터베이스 접속

Connection con = DBUtility.getConnection();

//데이터베이스 입력 반환값

int result = 0;

//데이터베이스 연결 확인

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

//삽입 명령문 전달(학번, 이름, 성별, 국어, 영어, 수학, 총점, 평균, 등급)

String insertQuery = "insert into studentprogram.studentdatatbl values (?, ?, ?, ?, ?, ?, ?, ?, ?)";

PreparedStatement ps = null;

try {

//insert query binding(id, name, gender, kor, eng, math, total, avr, grade)

ps = (PreparedStatement)con.prepareStatement(insertQuery);

ps.setString(1, sd.getId());

ps.setString(2, sd.getName());

ps.setString(3, sd.getGender());

ps.setInt(4, sd.getKor());

ps.setInt(5, sd.getEng());

ps.setInt(6, sd.getMath());

ps.setInt(7, sd.getTotal());

ps.setDouble(8, sd.getAvr());

//등급 : 문자형을 String 값으로 변환

ps.setString(9, String.valueOf(sd.getGrade()));

//쿼리 실행 명령

result = ps.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close();

ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

//결과값 반환

return result;

}

//데이터 조회 : select 쿼리문

public static List<StudentModel> dataSearch(String searchData, int num) {

final int NAME=1, ID=2;

//데이터베이스 접속

Connection con = DBUtility.getConnection();

//데이터베이스 연결 확인

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

//데이터베이스 입력 반환값

List<StudentModel> list = new ArrayList<StudentModel>();

ResultSet resultSet = null;

//조회 명령문 전달(학번, 이름)

String searchQuery = null;

PreparedStatement ps = null;

try {

switch(num) {

case NAME :

searchQuery = "select\* from studentprogram.studentdatatbl where name=?";

ps = (PreparedStatement)con.prepareStatement(searchQuery);

ps.setString(1, searchData); break;

case ID :

searchQuery = "select\* from studentprogram.studentdatatbl where id=?";

ps = (PreparedStatement)con.prepareStatement(searchQuery);

ps.setInt(2, Integer.parseInt(searchData)); break;

}

//쿼리 실행 명령

resultSet = ps.executeQuery();

//resultSet을 list에 저장함수 호출

resultSetToList(resultSet, list);

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close(); ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

return list;

}

public static List<StudentModel> dataSelect() {

Connection con = DBUtility.getConnection();

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

List<StudentModel> list = new ArrayList<StudentModel>();

ResultSet resultSet = null;

String selectQuery = null;

PreparedStatement ps = null;

try {

selectQuery = "select\* from studentprogram.studentdatatbl";

ps = (PreparedStatement)con.prepareStatement(selectQuery);

//쿼리 실행 명령

resultSet = ps.executeQuery();

//resultSet을 list에 저장함수 호출

resultSetToList(resultSet, list);

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close(); ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

//결과값 반환

return list;

}

public static List<StudentModel> dataSort(int menu) {

final int NAME=1, TOTAL=2;

//데이터베이스 접속

Connection con = DBUtility.getConnection();

//데이터베이스 연결 확인

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

//데이터베이스 입력 반환값

List<StudentModel> list = new ArrayList<StudentModel>();

ResultSet resultSet = null;

String sortQuery = null;

PreparedStatement ps = null;

switch(menu) {

case NAME: sortQuery = "select\* from studentprogram.studentdatatbl order by name"; break;

case TOTAL: sortQuery = "select\* from studentprogram.studentdatatbl order by total desc"; break;

}

try {

//쿼리 실행 명령

ps = (PreparedStatement)con.prepareStatement(sortQuery);

resultSet = ps.executeQuery();

//resultSet을 list에 저장함수 호출

resultSetToList(resultSet, list);

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close(); ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

//결과값 반환

return list;

}

//resultSet을 list에 저장

private static void resultSetToList(ResultSet resultSet, List<StudentModel> list) {

try {

while(resultSet.next()) {

String id = resultSet.getString(1);

String name = resultSet.getString(2);

String gender = resultSet.getString(3);

int kor = resultSet.getInt(4);

int eng = resultSet.getInt(5);

int math = resultSet.getInt(6);

int total = resultSet.getInt(7);

double avr = resultSet.getDouble(8);

char grade = resultSet.getString(9).charAt(0);

StudentModel sd = new StudentModel(id, name, gender, kor, eng, math, total, avr, grade);

list.add(sd);

}

} catch (SQLException e) {

e.printStackTrace();

}

}

//데이터 수정

public static int dataUpdate(StudentModel sd, int menu) {

final int KOR=1, ENG=2, MATH=3;

//데이터베이스 접속

Connection con = DBUtility.getConnection();

//데이터베이스 입력 반환값

int result = 0;

//데이터베이스 연결 확인

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

//수정 명령문 전달

String updateQuery = null;

PreparedStatement ps = null;

try {

//update query binding

switch(menu) {

case KOR:

updateQuery = "update studentprogram.studentdatatbl set kor=?, total=?, avr=?, grade=? where id=?";

ps = (PreparedStatement)con.prepareStatement(updateQuery);

ps.setInt(1, sd.getKor()); break;

case ENG:

updateQuery = "update studentprogram.studentdatatbl set eng=?, total=?, avr=?, grade=? where id=?";

ps = (PreparedStatement)con.prepareStatement(updateQuery);

ps.setInt(1, sd.getEng()); break;

case MATH:

updateQuery = "update studentprogram.studentdatatbl set math=?, total=?, avr=?, grade=? where id=?";

ps = (PreparedStatement)con.prepareStatement(updateQuery);

ps.setInt(1, sd.getMath()); break;

}

ps.setInt(2, sd.getTotal());

ps.setDouble(3, sd.getAvr());

ps.setString(4, String.valueOf(sd.getGrade()));

ps.setString(5, sd.getId());

//쿼리 실행 명령

result = ps.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close(); ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

//결과값 반환

return result;

}

//데이터 삭제

public static int dataDelete(String id) {

//데이터베이스 접속

Connection con = DBUtility.getConnection();

//데이터베이스 입력 반환값

int result = 0;

//데이터베이스 연결 확인

if(con==null) System.out.println("MySQL 연결에 실패하였습니다.");

//삭제 명령문 전달(학번)

String deleteQuery = "delete from studentprogram.studentdatatbl where id=?";

PreparedStatement ps = null;

try {

//delete query binding

ps = (PreparedStatement)con.prepareStatement(deleteQuery);

ps.setString(1, id);

//쿼리 실행 명령

result = ps.executeUpdate();

} catch (SQLException e) {

e.printStackTrace();

} finally {

try {

con.close(); ps.close();

} catch (SQLException e) {

e.printStackTrace();

}

}

//결과값 반환

return result;

}

}

1. StudentMain.java

**package** studentProgram;

**import** java.util.ArrayList;

**import** java.util.InputMismatchException;

**import** java.util.List;

**import** java.util.Scanner;

**import** java.util.regex.Matcher;

**import** java.util.regex.Pattern;

**public** **class** StudentMain {

//메뉴 : 성적 입력, 개별 성적 조회, 전체 성적 조회, 성적순 정렬, 성적 수정, 성적 삭제, 종료

**public** **static** **final** **int** ***INSERT***=1, ***SEARCH***=2, ***PRINT***=3, ***SORT***=4, ***UPDATE***=5, ***DELETE***=6, ***EXIT***=7;

//스캐너 생성

**public** **static** Scanner *scan* = **new** Scanner(System.***in***);

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

**int** menu = 0;

**boolean** flag = **false**;

//메뉴 선택 : 성적 입력, 개별 성적 조회, 전체 성적 조회, 성적순 정렬, 성적 수정, 성적 삭제, 종료

**while**(!flag) {

//메뉴 선택 함수 호출

menu = *selectMenu*();

**switch**(menu) {

**case** ***INSERT***: *studentInsert*(); **break**;

**case** ***SEARCH***: *studentSearch*(); **break**;

**case** ***PRINT***: *studentPrint*(); **break**;

**case** ***SORT***: *studentSort*(); **break**;

**case** ***UPDATE***: *studentUpdate*(); **break**;

**case** ***DELETE***: *studentDelete*(); **break**;

**case** ***EXIT***: System.***out***.println("성적 프로그램을 종료합니다."); flag = **true**; **break**;

}

}

}

//메뉴 선택

**private** **static** **int** selectMenu() {

**boolean** flag = **false**;

**int** menu = 0;

**while**(!flag) {

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println(" 1. 입력 2. 성적 조회 3. 전체 성적 조회 4. 정렬 5. 수정 6. 삭제 7. 종료");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.print("메뉴 선택 >> ");

**try** {

menu = Integer.*parseInt*(*scan*.nextLine());

} **catch**(InputMismatchException e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

} **catch**(Exception e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

}

**if**(menu>0 && menu<8) flag = **true**;

**else** System.***out***.println("1~7 사이의 숫자를 입력해주세요.");

}

**return** menu;

}

//성적 데이터 입력

**private** **static** **void** studentInsert() {

//멤버변수 : 학번, 이름, 성별, 국어, 영어, 수학, 총점, 평균, 등급

String id = **null**;

String name = **null**;

String gender = **null**;

**int** kor = 0;

**int** eng = 0;

**int** math = 0;

**int** total = 0;

**double** avr = 0.0;

**char** grade = '\u0000';

//학번 입력 함수 호출

**while**(**true**) {

System.***out***.print("ID를 입력해 주세요. : ");

id = *scan*.nextLine();

**if**(id.length()>=1 && id.length() <= 10) {**break**;}

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

//데이터 전체 조회

List<StudentModel> list = **new** ArrayList<StudentModel>();

list = DBController.*dataSelect*();

//id 중복 확인

**if**(list.size()>0) {

**for**(StudentModel data : list) {

**if**(id==data.getId()) {

System.***out***.println("이미 입력된 id입니다.");

**return**;

}

}

}

//이름 입력

**while**(**true**) {

System.***out***.print("이름을 입력해 주세요. : ");

name = *scan*.nextLine();

Pattern pattern = Pattern.*compile*("^[가-힣]\*$");

Matcher matcher = pattern.matcher(name);

**if**(matcher.matches()) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

//성별 입력

**while**(**true**) {

System.***out***.print("성별을 입력해 주세요.(남성/여성) : ");

gender = *scan*.nextLine();

**if**(gender.equals("남성")||gender.equals("여성")) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

//국어, 영어, 수학 성적 입력

**while**(**true**) {

System.***out***.print("국어점수를 입력해 주세요. : ");

kor = *scan*.nextInt();

**if**(kor >= 0 && kor <= 100) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

**while**(**true**) {

System.***out***.print("영어점수를 입력해 주세요. : ");

eng = *scan*.nextInt();

**if**(kor >= 0 && kor <= 100) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

**while**(**true**) {

System.***out***.print("수학점수를 입력해 주세요. : ");

math = *scan*.nextInt();

**if**(kor >= 0 && kor <= 100) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

//StudentData 객체 생성

StudentModel sd = **new** StudentModel(id, name, gender, kor, eng, math);

//총합, 평균, 등급 계산 함수 호출(StudentData)

total = sd.calTotal();

sd.setTotal(total);

avr = sd.calAvr();

sd.setAvr(avr);

grade = sd.calGrade(avr);

sd.setGrade(grade);

//데이터베이스에 저장

**int** result = DBController.*dataInsert*(sd);

**if**(result!=0) System.***out***.println(name+"님의 성적 입력이 완료되었습니다.");

**else** System.***out***.println(name+"님의 성적 입력에 실패하였습니다.");

}

//개별 성적 조회

**private** **static** **void** studentSearch() {

String name = **null**;

String searchData = **null**;

**while**(**true**) {

System.***out***.println("검색할 이름을 입력해 주세요.");

System.***out***.print(">> ");

name = *scan*.nextLine();

Pattern pattern = Pattern.*compile*("^[가-힣]\*$");

Matcher matcher = pattern.matcher(name);

**if**(matcher.matches()) **break**;

**else** System.***out***.println("잘못 입력하셨습니다. 다시 입력해주세요.");

}

searchData = name;

//데이터베이스 조회 : 개별 데이터

List<StudentModel> list = **new** ArrayList<StudentModel>();

list = DBController.*dataSearch*(searchData, 1);

//조회 결과 출력

**for**(StudentModel data : list) {

System.***out***.println(data);

}

}

//전체 성적 조회

**private** **static** **void** studentPrint() {

List<StudentModel> list = **new** ArrayList<StudentModel>();

//데이터베이스 조회 : 전체 데이터

list = DBController.*dataSelect*();

**if**(list.size()<=0) {

System.***out***.println("입력된 데이터가 없습니다.");

**return**;

}

//데이터 출력

*dataPrint*(list);

//전체 학생수, 전체 총점, 전체 평균 계산, 과목별 평균

**int** count = list.size();

**int** sum = 0;

**double** totalAvr = 0.0;

**double** korAvr = 0.0;

**double** engAvr = 0.0;

**double** mathAvr = 0.0;

**for**(StudentModel data : list) {

sum += data.getTotal();

totalAvr += data.getAvr();

korAvr += data.getKor();

engAvr += data.getEng();

mathAvr += data.getMath();

}

totalAvr /= (**double**)count;

korAvr /= (**double**)count;

engAvr /= (**double**)count;

mathAvr /= (**double**)count;

System.***out***.println("++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++");

System.***out***.println("전체 학생수 :"+count +"명 \t 전체 총점 : "+sum+"점 \t 전체 평균 : "+ String.*format*("%.2f", totalAvr)+"점");

System.***out***.println("국어 평균 : "+String.*format*("%.2f", korAvr)+"점 \t 영어 평균 : "+String.*format*("%.2f", engAvr) +"점 \t 수학 평균 : "+String.*format*("%.2f", mathAvr)+"점");

System.***out***.println("");

}

//성적 정렬 : 학번순, 이름순, 성적순

**private** **static** **void** studentSort() {

//조회 메뉴 선택

**int** menu = *sortMenu*();

**if**(menu==4) {

System.***out***.println("취소합니다.");

**return**;

}

//데이터베이스 조회 : 데이터 정렬

List<StudentModel> list = **new** ArrayList<StudentModel>();

list = DBController.*dataSort*(menu);

**if**(list.size()<=0) {

System.***out***.println("데이터가 없습니다.");

**return**;

}

//정렬된 성적 출력

*dataPrint*(list);

}

//성적 정렬 메뉴 선택 : 학번, 이름, 총점

**private** **static** **int** sortMenu() {

**boolean** flag = **false**;

**int** menu = 0;

**while**(!flag) {

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println(" 1.이름순 정렬 2. 성적순 정렬 3. 취소");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.print("메뉴 선택 >> ");

**try** {

menu = Integer.*parseInt*(*scan*.nextLine());

} **catch**(InputMismatchException e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

} **catch**(Exception e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

}

**if**(menu>0 && menu<5) flag = **true**;

**else** System.***out***.println("1~4 사이의 숫자를 입력해주세요.");

}

**return** menu;

}

//데이터 출력

**private** **static** **void** dataPrint(List<StudentModel> list) {

System.***out***.println("----------------------------------------------------------------------");

System.***out***.println("학번\t"+"이름\t"+"성별\t"+" 국어\t"+" 영어\t"+" 수학\t"+" 총점\t"+" 평균\t"+" 등급");

System.***out***.println("----------------------------------------------------------------------");

**for**(StudentModel data : list) System.***out***.println(data);

}

//성적 수정 : ID로 조회, 과목(국,영,수) 선택하여 수정

**private** **static** **void** studentUpdate() {

**final** **int** KOR=1, ENG=2, MATH=3, EXIT=4;

String name = **null**;

**int** menu = 0;

**int** kor = 0;

**int** eng = 0;

**int** math = 0;

**int** result = 0;

//학번 입력

System.***out***.print("수정할 이름을 입력하세요 : ");

name = *scan*.nextLine();

//수정 전 데이터 확인 : 학번으로 데이터 조회

List<StudentModel> list = **new** ArrayList<StudentModel>();

list = DBController.*dataSearch*(String.*valueOf*(name), 1);

**if**(list.size()<=0) {

System.***out***.println("입력된 데이터가 없습니다.");

**return**;

}

System.***out***.println("현재 성적은 다음과 같습니다.");

*dataPrint*(list);

//StudentData 객체로 저장

StudentModel sd = list.get(0);

//수정할 성적 선택

menu = *updateMenu*();

**switch**(menu) {

**case** KOR :

System.***out***.print("수정할 국어점수를 입력하세요 : ");

kor = *scan*.nextInt();

sd.setKor(kor); **break**;

**case** ENG :

System.***out***.print("수정할 영어점수를 입력하세요 : ");

eng = *scan*.nextInt();

sd.setEng(eng); **break**;

**case** MATH :

System.***out***.print("수정할 수학점수를 입력하세요 : ");

math = *scan*.nextInt();

sd.setMath(math); **break**;

**case** EXIT :

System.***out***.println("수정을 취소합니다.");

**return**;

}

//총점, 평균, 등급 계산

**int** total = sd.calTotal();

sd.setTotal(total);

**double** avr = sd.calAvr();

sd.setAvr(avr);

**char** grade = sd.calGrade(avr);

sd.setGrade(grade);

//데이터베이스 수정

result = DBController.*dataUpdate*(sd, menu);

**if**(result!=0) System.***out***.println("성적 수정이 완료되었습니다.");

**else** System.***out***.println("성적 수정에 실패하였습니다.");

}

//수정할 과목 선택

**private** **static** **int** updateMenu() {

**boolean** flag = **false**;

**int** menu = 0;

**while**(!flag) {

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.println(" 1. 국어수정 2. 영어수정 3. 수학수정 4. 취소");

System.***out***.println("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*");

System.***out***.print("메뉴 선택 >> ");

**try** {

menu = Integer.*parseInt*(*scan*.nextLine());

} **catch**(InputMismatchException e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

} **catch**(Exception e) {

System.***out***.println("숫자를 입력해주세요.");

**continue**;

}

**if**(menu>0 && menu<5) flag = **true**;

**else** System.***out***.println("1~4 사이의 숫자를 입력해주세요.");

}

**return** menu;

}

//성적 삭제 : ID로 검색

**private** **static** **void** studentDelete() {

String id = **null**;

**int** result = 0;

//학번 입력

System.***out***.print("삭제할 ID를 입력하세요 : ");

id = *scan*.nextLine();

//데이터베이스 삭제

result = DBController.*dataDelete*(id);

**if**(result!=0) System.***out***.println("성적 삭제가 완료되었습니다.");

**else** System.***out***.println("성적 삭제에 실패하였습니다.");

}

}

1. StudentModel.java

**package** studentProgram;

**import** java.util.Objects;

**public** **class** StudentModel **implements** Comparable<Object> {

**public** **static** **final** **int** ***SU***;

**public** **static** **int** *totalCount*;

**public** **static** **int** *totalSum*;

**public** **static** **double** *totalAvr*;

**private** String id;

**private** String name;

**private** String gender;

**private** **int** kor;

**private** **int** eng;

**private** **int** math;

**private** **int** total;

**private** **double** avr;

**private** **char** grade;

**static** {

*totalCount* = 0;

*totalSum* = 0;

*totalAvr* = 0;

***SU*** = 3;

}

**public** StudentModel() {

**this**(**null**, **null**, **null**, 0, 0, 0);

}

**public** StudentModel(String id, String name, String gender, **int** kor, **int** eng, **int** math) {

**super**();

**this**.id = id;

**this**.name = name;

**this**.gender = gender;

**this**.kor = kor;

**this**.eng = eng;

**this**.math = math;

**this**.total = 0;

**this**.avr = 0.0;

**this**.grade = '\u0000';

}

**public** StudentModel(String id, String name, String gender, **int** kor, **int** eng, **int** math, **int** total, **double** avr,

**char** grade) {

**this**(id, name, gender, kor, eng, math);

**this**.total = total;

**this**.avr = avr;

**this**.grade = grade;

}

**public** **int** calTotal() {

**return** **this**.total = **this**.kor + **this**.eng + **this**.math;

}

**public** **double** calAvr() {

**return** **this**.avr = **this**.total / (**double**)***SU***;

}

**public** **char** calGrade(**double** avr) {

**if**(**this**.avr>= 90.0) {

**return** **this**.grade = 'A';

}

**else** **if**(**this**.avr>= 80.0) {

**return** **this**.grade = 'B';

}

**else** **if**(**this**.avr>= 70.0) {

**return** **this**.grade = 'C';

}

**else** **if**(**this**.avr>= 60.0) {

**return** **this**.grade = 'D';

}

**else** {

**return** **this**.grade = 'F';

}

}

@Override

**public** **int** hashCode() {

**return** Objects.*hash*(id, name);

}

@Override

**public** **boolean** equals(Object obj) {

**if**(!(obj **instanceof** StudentModel))

**throw** **new** IllegalArgumentException("No Model in this class");

StudentModel sm = (StudentModel)obj;

**return** **this**.id.equals(sm.id);

}

@Override

**public** String toString() {

**return** "[" + name + " \t " + gender + " \t " + id + " \t " + kor + " \t " + math

+ " \t " + eng + " \t " + total + " \t " + String.*format*("%.2f", avr) + " \t " + grade + "]";

}

@Override

**public** **int** compareTo(Object obj) {

**if**(!(obj **instanceof** StudentModel))

**throw** **new** IllegalArgumentException("No Model in this class");

StudentModel sm = (StudentModel)obj;

**if**(**this**.total > sm.total) {

**return** 1;

}

**if**(**this**.total < sm.total) {

**return** -1;

}

**else** **return** 0;

}

**public** String getName() {

**return** name;

}

**public** **void** setName(String name) {

**this**.name = name;

}

**public** String getGender() {

**return** gender;

}

**public** **void** setGender(String gender) {

**this**.gender = gender;

}

**public** String getId() {

**return** id;

}

**public** **void** setId(String id) {

**this**.id = id;

}

**public** **int** getKor() {

**return** kor;

}

**public** **void** setKor(**int** kor) {

**this**.kor = kor;

}

**public** **int** getMath() {

**return** math;

}

**public** **void** setMath(**int** math) {

**this**.math = math;

}

**public** **int** getEng() {

**return** eng;

}

**public** **void** setEng(**int** eng) {

**this**.eng = eng;

}

**public** **int** getTotal() {

**return** total;

}

**public** **void** setTotal(**int** total) {

**this**.total = total;

}

**public** **double** getAvr() {

**return** avr;

}

**public** **void** setAvr(**double** avr) {

**this**.avr = avr;

}

**public** **char** getGrade() {

**return** grade;

}

**public** **void** setGrade(**char** grade) {

**this**.grade = grade;

}

}