|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **작업 수행 보고서**   |  |  |  |  |  | | --- | --- | --- | --- | --- | | 교과목 | 데이터베이스 활용 및 구현 | 학생번호 : 2750 성명: 송훈일 | | | | 능력단위 | SQL 활용(2001020413\_16v3) | | 평가영역 | 1.1, 1.2, 1.3, 1.4  2.1, 2.2, 2.3 | | 능력단위요소 | 1. 기본 SQL 작성하기(2001020413\_16v3.1) | | | 2. 고급 SQL 작성하기(2001020413\_16v3.2) | | | 평가방법 | 문제해결시나리오 | |  |  | | --- | | **Subject : 모대학교 학사관리 콘텐츠 테이블 생성 및 DAO 클래스 생성 구현**  모대학교 학사관리 컨텐츠의 데이터베이스 오브젝트 생성을 위해 다음과 같은 요구사항이 있다.  1. 개인별 프로젝트 컨텐츠 제목 선정  - 데이터베이스를 사용할 수 있는 컨텐츠 선정한다.  2. 개인별 프로젝트 컨텐츠의 요구사항  - 데이터베이스 구축에 필요한 요구사항을 설정한다.  3. 요구사항에 필요한 데이터베이스 모델링 설계  - 논리적 모델링과 물리적 모델링을 ERD로 설계한다.  4. 물리적 모델링을 통해 SQL 쿼리문을 작성한다.  - DDL 문으로 작성한다.  5. 생성된 테이블에 데이터를 삽입, 수정, 삭제하고 행을 조회하는 DML(Data Manipulation Language) 명령문을 작성  - DML문을 작성한다.  - 테이블 조인과 서브 쿼리를 사용한다. |  |  | | --- | | **테이블 모델링** | | EMB00002928588f  **drop schema if exists `mirae\_institution\_db`;**  **create schema if not exists `mirae\_institution\_db`;**  **use `mirae\_institution\_db`;**  **drop table if exists `professor\_tbl`;**  **create table if not exists `professor\_tbl` (**  **`id` varchar(15) unique not null,**  **`name` varchar(10) not null,**  **`phone` varchar(11) null,**  **constraint `pk\_id` primary key(`id`)**  **) engine=innodb default charset=utf8;**  **drop table if exists `course\_tbl`;**  **create table if not exists `course\_tbl` (**  **`id` int auto\_increment not null,**  **`professor\_id` varchar(15) unique not null,**  **`major` varchar(10) not null,**  **constraint `pk\_id` primary key(`id`),**  **constraint `fk\_professor\_id` foreign key(`professor\_id`) references `professor\_tbl`(`id`) on delete restrict on update cascade**  **) engine=innodb default charset=utf8;**  **drop table if exists `student\_tbl`;**  **CREATE TABLE if not exists `student\_tbl` (**  **`id` varchar(10) NOT NULL,**  **`name` varchar(10) NOT NULL,**  **`course\_id` int not null,**  **`birthdate` varchar(10) NOT NULL,**  **`gender` varchar(1) NOT NULL,**  **`email` varchar(40) NOT NULL,**  **`phone` varchar(20) NOT NULL,**  **`c` int NOT NULL,**  **`java` int NOT NULL,**  **`android` int NOT NULL,**  **`web` int not null,**  **`total` int null,**  **`avg` double null,**  **`grade` varchar(1) null,**  **constraint `pk\_id` PRIMARY KEY (`id`),**  **constraint `fk\_course\_id` foreign key(`course\_id`) references `course\_tbl`(`id`) on delete cascade on update cascade**  **) ENGINE=InnoDB DEFAULT CHARSET=utf8;**  **프로시저 프로그램 문장 구현 내용**  **drop procedure if exists `proc\_insert\_then\_calc`;**  **delimiter //**  **create procedure `proc\_insert\_then\_calc`(in `id` varchar(10), in `name` varchar(10), in `course\_id` int, in `birthdate` varchar(10), in `gender` varchar(1), in `email` varchar(40), in `phone` varchar(20),**  **in `c` int, in `java` int, in `android` int, in `web` int)**  **begin**  **declare total int;**  **declare avg double;**  **declare grade varchar(1);**    **set total = `fun\_calcTotal`(`c`, `java`, `android`, `web`);**  **set avg = total / 4;**  **set grade = `fun\_calcGrade`(avg);**  **insert into `student\_tbl` values(`id`, `name`, `course\_id`, `birthdate`, `gender`, `email`, `phone`, `c`, `java`, `android`, `web`, total, avg, grade);**  **end //**  **delimiter ;**  **call `proc\_insert\_then\_calc`('feean2468', '송훈일', 1, '19920329', '남', 'freean2468@gmail.com', '01079978395', 100, 100, 100, 100);**  **call `proc\_insert\_then\_calc`('test1', '학생1', 1, '19950329', '여', 'fran2468@gmail.com', '01079732945', 80, 10, 20, 80);**  **call `proc\_insert\_then\_calc`('test2', '학생2', 1, '19980329', '남', 'freean8@gmail.com', '01067973095', 60, 60, 60, 30);**  **펑션 프로그램 문장 구현 내용**  **drop function if exists `fun\_calcTotal`;**  **delimiter //**  **create function `fun\_calcTotal`(`c` int, `java` int, `android` int, `web` int) returns int**  **begin**  **return `c` + `java` + `android` + `web`;**  **end //**  **delimiter ;**  **drop function if exists `fun\_calcGrade`;**  **delimiter //**  **create function `fun\_calcGrade`(`avg` double) returns varchar(1)**  **begin**  **declare grade varchar(1);**  **if (`avg` >= 90.0) then set grade = 'A';**  **elseif (`avg` >= 80.0) then set grade = 'B';**  **elseif (`avg` >= 70.0) then set grade = 'C';**  **elseif (`avg` >= 60.0) then set grade = 'D';**  **else set grade = 'F';**  **end if;**  **return grade;**  **end //**  **delimiter ;**  **인덱스 처리문장 구현 내용**  **create index `idx\_professor\_id` on `professor\_tbl`(`id`);**  **create index `idx\_course\_id` on `course\_tbl`(`id`);**  **create index `idx\_student\_id` on `student\_tbl`(`id`);** |  |  | | --- | | **프로젝트 계층도** | | EMB000029285890  EMB000029285891 |  |  | | --- | | **자바 클래스 구현** | | **- Main 클래스**  /\*\*  \*  \* @author 송훈일 (Neil Son)  \* @version 0.1  \* 개발 시작 일자 : 2021\_02\_3  \* last\_updated : 2021\_02\_19  \* 목적 : MySQA과 java 연동 후 JavaFX로 출력  \* GitHub : https://github.com/freean2468/java\_fx\_mysql  \* 실행 영상 :  \*  \*/  package application;    import javafx.application.Application;  import javafx.fxml.FXMLLoader;  import javafx.scene.Parent;  import javafx.scene.Scene;  import javafx.stage.Stage;  public class Main extends Application {  public static RootController rootController = null;    @Override  public void init() {  System.out.println(Thread.currentThread().getName() + ": init()");  }    @Override  public void start(Stage primaryStage) {  try {  FXMLLoader fxmlLoader = new FXMLLoader(getClass().getResource("../resources/root.fxml"));  Parent root = fxmlLoader.load();  rootController = (RootController) fxmlLoader.getController();  Scene scene = new Scene(root);  scene.getStylesheets().add(getClass().getResource("../resources/application.css").toExternalForm());  primaryStage.setScene(scene);  primaryStage.setTitle("JavaFX-Java-MySQL project");  primaryStage.show();  } catch(Exception e) {  e.printStackTrace();  }  }    @Override  public void stop() {  System.out.println(Thread.currentThread().getName() + ": stop()");  }    public static void main(String[] args) {  launch(args);  }  }  **- RootController 클래스**  package application;  import java.io.IOException;  import java.net.URL;  import java.util.ArrayList;  import java.util.HashSet;  import java.util.Iterator;  import java.util.ResourceBundle;  import data.Person;  import data.ScoreTable;  import data.Student;  import javafx.collections.FXCollections;  import javafx.collections.ObservableList;  import javafx.fxml.FXML;  import javafx.fxml.FXMLLoader;  import javafx.fxml.Initializable;  import javafx.scene.Parent;  import javafx.scene.Scene;  import javafx.scene.control.Button;  import javafx.scene.control.ComboBox;  import javafx.scene.control.Label;  import javafx.scene.control.TableColumn;  import javafx.scene.control.TableView;  import javafx.scene.control.TextField;  import javafx.scene.control.cell.PropertyValueFactory;  import javafx.stage.Modality;  import javafx.stage.Stage;  import javafx.stage.StageStyle;  import model.StudentVO;  import mysql.MySQLQueries;  /\*\*  \* @apiNote JavaFX UI 컨트롤을 담당  \* 싱글톤으로 선언될 수 없다.  \*  \* last\_updated : 2021\_02\_19  \*  \*/  public class RootController implements Initializable {  /\*\*  \* `mirae\_institution\_db`.`student\_tbl` 테이블 필드 상수 선언  \*/  public static final int ID\_COLUMN = 0;  public static final int NAME\_COLUMN = 1;  public static final int COURSE\_ID\_COLUMN = 2;  public static final int BIRTHDATE\_COLUMN = 3;  public static final int GENDER\_COLUMN = 4;  public static final int EMAIL\_COLUMN = 5;  public static final int PHONE\_COLUMN = 6;  public static final int C\_COLUMN = 7;  public static final int JAVA\_COLUMN = 8;  public static final int ANDROID\_COLUMN = 9;  public static final int WEB\_COLUMN = 10;  public static final int TOTAL\_COLUMN = 11;  public static final int AVG\_COLUMN = 12;  public static final int GRADE\_COLUMN = 13;    public static final int INITIAL\_NUM = 5;    // DB transaction 결과물을 담을 Set  private HashSet<Student> set = new HashSet<>();    @FXML private TableView<StudentVO> tableView;    /\*\*  \* @apiNote FX UI를 담당하는 JavaFX Application Thread가 호출하는 초기화 함수  \*/  @Override public void initialize(URL location, ResourceBundle resources) {  System.out.println(Thread.currentThread().getName() + "'s calling initialize");    if (MySQLQueries.createTable("student\_tbl")) {  //  // 테이블이 비어있다면 자동으로 생성해주는 부분.  //  // while (set.size() < INITIAL\_NUM) {  // String id = Person.generateId();  //  // set.add(new Student(  // Person.generateBirthdate(),  // Person.generateName(),  // id,  // Person.generateGender(),  // Person.generateEmail(id),  // Person.generatePhone(),  // new ScoreTable(ScoreTable.generateRandomScore(),  // ScoreTable.generateRandomScore(), ScoreTable.generateRandomScore())  // ));  // }  //  // System.out.println("insert into student");  //  // Iterator<Student> itr = set.iterator();  // while (itr.hasNext()) {  // MySQLQueries.insertStudent(itr.next());  // }  } else {  //  // 테이블이 비어있지 않다면 그냥 모두 가져온다  //  System.out.println("select \* from student\_tbl");  MySQLQueries.selectStudent(set);  }    this.refreshTable();    this.studentTblBinding();  }    @FXML public void handleInsert() {  boolean flag = true;  Stage dialog = new Stage(StageStyle.UTILITY);  dialog.initModality(Modality.WINDOW\_MODAL);  dialog.initOwner(tableView.getScene().getWindow());  dialog.setTitle("Insert");    try {  Parent parent = FXMLLoader.load(getClass().getResource("../resources/form.fxml"));  Button insertConfirmButton = (Button)parent.lookup("#confirm");    insertConfirmButton.setOnAction(e->{  try {  TextField id = (TextField)parent.lookup("#id");  TextField courseId = (TextField)parent.lookup("#courseId");  TextField name = (TextField)parent.lookup("#name");  TextField birthdate = (TextField)parent.lookup("#birthdate");  @SuppressWarnings("unchecked")  ComboBox<String> email = (ComboBox<String>)parent.lookup("#email");  TextField phone = (TextField)parent.lookup("#phone");  @SuppressWarnings("unchecked")  ComboBox<String> gender = (ComboBox<String>)parent.lookup("#gender");  TextField c = (TextField)parent.lookup("#c");  TextField java = (TextField)parent.lookup("#java");  TextField android = (TextField)parent.lookup("#android");  TextField web = (TextField)parent.lookup("#web");    Student student = new Student(  id.getText(),  name.getText(),  Integer.parseInt(courseId.getText()),  birthdate.getText(),  gender.getValue(),  id.getText()+email.getValue(),  phone.getText(),  new ScoreTable(Integer.parseInt(c.getText()),  Integer.parseInt(java.getText()),  Integer.parseInt(android.getText()),  Integer.parseInt(web.getText()))  );    boolean res = addIntoSet(student);  if (res == false) {  showMessage(dialog, "중복입니다.");  } else {  dialog.close();  }  } catch (NumberFormatException nfe) {  showMessage(dialog, "올바르지 않은 입력");  }  });    Button insertCancelButton = (Button)parent.lookup("#cancel");  insertCancelButton.setOnAction(e->dialog.close());    Scene scene = new Scene(parent);  dialog.setScene(scene);  dialog.setResizable(false);  dialog.show();  } catch (IOException e) {  e.printStackTrace();  }  }    @FXML public void handleUpdate() {  boolean flag = true;  Stage dialog = new Stage(StageStyle.UTILITY);  dialog.initModality(Modality.WINDOW\_MODAL);  dialog.initOwner(tableView.getScene().getWindow());  dialog.setTitle("Update");  StudentVO m = tableView.getSelectionModel().getSelectedItem();  try {  Parent parent = FXMLLoader.load(getClass().getResource("../resources/form.fxml"));  Button confirm = (Button)parent.lookup("#confirm");  TextField id = (TextField)parent.lookup("#id");  TextField courseId = (TextField)parent.lookup("#courseId");  TextField name = (TextField)parent.lookup("#name");  TextField birthdate = (TextField)parent.lookup("#birthdate");  @SuppressWarnings("unchecked")  ComboBox<String> email = (ComboBox<String>)parent.lookup("#email");  TextField phone = (TextField)parent.lookup("#phone");  @SuppressWarnings("unchecked")  ComboBox<String> gender = (ComboBox<String>)parent.lookup("#gender");  TextField c = (TextField)parent.lookup("#c");  TextField java = (TextField)parent.lookup("#java");  TextField android = (TextField)parent.lookup("#android");  TextField web = (TextField)parent.lookup("#web");    id.setText(m.getId());  id.setDisable(true);  courseId.setText(String.valueOf(m.getCourseId()));    name.setText(m.getName());  birthdate.setText(m.getBirthDate());  email.setValue("@"+m.getEmail().split("@")[1]);  phone.setText(m.getPhone());  gender.setValue(m.getGender());  c.setText(String.valueOf(m.getC()));  java.setText(String.valueOf(m.getJava()));  android.setText(String.valueOf(m.getAndroid()));  web.setText(String.valueOf(m.getWeb()));    confirm.setOnAction(e->{  try {  Student student = new Student(  id.getText(),  name.getText(),  Integer.parseInt(courseId.getText()),  birthdate.getText(),  gender.getValue(),  id.getText()+email.getValue(),  phone.getText(),  new ScoreTable(Integer.parseInt(c.getText()),  Integer.parseInt(java.getText()),  Integer.parseInt(android.getText()),  Integer.parseInt(web.getText()))  );    boolean res = updateSetWhereId(student);  if (res == false) {  showMessage(dialog, "update 오류");  } else {  dialog.close();  }  } catch (NumberFormatException nfe) {  showMessage(dialog, "올바르지 않은 입력");  }  });    Button cancel = (Button)parent.lookup("#cancel");  cancel.setOnAction(e->dialog.close());    Scene scene = new Scene(parent);  dialog.setScene(scene);  dialog.setResizable(false);  dialog.show();  } catch (IOException e) {  e.printStackTrace();  } catch (NullPointerException npe) {  this.showMessage(dialog, "수정하고자 하는 학생 선택");  }  }    @FXML public void handleSearch() {  boolean flag = true;  Stage dialog = new Stage(StageStyle.UTILITY);  dialog.initModality(Modality.WINDOW\_MODAL);  dialog.initOwner(tableView.getScene().getWindow());  dialog.setTitle("Search");    try {  Parent parent = FXMLLoader.load(getClass().getResource("../resources/form.fxml"));  Button insertConfirmButton = (Button)parent.lookup("#confirm");    insertConfirmButton.setOnAction(e->{  TextField id = (TextField)parent.lookup("#id");  TextField name = (TextField)parent.lookup("#name");  TextField courseId = (TextField)parent.lookup("#courseId");  TextField birthdate = (TextField)parent.lookup("#birthdate");  @SuppressWarnings("unchecked")  ComboBox<String> email = (ComboBox<String>)parent.lookup("#email");  TextField phone = (TextField)parent.lookup("#phone");  @SuppressWarnings("unchecked")  ComboBox<String> gender = (ComboBox<String>)parent.lookup("#gender");  TextField c = (TextField)parent.lookup("#c");  c.setDisable(true);  TextField java = (TextField)parent.lookup("#java");  java.setDisable(true);  TextField android = (TextField)parent.lookup("#android");  android.setDisable(true);  TextField web = (TextField)parent.lookup("#web");  web.setDisable(true);    // 아무 내용도 입력하지 않으면 select \*  if (id.getText().equals("") &&  courseId.getText().equals("") &&  name.getText().equals("") &&  birthdate.getText().equals("") &&  email.getValue() == null &&  phone.getText().equals("") &&  gender.getValue() == null) {  MySQLQueries.selectStudent(set);  this.refreshTable();  dialog.close();  return;  }    int nCourseId = 0;    try {  nCourseId = Integer.parseInt(courseId.getText());  } catch (NumberFormatException nfe) {  nCourseId = 0;  }    boolean res = updateAfterSelect(id.getText(), nCourseId,  name.getText(), birthdate.getText(), email.getValue(), phone.getText(),  gender.getValue());  if (res == false) {  showMessage(dialog, "검색 결과 없음.");  } else {  dialog.close();  }  });    Button insertCancelButton = (Button)parent.lookup("#cancel");  insertCancelButton.setOnAction(e->dialog.close());    Scene scene = new Scene(parent);  dialog.setScene(scene);  dialog.setResizable(false);  dialog.show();  } catch (IOException e) {  e.printStackTrace();  }  }    @FXML public void handleSort() {  ArrayList<Student> list = new ArrayList<>();  MySQLQueries.selectStudentOrderById(list);  ObservableList<StudentVO> tableList = FXCollections.observableArrayList();    for (Student s : list) {  tableList.add(new StudentVO(s));  }    tableView.setItems(tableList);  tableView.refresh();  }    @FXML public void handleDelete() {  boolean flag = true;  Stage dialog = new Stage(StageStyle.UTILITY);  dialog.initModality(Modality.WINDOW\_MODAL);  dialog.initOwner(tableView.getScene().getWindow());  dialog.setTitle("Delete");  StudentVO m = tableView.getSelectionModel().getSelectedItem();  try {  Parent parent = FXMLLoader.load(getClass().getResource("../resources/form.fxml"));  Button insertConfirmButton = (Button)parent.lookup("#confirm");  TextField id = (TextField)parent.lookup("#id");  TextField name = (TextField)parent.lookup("#name");  TextField courseId = (TextField)parent.lookup("#courseId");  TextField birthdate = (TextField)parent.lookup("#birthdate");  @SuppressWarnings("unchecked")  ComboBox<String> email = (ComboBox<String>)parent.lookup("#email");  TextField phone = (TextField)parent.lookup("#phone");  @SuppressWarnings("unchecked")  ComboBox<String> gender = (ComboBox<String>)parent.lookup("#gender");  TextField c = (TextField)parent.lookup("#c");  TextField java = (TextField)parent.lookup("#java");  TextField android = (TextField)parent.lookup("#android");  TextField web = (TextField)parent.lookup("#web");    id.setText(m.getId()); id.setDisable(true);  name.setText(m.getName()); name.setDisable(true);  courseId.setText(String.valueOf(m.getCourseId())); courseId.setDisable(true);  birthdate.setText(m.getBirthDate()); birthdate.setDisable(true);  email.setValue("@"+m.getEmail().split("@")[1]); email.setDisable(true);  phone.setText(m.getPhone()); phone.setDisable(true);  gender.setValue(m.getGender()); gender.setDisable(true);  c.setText(String.valueOf(m.getC())); c.setDisable(true);  java.setText(String.valueOf(m.getJava())); java.setDisable(true);  android.setText(String.valueOf(m.getAndroid())); android.setDisable(true);  web.setText(String.valueOf(m.getWeb())); web.setDisable(true);    insertConfirmButton.setOnAction(e->{  try {  boolean res = removeSetWhereId(id.getText());  if (res == false) {  showMessage(dialog, "delete 오류");  } else {  dialog.close();  }  } catch (NumberFormatException nfe) {  showMessage(dialog, "올바르지 않은 입력");  }  });    Button insertCancelButton = (Button)parent.lookup("#cancel");  insertCancelButton.setOnAction(e->dialog.close());    Scene scene = new Scene(parent);  dialog.setScene(scene);  dialog.setResizable(false);  dialog.show();  } catch (IOException e) {  e.printStackTrace();  } catch (NullPointerException npe) {  this.showMessage(dialog, "삭제하고자 하는 학생 선택");  }  }    @FXML public void handleQuit() {  System.out.println("Quit");  Stage stage = (Stage)tableView.getScene().getWindow();  stage.close();  }    /\*\*  \* @apiNote 상황에 따른 메세지를 팝업창으로 전달  \* @param Stage : 현재 메인 화면  \* @param message : 전달할 메세지  \*/  public void showMessage(Stage dialog, String message) {  Stage messageDialog = new Stage(StageStyle.UTILITY);  messageDialog.initModality(Modality.WINDOW\_MODAL);  messageDialog.initOwner(dialog);  messageDialog.setTitle("메세지");    try {  Parent messageParent = FXMLLoader.load(getClass().getResource("../resources/messageDialog.fxml"));  Label messageLabel = (Label)messageParent.lookup("#message");  messageLabel.setText(message);  Button messageButton = (Button)messageParent.lookup("#confirm");  messageButton.setOnAction(messageEvent->messageDialog.close());  Scene scene = new Scene(messageParent);  messageDialog.setScene(scene);  messageDialog.setResizable(false);  messageDialog.show();  } catch (IOException messageException) {  messageException.printStackTrace();  }  }    /\*\*  \* @apiNote DB transaction 결과물을 가지고 있는 Set은 그 데이터가 수정될 때  다음과 같은 작업을 병행해줘야 하기 때문에 따로 함수들을 만듬.  1. DB transaction  2. set data 반영  3. UI data 모델 반영  \* @param s : set에 넣으려는 학생 객체  \* @return 정상적으로 넣었으면 true, 이미 존재하면 false 반환  \*/  public boolean addIntoSet(Student s) {  boolean res = set.add(s);  if (res == true) {  ObservableList<StudentVO> tableList = tableView.getItems();  tableList.add(new StudentVO(s));  tableView.setItems(tableList);  MySQLQueries.insertStudent(s);  }  return res;  }    /\*\*  \* @apiNote student의 id를 기준으로 set에서 삭제, db에서도 삭제, ui에서도 삭제  \* @param id 삭제하려는 student의 id  \* @return 삭제했으면 true 실패면 false  \*/  public boolean removeSetWhereId(String id) {  boolean res = false;  Iterator<Student> itr = set.iterator();  while(itr.hasNext()) {  Student item = itr.next();  if (id.equals(item.getId())) {  itr.remove();  MySQLQueries.deleteStudentById(id);  res = true;  ObservableList<StudentVO> tableList = tableView.getItems();  Iterator<StudentVO> mItr = tableList.iterator();    while (mItr.hasNext()) {  StudentVO m = mItr.next();  if (id.equals(m.getId())) {  mItr.remove();  break;  }  }  tableView.setItems(tableList);  tableView.refresh();  break;  }  }  return res;  }    /\*\*  \* @apiNote 새로운 Student 객체를 받아 set 안 기존에 있던 객체에 덮어 씌운다.  \* 마찬가지로 DB와 UI에도 반영  \* @param s 새로운 Student 객체  \* @return 성공 true 실패 false  \*/  public boolean updateSetWhereId(Student s) {  boolean res = false;  Iterator<Student> itr = set.iterator();  while(itr.hasNext()) {  Student item = itr.next();  if (s.getId().equals(item.getId())) {  item.setBirthdate(s.getBirthdate());  item.setCourseId(s.getCourseId());  item.setEmail(s.getEmail());  item.setName(s.getName());  item.setGender(s.getGender());  item.setPhone(s.getPhone());  item.setScoreTable(s.getScoreTable());  MySQLQueries.updateStudentWhereId(s);  res = true;  ObservableList<StudentVO> tableList = tableView.getItems();  Iterator<StudentVO> mItr = tableList.iterator();    while (mItr.hasNext()) {  StudentVO m = mItr.next();  if (s.getId().equals(m.getId())) {  m.setBirthDate(s.getBirthdate());  m.setEmail(s.getEmail());  m.setGender(s.getGender());  m.setGrade(s.getGender());  m.setName(s.getName());  m.setPhone(s.getPhone());  m.setC(s.getScoreTable().getC());  m.setJava(s.getScoreTable().getJava());  m.setAndroid(s.getScoreTable().getAndroid());  m.setWeb(s.getScoreTable().getWeb());  m.setTotal(s.getScoreTable().getTotal());  m.setAvg(s.getScoreTable().getAvg());  }  }  tableView.setItems(tableList);  tableView.refresh();  break;  }  }  return res;  }    /\*\*  \* @apiNote 필드 중 아무 값이나 선택해 입력하면 해당 필드를 가진 튜플을 검색해주는 함수  \* set, DB, UI 모두 반영  \* @param id  \* @param courseId  \* @param name  \* @param birthdate  \* @param email  \* @param phone  \* @param gender  \* @return  \*/  public boolean updateAfterSelect(String id, int courseId, String name, String birthdate,  String email, String phone, String gender) {  boolean res = false;    res = MySQLQueries.selectStudent(set, id, courseId, name, birthdate, gender, email, phone);    Iterator<Student> itr = set.iterator();  ObservableList<StudentVO> tableList = tableView.getItems();  tableList.clear();    while(itr.hasNext()) {  Student item = itr.next();  tableList.add(new StudentVO(item));  }  tableView.setItems(tableList);  tableView.refresh();  return res;  }    /\*\*  \* @apiNote  \* 쓸데없이 시간을 많이 잡아먹게 만든 부분,  \* 당연히 합리적인 논리의 전개로 tableColumn 데이터가 바뀌면 바로바로 tableView에 반영이 될 줄 알았는데 그렇지가 않아서  \* 직접 refresh()를 호출해줘야 한다;  \*/  public void refreshTable() {  ObservableList<StudentVO> tableList = FXCollections.observableArrayList();    Iterator<Student> itr = set.iterator();  while (itr.hasNext()) {  tableList.add(new StudentVO(itr.next()));  }    tableView.setItems(tableList);  tableView.refresh();  }    /\*\*  \* @apiNote db student\_tbl <-> java VO binding  \* 각 테이블 필드를 Factory 함수를 통해 생성되도록 각 Student field와 바인딩.  \* 이를 위해 Student 클래스를 모델링한 StudentVO 클래스를 만듬.  \*/  private void studentTblBinding() {  @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcId = (TableColumn<StudentVO, String>) tableView.getColumns().get(ID\_COLUMN);  tcId.setCellValueFactory(new PropertyValueFactory<>("id"));  tcId.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcBirthDate = (TableColumn<StudentVO, String>) tableView.getColumns().get(BIRTHDATE\_COLUMN);  tcBirthDate.setCellValueFactory(new PropertyValueFactory<>("birthDate"));  tcBirthDate.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcName = (TableColumn<StudentVO, String>)tableView.getColumns().get(NAME\_COLUMN);  tcName.setCellValueFactory(new PropertyValueFactory<>("name"));  tcName.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcCourseId = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(COURSE\_ID\_COLUMN);  tcCourseId.setCellValueFactory(new PropertyValueFactory<>("courseId"));  tcCourseId.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcGender = (TableColumn<StudentVO, String>)tableView.getColumns().get(GENDER\_COLUMN);  tcGender.setCellValueFactory(new PropertyValueFactory<>("gender"));  tcGender.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcEmail = (TableColumn<StudentVO, String>)tableView.getColumns().get(EMAIL\_COLUMN);  tcEmail.setCellValueFactory(new PropertyValueFactory<>("email"));  tcEmail.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcPhone = (TableColumn<StudentVO, String>)tableView.getColumns().get(PHONE\_COLUMN);  tcPhone.setCellValueFactory(new PropertyValueFactory<>("phone"));  tcPhone.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcC = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(C\_COLUMN);  tcC.setCellValueFactory(new PropertyValueFactory<>("c"));  tcC.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcJava = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(JAVA\_COLUMN);  tcJava.setCellValueFactory(new PropertyValueFactory<>("java"));  tcJava.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcAndroid = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(ANDROID\_COLUMN);  tcAndroid.setCellValueFactory(new PropertyValueFactory<>("android"));  tcAndroid.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcWeb = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(WEB\_COLUMN);  tcWeb.setCellValueFactory(new PropertyValueFactory<>("web"));  tcWeb.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Integer> tcTotal = (TableColumn<StudentVO, Integer>)tableView.getColumns().get(TOTAL\_COLUMN);  tcTotal.setCellValueFactory(new PropertyValueFactory<>("total"));  tcTotal.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, Double> tcAvg = (TableColumn<StudentVO, Double>)tableView.getColumns().get(AVG\_COLUMN);  tcAvg.setCellValueFactory(new PropertyValueFactory<>("avg"));  tcAvg.setStyle("-fx-alignment: CENTER;");    @SuppressWarnings("unchecked")  TableColumn<StudentVO, String> tcGrade = (TableColumn<StudentVO, String>)tableView.getColumns().get(GRADE\_COLUMN);  tcGrade.setCellValueFactory(new PropertyValueFactory<>("grade"));  tcGrade.setStyle("-fx-alignment: CENTER;");  }  }  **- Person 클래스**  package data;  public class Person implements Printable {  public static final int ***MAX\_ID\_LENGTH*** = 10;  public static final int ***MIN\_ID\_LENGTH*** = 3;  public static final int ***BIRTHDATE\_LENGTH*** = 8;  public static final int ***MAX\_NAME\_LENGTH*** = 4;  public static final int ***PHONE\_LENGTH*** = 13;    private String birthdate;  private String name;  private String id;  private String gender;  private String email;  private String phone;    public Person() {  this(null, null, null, null, null, null);  }    public Person(String id, String name, String birthdate, String gender, String email, String phone) {  this.birthdate = birthdate;  this.name = name;  this.id = id;  this.gender = gender;  this.email = email;  this.phone = phone;  }    public String getId() { return id; }    public String getBirthdate() { return birthdate; }  public void setBirthdate(String birthdate) { this.birthdate = birthdate; }  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 getEmail() { return email; }  public void setEmail(String email) { this.email = email; }  public String getPhone() { return phone; }  public void setPhone(String phone) { this.phone = phone; }    *@Override*  public String toString() {  return "birthdate:" + birthdate + ", name:" + name + ", id:" + id  + ", gender:" + gender + ", email:" + email + ", phone:" + phone;  }    *@Override*  public void printMember() {  System.***out***.println(this.toString());  }    public static String generateBirthdate() {  String year = String.*valueOf*((int)(Math.*random*()\*20) + 2000);  int nMonth = (int)(Math.*random*()\*12)+1;  int nDate = (int)(Math.*random*()\*31) + 1;  String sMonth = new String();  String sDate = new String();    sMonth = (nMonth < 10) ? sMonth = "0"+String.*valueOf*(nMonth) : String.*valueOf*(nMonth);  sDate = (nDate < 10) ? sDate = "0"+String.*valueOf*(nDate) : String.*valueOf*(nDate);    return year + sMonth + sDate;  }    public static String[] *LAST\_NAMES* = {"김", "송", "허", "신", "박"};  public static String[] *FIRST\_NAMES* = {"훈", "일", "재", "준", "성", "철", "수", "연", "민", "지", "선", "영", "도", "양", "원"};    public static String generateName() {  return *LAST\_NAMES*[(int)(Math.*random*()\**LAST\_NAMES*.length)] + *FIRST\_NAMES*[(int)(Math.*random*()\**FIRST\_NAMES*.length)] + *FIRST\_NAMES*[(int)(Math.*random*()\**FIRST\_NAMES*.length)];  }    // 최대 10자리  public static String generateId() {  int places = (int)(Math.*random*()\*7) + 4;  String id = new String();    for (int i = 0; i < places; ++i) {  id += (i%((int)(Math.*random*()\*2)+1)) == 0 ? *getRandomAlphabet*() : String.*valueOf*((int)(Math.*random*()\*10));  }    return id;  }    public static char getRandomAlphabet() {  return (char)(((int)(Math.*random*()\*(int)('z'-'a')))+(int)'a');  }    public static String generateGender() {  return (1%((int)(Math.*random*()\*2)+1)) == 0 ? "남" : "여";  }    public static String[] *DOMAINS* = {"naver.com", "google.com", "amazon.com"};    public static String generateEmail(String id) {  return id+"@"+*DOMAINS*[(int)(Math.*random*()\**DOMAINS*.length)];  }    public static String generatePhone() {  String phone = "010-";    for (int i = 0; i < 4; ++i) {  phone += String.*valueOf*((int)(Math.*random*()\*10));  }    phone+="-";    for (int i = 0; i < 4; ++i) {  phone += String.*valueOf*((int)(Math.*random*()\*10));  }    return phone;  }  }  **- Printable 클래스**  package data;  public interface Printable {  public abstract void printMember();  }  **- Student 클래스**  package data;  /\*\*  \* **@apiNote** Student 구조를 담당할 클래스, value는 StudentVO에게 일임.  \*  \* 첫 개발 일자 : 2021\_02\_03  \* last\_updated : 2021\_02\_19  \*  \*/  public class Student extends Person {  private ScoreTable scoreTable;  private int courseId;    public Student() {  this(null, null, 0, null, null, null, null, null);  }    /\*  \* public Student(String birthdate, String name, String id, String gender,  String email, String phone, ScoreTable scoreTable, int courseId) {  \*/  public Student(String id, String name, int courseId, String birthdate, String gender,  String email, String phone, ScoreTable scoreTable) {  super(id, name, birthdate, gender, email, phone);  this.courseId = courseId;  this.scoreTable = scoreTable;  }    *@Override*  public String toString() {  return super.toString() + ", " + this.scoreTable.toString();  }    *@Override*  public void printMember() {  System.***out***.println(this.toString());  }    *@Override*  public int hashCode() {  // assume all student objects are equal at the hashCode comparison stage.  return 0;  }    *@Override*  public boolean equals(Object obj) {  boolean flag = false;  if (obj instanceof Student) {  Student arg = (Student)obj;  if (super.getId().equals(arg.getId())  || super.getPhone().equals(arg.getPhone()))  flag = true;  }  return flag;  }    public ScoreTable getScoreTable() { return scoreTable; }  public void setScoreTable(ScoreTable st) { scoreTable = st; }    public int getCourseId() { return courseId; }  public void setCourseId(int courseId) { this.courseId = courseId; }  }  **- ScoreTable 클래스**  package data;  import java.util.ArrayList;  /\*\*  \* @apiNote 학생 테이블이 가지고 있는 성적 계산을 담당하려 했던 클래스인데  \* 이 모든 일을 DB에서 처리하면 된다 걸 알았으므로 필요가 없어졌다.  \*  \* 첫 개발 일자 : 2021\_02\_03  \* last\_updated : 2021\_02\_19  \*  \*/  public class ScoreTable implements Printable {  public static final int MAX\_VALUE = 100;  public static final int MIN\_VALUE = 0;    enum Subjects {  C, JAVA, ANDROID, WEB, LAST  }  public static final int AVG\_FLOAT\_POINT\_PLACE\_EXP = 2;    private ArrayList<Integer> scores;  private int rank;    public ScoreTable() {  this(0, 0, 0, 0);  }    public ScoreTable(int c, int java, int android, int web) {  scores = new ArrayList<Integer>(Subjects.LAST.ordinal());  for (int i = 0; i < Subjects.LAST.ordinal(); ++i) {  scores.add(0);  }    setC(c);  setJava(java);  setAndroid(android);  setWeb(web);  }    @Override  public String toString() {  return "rank:" + getRank() + ", C:" + getC() + ", Java:" + getJava() + ", Android:"  + getAndroid() + ", Web:" + getWeb() + ", Total:" + getTotal() + ", Avg:" + getAvg() + ", grade:" + getGrade();  }    @Override  public void printMember() {  System.out.println(this.toString());  }    public void setC(int val) {  if (ScoreTable.MAX\_VALUE < val || val < ScoreTable.MIN\_VALUE) {  System.out.println("범위 초과");  scores.set(Subjects.C.ordinal(), (int)(Math.random()\*(ScoreTable.MAX\_VALUE+1)));  } else {  scores.set(Subjects.C.ordinal(), val);  }  }    public int getC() {  return scores.get(Subjects.C.ordinal());  }    public void setJava(int val) {  if (ScoreTable.MAX\_VALUE < val || val < ScoreTable.MIN\_VALUE) {  System.out.println("범위 초과");  scores.set(Subjects.JAVA.ordinal(), (int)(Math.random()\*(ScoreTable.MAX\_VALUE+1)));  } else {  scores.set(Subjects.JAVA.ordinal(), val);  }  }    public int getJava() {  return scores.get(Subjects.JAVA.ordinal());  }    public void setWeb(int val) {  if (ScoreTable.MAX\_VALUE < val || val < ScoreTable.MIN\_VALUE) {  System.out.println("범위 초과");  scores.set(Subjects.WEB.ordinal(), (int)(Math.random()\*(ScoreTable.MAX\_VALUE+1)));  } else {  scores.set(Subjects.WEB.ordinal(), val);  }  }    public int getWeb() {  return scores.get(Subjects.WEB.ordinal());  }    public void setAndroid(int val) {  if (ScoreTable.MAX\_VALUE < val || val < ScoreTable.MIN\_VALUE) {  System.out.println("범위 초과");  scores.set(Subjects.ANDROID.ordinal(), (int)(Math.random()\*(ScoreTable.MAX\_VALUE+1)));  } else {  scores.set(Subjects.ANDROID.ordinal(), val);  }  }    public int getAndroid() {  return scores.get(Subjects.ANDROID.ordinal());  }    public void setRank(int rank) {  this.rank = rank;  }    public int getRank() {  return rank;  }    public int getTotal() {  int sum = 0;    for (Integer score : scores) {  sum += score;  }    return sum;  }    public double getAvg() {  int total = getTotal();  double avgFloatPointPlace = Math.pow(10, AVG\_FLOAT\_POINT\_PLACE\_EXP);    return Math.round(avgFloatPointPlace\*total/scores.size())/avgFloatPointPlace;  }    public String getGrade() {  String grade = new String();    switch((int)(getAvg()/10)) {  case 10: case 9: grade = "A"; break;  case 8: case 7: grade = "B"; break;  case 6: case 5: grade = "C"; break;  case 4: case 3: grade = "D"; break;  default: grade = "F"; break;  }    return grade;  }    public static int generateRandomScore() {  return (int)(Math.random()\*101);  }    public static void calcRank(ArrayList<ScoreTable> array) {  for (int i = 0; i < array.size(); ++i) {  int count = 1;  for (int j = 0; j < array.size(); ++j) {  if (array.get(i).getTotal() < array.get(j).getTotal()) count++;  }  array.get(i).setRank(count);  }  }  }  **- ScoreVO 클래스**  **package model;**  **import data.Student;**  **import javafx.beans.property.SimpleDoubleProperty;**  **import javafx.beans.property.SimpleIntegerProperty;**  **import javafx.beans.property.SimpleStringProperty;**  **/\*\***  **\* @apiNote JavaFX UI tableColumn에 값 바인딩을 위해 값만을 가지고 있을 StudentVO 클래스.**  **\* 실제 Student 구조는 Student 클래스에서 담당**  **\***  **\* 첫 개발 일자 : 2021\_02\_03**  **\* last\_updated : 2021\_02\_19**  **\***  **\*/**  **public class StudentVO {**  **private SimpleStringProperty birthDate;**  **private SimpleStringProperty name;**  **private SimpleStringProperty id;**  **private SimpleStringProperty gender;**  **private SimpleStringProperty email;**  **private SimpleStringProperty phone;**  **private SimpleIntegerProperty courseId;**  **private SimpleIntegerProperty c;**  **private SimpleIntegerProperty java;**  **private SimpleIntegerProperty android;**  **private SimpleIntegerProperty web;**  **private SimpleIntegerProperty total;**  **private SimpleDoubleProperty avg;**  **private SimpleStringProperty grade;**    **public StudentVO(Student s) {**  **this.birthDate = new SimpleStringProperty(s.getBirthdate());**  **this.name = new SimpleStringProperty(s.getName());**  **this.id = new SimpleStringProperty(s.getId());**  **this.courseId = new SimpleIntegerProperty(s.getCourseId());**  **this.gender = new SimpleStringProperty(s.getGender());**  **this.email = new SimpleStringProperty(s.getEmail());**  **this.phone = new SimpleStringProperty(s.getPhone());**  **this.grade = new SimpleStringProperty(s.getScoreTable().getGrade());**  **this.c = new SimpleIntegerProperty(s.getScoreTable().getC());**  **this.java = new SimpleIntegerProperty(s.getScoreTable().getJava());**  **this.android = new SimpleIntegerProperty(s.getScoreTable().getAndroid());**  **this.web = new SimpleIntegerProperty(s.getScoreTable().getWeb());**  **this.total = new SimpleIntegerProperty(s.getScoreTable().getTotal());**  **this.avg = new SimpleDoubleProperty(s.getScoreTable().getAvg());**  **}**    **public String getBirthDate() { return birthDate.get(); }**  **public void setBirthDate(String birthdate) { this.birthDate.set(birthdate); }**  **public String getName() { return name.get(); }**  **public void setName(String name) { this.name.set(name); }**  **public String getId() { return id.get(); }**  **public String getGender() { return gender.get(); }**  **public void setGender(String gender) { this.gender.set(gender); }**  **public String getEmail() { return email.get(); }**  **public void setEmail(String email) { this.email.set(email); }**  **public String getPhone() { return phone.get(); }**  **public void setPhone(String phone) { this.phone.set(phone); }**  **public String getGrade() { return grade.get(); }**  **public void setGrade(String grade) { this.grade.set(grade); }**    **public int getCourseId() { return courseId.get(); }**  **public int getC() { return c.get(); }**  **public void setC(int val) { this.c.set(val); }**  **public int getJava() { return java.get(); }**  **public void setJava(int val) { this.java.set(val); }**  **public int getAndroid() { return android.get(); }**  **public void setAndroid(int val) { this.android.set(val); }**  **public int getWeb() { return web.get(); }**  **public void setWeb(int val) { this.web.set(val); }**  **public int getTotal() { return total.get(); }**  **public void setTotal(int total) { this.total.set(total); }**  **public double getAvg() { return avg.get(); }**  **public void setAvg(double avg) { this.avg.set(avg); }**  **}**  **- MySQLQueries 클래스**  **package mysql;**  **import java.sql.Connection;**  **import java.sql.PreparedStatement;**  **import java.sql.ResultSet;**  **import java.sql.SQLException;**  **import java.util.ArrayList;**  **import java.util.HashSet;**  **import data.ScoreTable;**  **import data.Student;**  **/\*\***  **\* @apiNote MySQL와 연결 - 쿼리 - 연결 종료 를 담당할 클래스**  **\***  **\* 첫 개발 일자 : 2021\_02\_03**  **\* last\_updated : 2021\_02\_19**  **\***  **\*/**  **public class MySQLQueries {**    **public static final String \_TABLE = "student\_tbl";**    **/\*\***  **\* @apiNote MySQL create table 호출 함수**  **\* @param name 생성하려는 table 이름**  **\* @return 성공 true 실패 false**  **\*/**  **public static boolean createTable(String name) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**  **String createTableQuery = "create table " + name + "("**  **+ "`id` varchar(10) NOT NULL,"**  **+ "`name` varchar(10) not null,"**  **+ "`course\_id` int not null,"**  **+ "`birthdate` varchar(10) not null,"**  **+ "`gender` varchar(1) not null,"**  **+ "`email` varchar(40) not null,"**  **+ "`phone` varchar(20) not null,"**  **+ "`c` int not null,"**  **+ "`java` int not null,"**  **+ "`android` int not null,"**  **+ "`web` int not null,"**  **+ "constraint `pk\_id` PRIMARY KEY (`id`),"**  **+ "constraint `fk\_course\_id` foreign key(`course\_id`) references `course\_tbl`(`id`) on delete cascade on update cascade"**  **+ ") ENGINE=InnoDB DEFAULT CHARSET=utf8";**  **String dropTableQuery = "drop table if exists student";**    **PreparedStatement preparedStatement = null;**    **try {**  **preparedStatement = connection.prepareStatement(dropTableQuery);**  **preparedStatement.executeUpdate();**  **preparedStatement.close();**    **preparedStatement = connection.prepareStatement(createTableQuery);**  **int count = preparedStatement.executeUpdate();**  **if (count == 0) {**  **System.out.println("Create table " + name + " completed.");**  **flag = true;**  **} else {**  **System.out.println("Create table " + name + " failed.");**  **}**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**    **return flag;**  **}**    **/\*\***  **\* @apiNote student\_tbl에 insert 쿼리 함수**  **\* @param s 삽입하려는 Student 객체**  **\* @return 성공 true 실패 false**  **\*/**  **public static boolean insertStudent(Student s) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**  **String insertQuery = "insert into "**  **+ \_TABLE**  **+ "(`id`, `name`, `course\_id`, `birthdate`, `gender`, `email`, `phone`, `c`, `java`, `android`, `web`) "**  **+ " value(?, ?, ?, ?, ?, ?, ?, ?, ?, ?, ?)";**    **PreparedStatement preparedStatement = null;**  **try {**  **preparedStatement = connection.prepareStatement(insertQuery);**  **preparedStatement.setString(1, s.getId());**  **preparedStatement.setString(2, s.getName());**  **preparedStatement.setInt(3, s.getCourseId());**  **preparedStatement.setString(4, s.getBirthdate());**  **preparedStatement.setString(5, s.getGender());**  **preparedStatement.setString(6, s.getEmail());**  **preparedStatement.setString(7, s.getPhone());**  **preparedStatement.setInt(8, s.getScoreTable().getC());**  **preparedStatement.setInt(9, s.getScoreTable().getJava());**  **preparedStatement.setInt(10, s.getScoreTable().getAndroid());**  **preparedStatement.setInt(11, s.getScoreTable().getWeb());**    **int count = preparedStatement.executeUpdate();**  **if (count == 1) {**  **System.out.println("Insert Completed");**  **flag = true;**  **} else {**  **System.out.println("Insert Failed");**  **}**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**  **return flag;**  **}**    **/\*\***  **\* @apiNote student\_tbl에 update 쿼리 함수**  **\* @param s 갱신하려는 Student 객체**  **\* @return 성공 true 실패 false**  **\*/**  **public static boolean updateStudentWhereId(Student s) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**  **String updateQuery = "update " + \_TABLE + " set name=?, course\_id=?, birthdate=?, gender=?, email=?, phone=?, "**  **+ "c=?, java=?, android=?, web=? where id = ?";**    **PreparedStatement preparedStatement = null;**  **try {**  **preparedStatement = connection.prepareStatement(updateQuery);**  **preparedStatement.setString(1, s.getName());**  **preparedStatement.setInt(2, s.getCourseId());**  **preparedStatement.setString(3, s.getBirthdate());**  **preparedStatement.setString(4, s.getGender());**  **preparedStatement.setString(5, s.getEmail());**  **preparedStatement.setString(6, s.getPhone());**  **preparedStatement.setInt(7, s.getScoreTable().getC());**  **preparedStatement.setInt(8, s.getScoreTable().getJava());**  **preparedStatement.setInt(9, s.getScoreTable().getAndroid());**  **preparedStatement.setInt(10, s.getScoreTable().getWeb());**  **preparedStatement.setString(11, s.getId());**    **int count = preparedStatement.executeUpdate();**  **if (count == 1) {**  **System.out.println("Update Completed");**  **flag = true;**  **} else {**  **System.out.println("Update Failed");**  **}**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**  **return flag;**  **}**  **/\*\***  **\* @apiNote student\_tbl 검색 함수**  **\* 아래 파라미터 중 원하는 조건의 파라미터만 넘겨주면 된다. 원하지 않는 값들은 타입의 디폴트값으로 설정한다.**  **\* @param set**  **\* @param id**  **\* @param name**  **\* @param birthdate**  **\* @param gender**  **\* @param email**  **\* @param phone**  **\* @return 성공 true (검색 결과가 없어도 true) 실패 false**  **\*/**  **public static boolean selectStudent(HashSet<Student> set, String id, int courseId, String name,**  **String birthdate, String gender, String email, String phone) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**    **String query = "select \* from " + \_TABLE + " where ";**    **query += "id LIKE '";**  **query += (id.length() != 0) ? id + "'": "%'";**  **query += (courseId != 0) ? "AND course\_id = " + courseId: "";**  **query += " AND name LIKE '";**  **query += (name.length() != 0) ? name + "'" : "%'";**  **query += " AND birthdate LIKE '";**  **query += (birthdate.length() != 0) ? birthdate + "'": "%'";**  **query += " AND gender LIKE '";**  **query += (gender != null) ? gender + "'" : "%'";**  **query += " AND email LIKE '";**  **query += (email != null) ? email + "'" : "%'";**  **query += " AND phone LIKE '";**  **query += (phone.length() != 0) ? phone + "';" : "%';";**    **PreparedStatement preparedStatement = null;**  **ResultSet resSet = null;**    **try {**  **preparedStatement = connection.prepareStatement(query);**    **resSet = preparedStatement.executeQuery();**    **set.clear();**  **while(resSet.next()) {**  **set.add(new Student(**  **resSet.getString(1),**  **resSet.getString(2),**  **Integer.parseInt(resSet.getString(3)),**  **resSet.getString(4),**  **resSet.getString(5),**  **resSet.getString(6),**  **resSet.getString(7),**  **new ScoreTable(resSet.getInt(8),resSet.getInt(9),resSet.getInt(10),resSet.getInt(11))**  **));**  **}**    **flag = true;**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**  **return flag;**  **}**    **/\*\***  **\* @apiNote 모든 student\_tbl 레코드 select**  **\* @param set**  **\* @return**  **\*/**  **public static boolean selectStudent(HashSet<Student> set) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**  **String selectQuery = "select \* from " + \_TABLE;**    **PreparedStatement preparedStatement = null;**  **ResultSet resSet = null;**    **try {**  **preparedStatement = connection.prepareStatement(selectQuery);**    **resSet = preparedStatement.executeQuery();**    **set.clear();**  **while(resSet.next()) {**  **set.add(new Student(**  **resSet.getString(1),**  **resSet.getString(2),**  **Integer.parseInt(resSet.getString(3)),**  **resSet.getString(4),**  **resSet.getString(5),**  **resSet.getString(6),**  **resSet.getString(7),**  **new ScoreTable(resSet.getInt(8),resSet.getInt(9),resSet.getInt(10),resSet.getInt(11))**  **));**  **}**    **flag = true;**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**  **return flag;**  **}**    **/\*\***  **\* @apiNote 모든 student\_tbl 레코드 정렬해서 select**  **\* @param list**  **\* @return**  **\*/**  **public static boolean selectStudentOrderById(ArrayList<Student> list) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**  **String selectQuery = "select \* from " + \_TABLE + " order by id";**    **PreparedStatement preparedStatement = null;**  **ResultSet resSet = null;**    **try {**  **preparedStatement = connection.prepareStatement(selectQuery);**    **resSet = preparedStatement.executeQuery();**    **list.clear();**  **while(resSet.next()) {**  **list.add(new Student(**  **resSet.getString(1),**  **resSet.getString(2),**  **Integer.parseInt(resSet.getString(3)),**  **resSet.getString(4),**  **resSet.getString(5),**  **resSet.getString(6),**  **resSet.getString(7),**  **new ScoreTable(resSet.getInt(8),resSet.getInt(9),resSet.getInt(10),resSet.getInt(11))**  **));**  **}**    **flag = true;**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**  **return flag;**  **}**    **/\*\***  **\* @apiNote student\_tbl에서 Student의 id를 기준으로 delete 쿼리**  **\* @param id**  **\* @return 성공 true 실패 false**  **\*/**  **public static boolean deleteStudentById(String id) {**  **boolean flag = false;**  **Connection connection = MySQLUtil.getConnection();**    **String deleteQuery = "delete from " + \_TABLE + " where id = ?";**  **PreparedStatement preparedStatement = null;**    **try {**  **preparedStatement = connection.prepareStatement(deleteQuery);**  **preparedStatement.setString(1, id);**  **int count = preparedStatement.executeUpdate();**    **if (count == 1) {**  **System.out.println("Delete Completed");**  **flag = true;**  **} else {**  **System.out.println("Delete Failed");**  **}**  **} catch (SQLException e) {**  **e.printStackTrace();**  **} finally {**  **try {**  **if (preparedStatement != null) preparedStatement.close();**  **if (connection != null) connection.close();**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**  **}**    **return flag;**  **}**  **}**  **- MySQLUtil 클래스**  **package mysql;**  **import java.io.FileReader;**  **import java.io.IOException;**  **import java.sql.Connection;**  **import java.sql.DriverManager;**  **import java.sql.SQLException;**  **import java.util.Properties;**  **/\*\***  **\* @apiNote db priperties 파일을 바탕으로 MySQL 최초 설정 담당 클래스**  **\***  **\* 첫 개발 일자 : 2021\_02\_03**  **\* last\_updated : 2021\_02\_19**  **\***  **\*/**  **public class MySQLUtil {**  **public static Connection getConnection() {**  **Properties properties = new Properties();**  **String path = MySQLUtil.class.getResource("db.properties").getPath();**  **path = path.substring(1);**    **try {**  **path = path.replaceAll("/", "\\\\");**  **properties.load(new FileReader(path));**  **} catch (IOException e) {**  **e.printStackTrace();**  **} catch (Exception e) {**  **e.printStackTrace();**  **}**    **try {**  **// MySQL 드라이버 메소드 영역에 적재**  **Class.forName(properties.getProperty("MYSQL\_DRIVER"));**  **} catch (Exception e) {**  **e.printStackTrace();**  **}**    **Connection connection = null;**  **try {**  **connection = DriverManager.getConnection(**  **properties.getProperty("URL"),**  **properties.getProperty("userName"),**  **properties.getProperty("password"));**    **if (connection != null) {**  **System.out.println("connected to student schema");**  **}**  **} catch (SQLException e) {**  **e.printStackTrace();**  **}**    **return connection;**  **}**  **}** |  |  | | --- | | **JavaFX fxml 구현** | | **- root.fxml**  <?**xml** version=*"1.0"* encoding=*"UTF-8"*?>  <?**import** javafx.geometry.Insets?>  <?**import** javafx.scene.control.Button?>  <?**import** javafx.scene.control.Label?>  <?**import** javafx.scene.control.TableColumn?>  <?**import** javafx.scene.control.TableView?>  <?**import** javafx.scene.layout.AnchorPane?>  <?**import** javafx.scene.layout.BorderPane?>  <?**import** javafx.scene.text.Font?>  <**BorderPane** prefHeight=*"900.0"* prefWidth=*"900.0"* xmlns=*"http://javafx.com/javafx/11.0.1"* xmlns:fx=*"http://javafx.com/fxml/1"* fx:controller=*"application.RootController"*>  <**top**>  <**Label** text=*"MySQL - Java - JavaFX 연동"* textAlignment=*"CENTER"* BorderPane.alignment=*"CENTER"*>  <**font**>  <**Font** size=*"50.0"* />  </**font**>  <**opaqueInsets**>  <**Insets** />  </**opaqueInsets**>  <**BorderPane.margin**>  <**Insets** bottom=*"20.0"* left=*"20.0"* right=*"20.0"* top=*"20.0"* />  </**BorderPane.margin**>  </**Label**>  </**top**>  <**center**>  <**TableView** fx:id=*"tableView"* prefHeight=*"271.0"* prefWidth=*"856.0"* BorderPane.alignment=*"CENTER"*>  <**columns**>  <**TableColumn** prefWidth=*"75.0"* text=*"id"* />  <**TableColumn** prefWidth=*"75.0"* text=*"courseId"* />  <**TableColumn** prefWidth=*"75.0"* text=*"name"* />  <**TableColumn** prefWidth=*"75.0"* text=*"birthdate"* />  <**TableColumn** prefWidth=*"75.0"* text=*"gender"* />  <**TableColumn** prefWidth=*"75.0"* text=*"email"* />  <**TableColumn** prefWidth=*"75.0"* text=*"phone"* />  <**TableColumn** prefWidth=*"75.0"* text=*"c"* />  <**TableColumn** prefWidth=*"75.0"* text=*"java"* />  <**TableColumn** prefWidth=*"75.0"* text=*"android"* />  <**TableColumn** prefWidth=*"75.0"* text=*"web"* />  <**TableColumn** prefWidth=*"75.0"* text=*"total"* />  <**TableColumn** prefWidth=*"75.0"* text=*"avg"* />  <**TableColumn** prefWidth=*"75.0"* text=*"grade"* />  </**columns**>  </**TableView**>  </**center**>  <**bottom**>  <**AnchorPane** prefHeight=*"100.0"* prefWidth=*"641.0"* BorderPane.alignment=*"CENTER"*>  <**children**>  <**Button** alignment=*"CENTER"* layoutX=*"100.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleInsert"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"입력"* />  <**Button** alignment=*"CENTER"* layoutX=*"225.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleUpdate"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"수정"* />  <**Button** alignment=*"CENTER"* layoutX=*"600.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleDelete"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"삭제"* />  <**Button** alignment=*"CENTER"* layoutX=*"350.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleSearch"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"검색"* />  <**Button** alignment=*"CENTER"* layoutX=*"475.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleSort"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"정렬"* />  <**Button** alignment=*"CENTER"* layoutX=*"720.0"* layoutY=*"30.0"* mnemonicParsing=*"false"* onAction=*"#handleQuit"* prefHeight=*"40.0"* prefWidth=*"80.0"* text=*"종료"* />  </**children**>  </**AnchorPane**>  </**bottom**>  </**BorderPane**>  **- form.fxml**  <?**xml** version=*"1.0"* encoding=*"UTF-8"*?>  <?**import** java.lang.String?>  <?**import** javafx.collections.FXCollections?>  <?**import** javafx.scene.control.Button?>  <?**import** javafx.scene.control.ComboBox?>  <?**import** javafx.scene.control.Label?>  <?**import** javafx.scene.control.TextField?>  <?**import** javafx.scene.layout.ColumnConstraints?>  <?**import** javafx.scene.layout.GridPane?>  <?**import** javafx.scene.layout.RowConstraints?>  <**GridPane** maxHeight=*"-Infinity"* maxWidth=*"-Infinity"* minHeight=*"-Infinity"* minWidth=*"-Infinity"* prefHeight=*"400.0"* prefWidth=*"600.0"* xmlns=*"http://javafx.com/javafx/11.0.1"* xmlns:fx=*"http://javafx.com/fxml/1"*>  <**columnConstraints**>  <**ColumnConstraints** hgrow=*"SOMETIMES"* minWidth=*"10.0"* prefWidth=*"100.0"* />  <**ColumnConstraints** hgrow=*"SOMETIMES"* minWidth=*"10.0"* prefWidth=*"100.0"* />  </**columnConstraints**>  <**rowConstraints**>  <**RowConstraints** maxHeight=*"128.0"* minHeight=*"10.0"* prefHeight=*"30.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"128.0"* minHeight=*"0.0"* prefHeight=*"28.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"267.0"* minHeight=*"0.0"* prefHeight=*"35.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"300.0"* minHeight=*"10.0"* prefHeight=*"42.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"262.0"* minHeight=*"10.0"* prefHeight=*"44.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"256.0"* minHeight=*"10.0"* prefHeight=*"45.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"10.0"* prefHeight=*"50.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"0.0"* prefHeight=*"43.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"10.0"* prefHeight=*"56.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"10.0"* prefHeight=*"47.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"10.0"* prefHeight=*"47.0"* vgrow=*"SOMETIMES"* />  <**RowConstraints** maxHeight=*"264.0"* minHeight=*"10.0"* prefHeight=*"47.0"* vgrow=*"SOMETIMES"* />  </**rowConstraints**>  <**children**>  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"id"* GridPane.rowIndex=*"1"* />  <**TextField** fx:id=*"id"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"1"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"name"* GridPane.rowIndex=*"2"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"email"* GridPane.rowIndex=*"5"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"gender"* GridPane.rowIndex=*"4"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"birthdate"* GridPane.rowIndex=*"3"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"phone"* GridPane.rowIndex=*"6"* />  <**TextField** fx:id=*"name"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"2"* />  <**TextField** fx:id=*"birthdate"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"3"* />  <**TextField** fx:id=*"phone"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"6"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"c"* GridPane.rowIndex=*"7"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"java"* GridPane.rowIndex=*"8"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"android"* GridPane.rowIndex=*"9"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"web"* GridPane.rowIndex=*"11"* />  <**TextField** fx:id=*"c"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"7"* />  <**TextField** fx:id=*"java"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"8"* />  <**TextField** fx:id=*"android"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"9"* />  <**ComboBox** fx:id=*"email"* prefHeight=*"22.0"* prefWidth=*"303.0"* visibleRowCount=*"3"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"5"*>  <**items**>  <**FXCollections** fx:factory=*"observableArrayList"*>  <**String** fx:value=*"@naver.com"* />  <**String** fx:value=*"@gmail.com"* />  <**String** fx:value=*"@amazon.com"* />  </**FXCollections**>  </**items**>  </**ComboBox**>  <**Button** fx:id=*"confirm"* alignment=*"CENTER"* contentDisplay=*"CENTER"* mnemonicParsing=*"false"* prefHeight=*"22.0"* prefWidth=*"302.0"* text=*"실행"* GridPane.rowIndex=*"11"* />  <**Button** fx:id=*"cancel"* alignment=*"CENTER"* contentDisplay=*"CENTER"* mnemonicParsing=*"false"* prefHeight=*"22.0"* prefWidth=*"301.0"* text=*"취소"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"11"* />  <**ComboBox** fx:id=*"gender"* prefHeight=*"22.0"* prefWidth=*"303.0"* visibleRowCount=*"3"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"4"*>  <**items**>  <**FXCollections** fx:factory=*"observableArrayList"*>  <**String** fx:value=*"남"* />  <**String** fx:value=*"여"* />  </**FXCollections**>  </**items**>  </**ComboBox**>  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"courseId"* />  <**TextField** fx:id=*"courseId"* GridPane.columnIndex=*"1"* />  <**Label** alignment=*"CENTER"* contentDisplay=*"CENTER"* prefHeight=*"14.0"* prefWidth=*"302.0"* text=*"web"* GridPane.rowIndex=*"10"* />  <**TextField** fx:id=*"web"* GridPane.columnIndex=*"1"* GridPane.rowIndex=*"10"* />  </**children**>  </**GridPane**>  **- messageDialog.fxml**  <?**xml** version=*"1.0"* encoding=*"UTF-8"*?>  <?**import** javafx.scene.control.Button?>  <?**import** javafx.scene.control.Label?>  <?**import** javafx.scene.layout.AnchorPane?>  <**AnchorPane** prefHeight=*"97.0"* prefWidth=*"301.0"* xmlns:fx=*"http://javafx.com/fxml/1"* xmlns=*"http://javafx.com/javafx/11.0.1"*>  <**children**>  <**Label** fx:id=*"message"* layoutX=*"14.0"* layoutY=*"14.0"* prefHeight=*"41.0"* prefWidth=*"275.0"* />  <**Button** fx:id=*"confirm"* layoutX=*"228.0"* layoutY=*"61.0"* mnemonicParsing=*"false"* text=*"확인"* />  </**children**>  </**AnchorPane**> |  |  | | --- | | **실행 결과** | |  | |