|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| **□ 수행평가 - 빅데이터를 활용한 IoT 시스템 개발** | | | | | | |
|  |  |  | |  | |  |
| **과정명** | | 빅데이터를 활용한 IoT시스템 개발 | | | | |
| **교과목명** | | 기반기술 | | **훈련교사** | | 이진만 |
| **과정명** | | 응용SW기술이해  Database 구문 이해 | | | | |
| **수행날짜** | | 2020.7.31 | 훈련생명 | | 최인정 | |
| **과제개요** | | | | | | |
| * Java Program과 Oracle Database를 연동하는 프로젝트를 기획한다. * Oracle Database에 Table을 구축한다. * Java를 이용하여 DAO를 작성하고 테스트를 진행한다. * Java AWT를 이용하여 UI를 개발한다. * UI와 DAO를 통합하여 프로젝트를 완료한다. | | | | | | |
| 작성 내용   * 기획 내용   : 학생의 성적관리를 위한 프로그램 개발  학생(Student)과 교사(Instructor)에 따라 기능이 달라지는 프로그램이다. 로그인 과정에서 선생님인지 학생인지 확인하도록 하고 각각의 데이터베이스와 비교하여 로그인이 진행 되도록 한다. 교사는 데이터베이스에 새로운 학생을 추가하거나 기존에 있는 학생들의 성적을 부여할 수 있고, 학생은 자신의 성적과 학점을 확인할 수 있는 통합 성적관리 프로그램을 구현한다. 교사가 성적을 부여하면 학생의 등급은 자동으로 부여된다. (95점 이상: A+, 90점 이상: A, 85점 이상: B+, 80점 이상: B, 75점 이상: C+, 70점 이상: C, 65점 이상: D+, 60점 이상: D, 60점 미만: F)   * 시스템구성도      * 사용 소프트웨어   : Java - 8버전, IDE - Eclipse 4.16.0, 오라클 - 11g   * 데이터베이스 구축 SQL문   **[ Students Table ]**  CREATE TABLE STUDENTS(  ID VARCHAR2(20) PRIMARY KEY,  PWD VARCHAR2(10),  NAME VARCHAR2(20),  SCORE NUMBER(5,2),  GRADE VARCHAR2(10)  );  **[ Instructors Table ]**  CREATE TABLE INSTRUCTORS(  ID VARCHAR2(20) PRIMARY KEY,  PWD VARCHAR2(10),  NAME VARCHAR2(20)  );   * 구동 화면   **[ Login 화면 ]**    **[ Student(학생) 화면 ]**    **[ Instructor(선생님) 화면 ]**     * 구현 소스   **[ Student와 Instructor를 구분하기 위한 체크박스 ]**  cb1 = **new** JCheckBox("STUDENT");  cb2 = **new** JCheckBox("INSTRUCTOR");  bg.add(cb1);  bg.add(cb2);  p1.add(cb1);  p1.add(cb2);  cb1.addItemListener(**new** ItemListener() {  @Override  **public** **void** itemStateChanged(ItemEvent e) {  **if** (e.getStateChange() == ItemEvent.***SELECTED***) {  a = "STUDENT";  }  }  });  cb2.addItemListener(**new** ItemListener() {  @Override  **public** **void** itemStateChanged(ItemEvent e) {  **if** (e.getStateChange() == ItemEvent.***SELECTED***) {  a = "INSTRUCTOR";  }  }  });  **[ 로그인 시 체크박스로 사용자 판별 - Student 또는 Instructor ]**  **if** (a.equals("STUDENT")) {  **try** {  dbstudent = studentDb.select(tid);  **if** (tpwd.equals(dbstudent.getPwd())) {  frame.setVisible(**false**);  SApp app = **new** SApp(studentDb, dbstudent.getId());  } **else** {  label.setText("Invalid PWD .. !!");  }  } **catch** (Exception e) {  label.setText("Invalid ID .. !!");  }  } **else** **if** (a.equals("INSTRUCTOR")) {  **try** {  dbinstructor = instructorDb.select(tid);  System.***out***.println(dbinstructor);  System.***out***.println(dbinstructor);  **if** (tpwd.equals(dbinstructor.getPwd())) {  frame.setVisible(**false**);  IApp app = **new** IApp(studentDb);  } **else** {  label.setText("Invalid PWD .. !!");  }  } **catch** (Exception e) {  label.setText("Invalid ID .. !!");  }  }  **[ 학생 UI로 넘어올 때 로그인 한 학생의 key값 ]**  **public** SApp(Db<String, Student> studentDb, String key) {  **this**.studentDb = studentDb;  **this**.key = key;  makeUi();  eventProcess();  }  **[ 학생 정보와 학점 정보 프레임 생성 및 폰트 사이즈 설정 ]**  Frame frame;  Panel p1, p2, p3, sp1, sp2;  Button b1;  Label label;  List list1, list2; // list1 - 학생 정보 출력, list2 - 학생 학점 출력  Font font1 = **new** Font("Serif", Font.***PLAIN***, 20);  Font font2 = **new** Font("Serif", Font.***BOLD***, 80);  Db<String, Student> studentDb;  ArrayList<Student> slist;  Student s;  String key; // 로그인한 사용자의 id를 받아오는 변수  **public** **void** makeUi() {  .  .  .  list1.setFont(font1);  list2.setFont(font2);  }  **[ 학생 키값에 따른 정보를 화면에 출력 ]**  **public** **void** selectStudent(String key) {  **try** {  s = studentDb.select(key);  list1.removeAll();  list2.removeAll();  list1.add(s.getId() + " " + s.getName() + " " + s.getScore());  list2.add("Grade: " + s.getGrade());  setLabel("Search Completed !!");  } **catch** (Exception e) {  setLabel("Search Fail !!");  e.printStackTrace();  }  }  **[ 점수와 계산된 학점을 업데이트 ]**  **public** **void** updateStudent() {  **double** score = Double.*parseDouble*(utf2.getText());  String grade = gradeSet(score);  temp.setScore(score);  temp.setGrade(grade);  System.***out***.println("temp : " + temp);  **try** {  studentDb.update(temp);  selectStudent();  setLabel("Update Completed !!");  } **catch** (Exception e) {  setLabel("Update Fail !!");  }  }  **[ 선생님이 입력한 점수에 따른 학점 계산 ]**  **public** String gradeSet(**double** score) {  String grade = **null**;  **if** (score >= 95) {  grade = "A+";  } **else** **if** (score >= 90) {  grade = "A";  } **else** **if** (score >= 85) {  grade = "B+";  } **else** **if** (score >= 80) {  grade = "B";  } **else** **if** (score >= 75) {  grade = "C+";  } **else** **if** (score >= 70) {  grade = "C";  } **else** **if** (score >= 65) {  grade = "D+";  } **else** **if** (score >= 60) {  grade = "D";  } **else** {  grade = "F";  }  **return** grade;  }  **[ 학생 객체 업데이트 - score와 grade는 0으로 리셋 ]**  **public** **void** insertStudent() {  String id = tf1.getText();  String pwd = tf2.getText();  String name = tf3.getText();  Student student = **new** Student(id, pwd, name);  **try** {  studentDb.insert(student);  selectStudent();  setLabel("Insert Completed !!");  } **catch** (Exception e) {  setLabel("Insert Fail !!");  }  }  **[ 학생들의 정보 출력 ]**  **public** **void** selectStudent() {  **try** {  slist = studentDb.select();  list.removeAll();  **for** (Student s : slist) {  list.add(s.getName() + " " + s.getScore() + " " + s.getGrade());  }  setLabel("Search Completed !!");  } **catch** (Exception e) {  setLabel("Search Fail !!");  }  } | | | | | | |
|  | | | | | | |