Design an interface application class `Grade` (in Grade.java) that is extended from `JFrame`. Please use buttons, text fields, color options, and other supports (of `JFrame`) to implement a grading system for both students and faculties. System.out.print (or println) and keyboard reading is not allowed.

a. Include 2 text fields (one for name and the other for password) and a button in the construction of `Grade` to support login. There are other fields, labels, and buttons but they are invisible until the access is granted.

b. Add private attribute username and password in class `Grade`. These will be initiated as “admin” and “PASS13”. The system first launches a `Grade` object and accept a matched name and password. After the access granted, wait for initialization work (refer to `m.input(1)` in ArrayOperations.java).

c. During this initialization, one course object, >2 faculty object, and >5 student objects (>5) are initiated. **We assume that there is no duplicate name (i.e., ID).** The password of each user (student or faculty) are determined by this administrator. The course assignment (not student enrollment) is also decided here. Note that the information is inputted through interface and will be stored in objects via the corresponding constructors.

d. Make a backstage class `CourseList` supporting the singleton object in the entire system, which includes the following private attributes and the corresponding accessors and mutators:

   - `names`: an array of String, includes course sequence and title, such as CST221 Database Systems
   - `instructor_usr`: an array of String, instructor’s user name for each course (matched the above)
   - `number_of_enroll`: an array of integer, number of students who registered the class, 0 as initiated
   - `student_list`: a 2-D array of String, students’ names for each course

   Both the names and instructor_usr will be determined by the administrator in step (b), with the mutators of this class. The other two will be determined by the students’ activity in step (h). This class has the accessor to verify if a student and a faculty (passing names by arguments) are related (in the above private attributes).

e. Make an `Instructor` class, its private attributes, and related successors and mutators:

   - `username`: String type
   - `password`: String type
   - `list`: String array, a list of names of courses in teaching

   The username, password, and list will be decided in step (b). This class has an accessor to verify the login.

f. Make a `Student` class, its private attributes, and related accessors and mutators:

   - `studentID`: student user name, given by the administer.
   - `studentPass`: student password, given by the administer.
   - `courses`: a string array to store the names of selected courses.
   - `hwGrade`: a two dimensional array
   - `projGrade`: a two dimensional array
   - `testGrade`: a two dimensional array
   - `letterGrade`: an array of char type, saving each letter score

   The studentID and studentPass will be determined in step (b). There is an accessor to verify the student login. Develop 3 private associate methods (addingHWgrade, addingProjGrade, addingTestGrade). You need reNuevo the array with 1 more unit, copy the old records and set the new one. Do not use fixed size array. Note that you do not have additional attributes for size control, with the “length” of array available only. Develop a private associate method “autoCal” for letterGrade, upon any change of related score.

g. Before this administration initialization, no record can be found and any other login must be blocked.

h. After the initialization, all actions are initiated from login window in (a). This is a combination of buttons’ responses, and its general format is described in the actionPerformed method of `Grade`. Those buttons can be initiated in constructor of `Grade` object, but may change its visibility.

   - If login button is clicked and a faculty logs in (via verification), faculty update button is on.
   - If login button is clicked by a student (verified), register, cancel, and view buttons are on.
   - If an update button is clicked, the student in that class taught by this user (verified) can have a chance to change any of scores.
   - If a view button is clicked, student grades will be located (with other information via interface).
   - If a register button is clicked, student will select the course to enroll. Such information will be updated into course object at the same application level that can be accessed by `Grade` object.
   - If a cancel button is clicked, the student can clean up the record in course object and the grade records.