Assignment 2 Explanation

In this algorithm, I implemented that the TA could manage students' grades in their course assignments. First of all, I initialized a list that keeps all courses assigned to each TA. If any TA is created, I added the courses assigned to the list. It gives the following options for TA to see grades of students who are enrolled in courses assigned to them. It fetches from admin's grade management system. If the TA wants to see grades, it iterates on the assigned course list and fetches grades associated with each of them for display.

It allows the TA to choose a course from his/her assigned list to update the students' grades and then to type in the name of the student and the new grade. The algorithm then checks if the selected course has grades available. If so, it updates the relevant grade in the corresponding data structure. Everything is highly user-friendly, and the menu-driven process by which the TAs will perform these activities is outlined below.

In this I created a generic class Feedback that will represent student feedback about any course he or she took. The algorithm is designed to hold such information as the student who provided the feedback, the course being reviewed, and the text of the feedback itself, which can be of any type because of the generic type parameter TA Feedback object will create these three parts using its constructor.

One of the methods I came up with includes viewFeedback(ShowCourses course), to enable professors to view student feedback for a course in which they teach. This algorithm starts with checking if the professor was assigned to that course or not. The methods that came into my mind include viewFeedback(ShowCourses course), which allows professors to view student feedback for a course they teach. This algorithm starts by checking whether the professor was assigned to that course or not.

Another method is assignTA(TA ta, ShowCourses course), which allows assigning a TA to a course. My algorithm first checks whether the professor is assigned to the course. If so, I call the assignToCourse(course) method on the TA object, linking it to that course. Finally, I print out a confirmation message identifying that successful assignment.

The most common use for the try-catch blocks in my ERP system will probably be when there is an expectation of possible input errors, usually when some input by the user needs to be converted into numbers. For example, while updating enrollment limits or course credits in the CourseDetailsManagement method, a try-catch block covers the call to Integer.parseInt(). It catches the NumberFormatException if the input isn't a valid number, thus preventing the program from crashing and showing an error message such as "Invalid input, please enter a

number." Thus, the program can handle this kind of exception more elegantly for smoother operation and also for better user experience.