Database Programming

[10% of grade]

Overview

For this assignment, you are creating a Java program that connects to your database using JDBC to build a simple course registration system for students. The program can be command line or GUI, and will allow students to enroll in courses, drop courses, view their transcript, and even check what courses they need to complete their degree.

Features

Your program should perform the following:

1) At startup, prompt the user for their student ID
    - Be sure to verify that the student exists

2) Prompt the user to select one of the operations: Get Transcript, Check Degree Requirements, Add Course, Remove Course, or Exit
    - Verify the input is a recognized operation
    - Continue to prompt the user for an operation until they choose Exit

3) Based on their input, run the appropriate queries using Dynamic SQL:

    a) **Get Transcript**: use a student's ID to query their transcript using Dynamic SQL. The transcript should include the course number, course title, semester, year, grade, and credits of every course the Student has taken in chronological order.

    b) **Check Degree Requirements:** use a student's ID to query all of the courses that the student still needs to take to complete their degree. That is, determine the course numbers and titles of courses the student not already taken, but are required for their major. Assume the required courses for a major are all of the courses for their department (e.g. all Physics courses).

    c) **Add Course**: use a student's ID and section identifier, try to enroll (via INSERT) the student in the specified section. Be sure to verify that the student is not already enrolled in the course and meets all of the prerequisites.
        - To get the section identifier, prompt the user for the semester and year, and list the course_id and sec_id of all matching sections. Number the matching sections (1, 2, 3, …) so that a user can use the number to specify which section they want to enroll in.

    d) **Remove Course**: using a student's ID and section identifier, try to remove (via DELETE) the specified section from the student's enrollment. Be sure to verify that the student is already enrolled in the course.
        - Get the section identifier similar to Add Course. List all sections they are enrolled in and prompt the user to specify which they want to remove.

Other Requirements

- The program must use JDBC, not an ORM like Hibernate
- Any queries using user input must use prepared statements
- Results must be displayed neatly
- Pass your database user id and password in as command line arguments:

```
args[0] <= user id    args[1] <= password
```

- Place the main method by itself in a Main.java class. The main method should have no program logic. It should only start the program. For example:

```
public static void main(String[] args) {
  RegistrationSystem hw = new RegistrationSystem(args[0], args[1]);
  hw.run();
}
```

- Check for and handle exceptions, so that the program does not crash

Submission

After you have completed and thoroughly tested the program, upload the following deliverables to Canvas in a single ZIP file:

- Source code (.JAVA files only)
    - Do NOT submit your password

Grading

Programs will be graded according to the following:

GENERAL
Prompts for student ID
Verifies user input is a valid student
Prompts for desired operation
Execution continues until user exits
GET TRANSCRIPT
Returns correct results, sorted appropriately, and displayed neatly
CHECK DEGREE REQUIREMENTS
Returns correct results, displayed neatly
ADD COURSE
Neatly displays available sections, based on user input
Checks if student can enroll in specified section
If checks pass, section added to student's enrollment
REMOVE COURSE
Neatly displays enrolled sections
Checks if student is enrolled in specified section
If check passes, section removed from student's enrollment
DESIGN
Reads DB credentials from command line arguments
Main method by itself in Main.java with no program logic
Queries built using prepared statements
Logic is broken up into short, simple methods
Uses constants instead of literals for reused values
Only uses static methods for utility like functions
TESTING
Program handles errors without crashing
STYLE
Descriptive names for variables, classes, methods;
Proper indentation for code blocks and spacing for legibility;
No unused or commented out code;