Instructions

Please read the instructions carefully and follow the naming conventions specified for each question. Solutions must be submitted in the Blackboard Dropbox created for the workshop 4. **The deliverables will be questions placed a single package named *btp400.lab4*. The submission shall be in single jar file (called *btp400lab4.jar*) which contains both source (*.java) and bytecode (*class) files.** Your solution should be well documented using the JavaDoc utility to describe both your interface and your solution design. Failure to submit in the described format will result in a 0.

Note that the deadline is strictly enforced. The system tracks the exact time that submissions are uploaded and late submissions will be rejected.

In this lab, you will apply the following concepts presented in the lectures.

- utilizing permissions. You will use *Java's Security Manger* to grant and deny privileges to an application
- building streams. You will create classes and methods that support multiple types
- accessing databases. You will connect and retrieve data from a remote relational database.

Additional Notes

- You must provide appropriate test case classes with your solution. Document any assumptions you make about the requirements. You will have to organize your code in a way such that the JUnit test runner does not require user input.
- you will connect and access data stored in a remote database using **JDBC**. You will be assigned your own database which you must use for this assignment. Credentials will be distributed via email. Review the following :

  1. JDBC drivers for MySQL `https://dev.mysql.com/downloads/connector/j/`
  2. Connecting using a database connection string from Java applications. `https://dev.mysql.com/doc/connector-j/8.0/en/connector-j-usagenotes-connect-drivermanager.html`

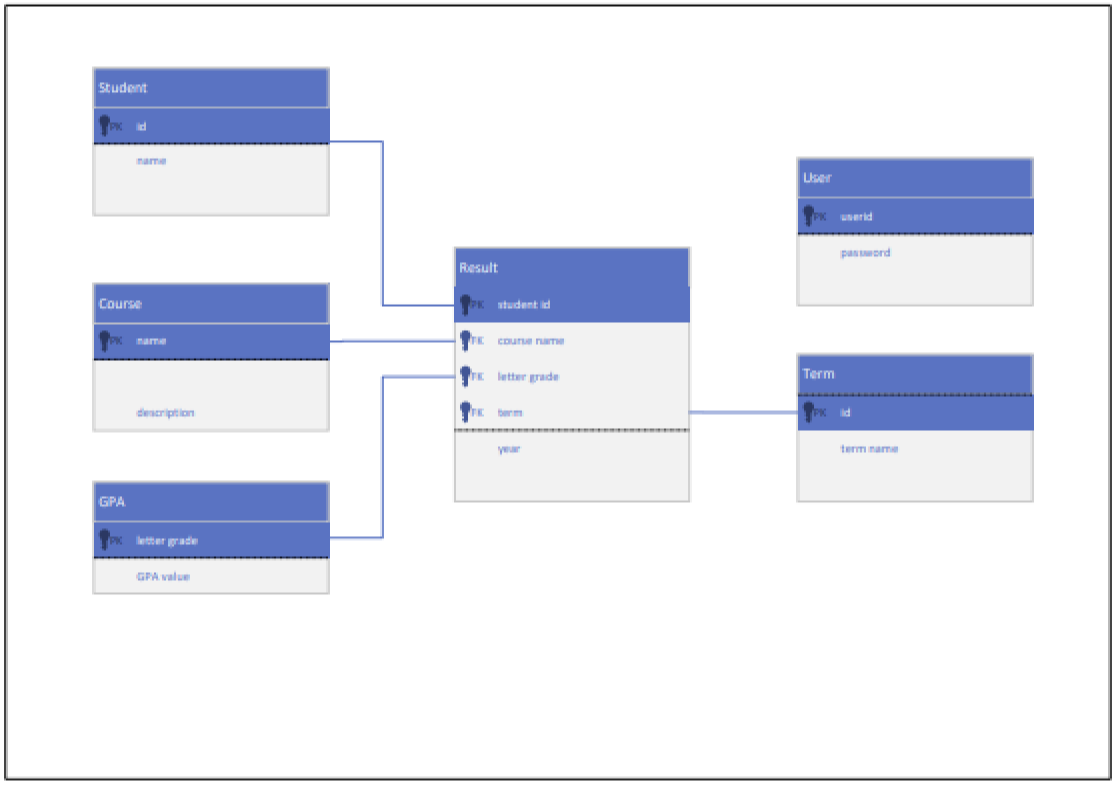

Figure 1: Database Schema

Question 1) (a) write a DB script which will pre-populate the User, GPA and Course tables. Use the recommended sources for the data.

- GPA, (`https://www.senecacollege.ca/about/policies/grading-policy.html`)
- Course, a sample set (`https://www.senecacollege.ca/programs/fulltime/BSD/courses.html`)
- User, create 2 or 3 test accounts of your choice

(b) use *JDBC* classes to complete the implementation of the following methods of the given *Student* class.

- *save*(), persist the students results to the database
- *Student*(), retreive a student's results from the database
- *printResults*()

(c) using *streams*, complete the empty implementations of the class. (Implementations that do not use Java Stream class will not receive any credit.)

- *getGPA*() , which calculates student's average GPA across all courses
- *getClassGPA* , which calculates average GPA for a given class

(d) add the following security features to the solution

- enable the *SecurityManager*. Include a policy file which allows the application to connect to your remote DB
- the string needed to connect to the database should be unmodifiable and must be stored in encrypted format.
- ensure that user passwords are encrypted while stored in the database (encrypted at rest)
- implement *login*() which generates a token used to authenticate user and check to ensure that all operations are only performed by valid users

Ensure that all exceptions are logged using Java provided logging classes.