All of your classes should exist in a *webd4201*.*lastnamefirstinitial* package (all lowercase).

For this assignment you are going to create several java classes/files, we will be building upon the existing code from previous assignments. It is suggested that you use the class files from “la2\_jdbc\_example.zip” on DC Connect as a starting point for the Connect and DA classes.

NOTES:

* Be sure to place your three (3) created \*.sql files into a sql sub-folder in that should reside in the webd4201/lastnamefirstinitial folder
* ZIP up the webd4201 folder that exists in your src folder (this will include your source code and the sql folder/files) DO NOT use \*.7z or \*.rar or any other compression scheme (there will be penalties otherwise). This \*.zip file should be submitted into DC Connect on or before the due date.
* ALL SQL statements for ALL labs are to be set up as prepared statements using the prepareStatement() method
* Passwords are to be hashed as “sha1”, in Java that entails setting up a

Submit complete program source code and database script files that satisfy the following requirements:

1. Create the following specialized exception classes (by extend’ing the generic Exception class). Your exceptions should overload both the default constructor (no arguments) and the overloaded one that takes a single String argument for a message)
   1. NotFoundException to be used with the retrieve() methods created below
   2. DuplicateException to be used with the insert() methods created below
2. You are to create a PostGreSQL database that runs on port 5432 (default), named webd4201\_db that is owned by a user named webd4201\_admin that has a password of webd4201\_password

NOTE: if you do not set your database up with these values exactly for testing, your program will not run on your instructor’s laptop for assessment (there will be penalties).

1. Create three separate SQL scripts to create tables named users, students, and faculty. These tables should have fields that correspond to their instance attributes created in Part 1a. The user attributes of the specialized faculty and student classes (with the exception of the id attribute) should be stored in the users table, not the students and faculty tables. The id field in all three tables should be PRIMARY KEYs

NOTES:

* All Java files are to be completely Javadoc commented (i.e. all classes, interfaces, attribute, class constants, constructor, methods, etc.)
* The \*.sql files are to be commented with your name, the date the file was created, and a brief description
* The password field for the users table should be 40 characters (the length of a sha1 hash)
* All passwords in your PostgreSQL table are to be hashed (i.e. digested as sha1)
* The student and faculty id fields should be FOREIGN KEYs of the user id (i.e. you cannot have student or faculty records unless there is a corresponding user sharing the same id).
* Create six (6) users, three (3) of type ‘s’ (i.e. students) and three (3) of type ‘f’ (i.e. faculty).
* One of the student records should correspond to your personal information
* One of the student records should create a user/student for Mike Jones (who has a password of “password” and a last access of the date the student SQL file was created)

Student Info for:

Mike Jones (100111111)

Currently in 3rd year of "Computer System Technology" (CSTY)

at Durham College

Enrolled: 11-Sep-2015

1. Create a DatabaseConnect.java class that is able to connect to the webd4201\_db PostGreSQL database created above. This class should have an *initialize()* class method that returns a Connection object and a *terminate()* class method that returns nothing but that closes the shared Connection object

NOTE: use the CustomerDatabaseConnect.java class provided in the example as a start point. Rename the class/file and modify the following information to connect to the new database:

static String *url* = "jdbc:postgresql://127.0.0.1:5432/webd4201\_db";

static Connection *aConnection*;

static String *user* = "webd4201\_admin";

static String *password* = "webd4201\_password";

NOTE: for your Java project to execute, you must add the PostGreSQL Driver JAR file (provided in DC Connect under the Software resource section) named postgresql-9.4-1200.jdbc4.jar into you project.

1. Create a StudentDA.java class that will contain the following:
   1. static attributes for a Connection object, a Statement object, a Student object, and one each for each instance attribute for a Student object (i.e. a long for an id, a String for a password, a String for a first name, …) NOTE: this is similar to the example provided.
   2. Include the following class constant. This formatter should be used to format any java.util.Date() object into a database compatible String:

private static final SimpleDateFormat SQL\_DF = new SimpleDateFormat("yyyy-MM-dd");

* 1. static methods named initialize(), that takes a Connection argument, and terminate(), that takes no arguments, that both return nothing but create a Statement object (place in the static Statement created above) and close the Statement object respectively. NOTE: this is identical to the methods in the example provided.
  2. a static method named create() that accepts a Student argument, returns a boolean, and INSERTs a new Student record based on the Student object passed, this method should throw a DuplicateException if a record with the same id already exists in the database. The method should use the retrieve() method created below to determine if a record already exists: if it does not, insert a new records; if there is an existing record (this would cause a conflict in the database), throw a DuplicateException giving the conflicting id as part of the exception message.

INSERT INTO Users (Id, Password, FirstName, LastName, EmailAddress, LastAccess, EnrolDate, Enabled,

Type) VALUES, ProgramCode, ProgramDescription, Year) VALUES (100222222, 'password', 'Robert', 'McReady', 'bob.mcready@dcmail.ca', '2016-03-07', '2015-09-03', true, 's')

INSERT INTO Students (Id, ProgramCode, ProgramDescription, Year) VALUES (100222222, 'CPA', 'Computer Programmer Analyst', 3)

* 1. a static method named retrieve() that accepts a long argument, returns a Student object based upon the id number passed, and SELECTs an existing this student record from the database based on the id argument passed. The retrieved record from the database should be used to create a Student object (that is returned by the method), the method should throw a NotFoundException if there is not a record with the id passed.

SELECT Users.Id, Password, FirstName, LastName, EmailAddress, LastAccess,

EnrolDate, Enabled, Type, ProgramCode, ProgramDescription, Year

FROM Users, Students WHERE Users.id = Students.id AND Users.Id = 100222222

* 1. a static method named update() that accepts a Student argument, return an integer (the number of records updated) and modifies an existing student using a SQL UPDATE command. The method should use the retrieve() method to determine that a record exists to be updated, the method should throw a NotFoundException if there is not record in the database with the passed object’s id. Be sure to use the executeUpdate() java.sql method in order to return the int (the number of records updated).

UPDATE Users SET Password='newpassword', FirstName= 'Robert',

LastName='McReady', EmailAddress='bob.mcready@dcmail.ca',

LastAccess='2016-03-10', EnrolDate='2015-09-03', Type='s',

Enabled=true

WHERE Id = 100222222

UPDATE Students SET ProgramCode='RPN', ProgramDescription='Registered Practical Nurse', Year=3

WHERE Id = 100222222

* 1. a static method named delete() that accepts a Student argument, return an integer (the number of records updated) and modifies an existing student using a SQL DELETE command. The method should use the retrieve() method to determine that a record exists to be deleted, the method should throw a NotFoundException if there is not record in the database with the passed object’s id. Be sure to use the executeUpdate() java.sql method in order to return the int (the number of records deleted).

DELETE FROM Students WHERE Id = 100222222

DELETE FROM Users WHERE Id = 100222222

**NOTES:**

* **Use the provided CustomerDA.java class in the provided example as a starting point. In reality, you should only have to change the field names in the SQL statements and the retrieval from the result set**
* **To facilitate debugging any new SQL statement, create the SQL statements and run them against the tables in pgAdmin, fix any errors that arise and then use the corrected statements to make the required prepared statements in Java**

1. Modify the Student.java class to have:
   1. class (i.e. static) methods named initialize() (that takes a Connection argument), terminate() (that take no arguments) and retrieve() (that takes a long argument) that call the similarly named methods created in the StudentDA.java class.
   2. Has instance (i.e. you need to be a Student object to call) methods named create(), update() and delete() (that return a boolean, an int and an int respectively). These methods should call the similarly named method in the StudentDA.java class, passing the calling object to the StudentDA method using the this keyword.
   3. For all of these classes, each should throw the same exception as its counterpart in the Student.DA.java class

1. Import the provided Java class file named Lab2Tester.java, that has a main method. If you have created your database and coded your project correctly (including the Exception messages) your output should match up with the following (take note of the exception messages). Add a try…catch block at the end of the program that retrieves your own student object and then displays it:

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Lab 2 Output \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Create a Student user to insert/delete later in the program, passing:

Student student1 = new Student(100222222L, "password", "Robert", "McReady", "bob.mcready@dcmail.ca", enrol, lastAccess, 's', true, "CPA", "Computer Programmer Analyst", 3);

Attempt to retrieve a student that does not exist (Id: 100222222)

Student with id of 100222222 not found in the database.

Attempt to retrieve a student that does exist (Id: 100111111)

Student record with id 100111111 retrieved from the database

Student Info for:

Mike Jones (100111111)

Currently in 3rd year of "Computer System Technology" (CSTY)

at Durham College

Enrolled: 11-Sep-2015

No marks on record

Attempt to insert a new student record for Robert McReady(Id: 100222222)

Student record added to the database.

Change the student object and attempt to update the student record for Robert McReady(Id: 100222222)

Student record updated in the database.

Students are encouraged to comment out the following try...catch block to verify the new record exists in pgAdmin by running the "SELECT \* FROM Students;" command

Attempt to delete the new student record for Robert McReady(Id: 100222222)

Student record with id 100222222 successfully removed from the database

Did not find student record with id 100222222