Object Oriented Programming
Toll System
CA5 Project - Stage 1
(There will be TWO stages in this Project)

Projects are normally undertaken individually but pairs are possible **subject to** approval by lecturer.

Year 2 - March 2020
BSc (Honours) in Computing in Software Development
BSc (Honours) in Computing in Games Development

Weighting: (Stages 1 & 2): 20% (of overall module mark)

Objectives

To practice the following:

- Design and implementation of an application in Java to demonstrate the use of the Collections framework
- To use regular expressions to validate input
- To design and develop a multithreaded application
- To design and develop a database-driven client-server application
- To design and implement a protocol that will allow communication over a network
- To gain experience testing the functionality of the application
- To use version control

Stage 1 - Part A (40 marks)

At a motorway toll bridge station, a camera scans the number plates of vehicles as they pass. A snapshot image of each vehicle's registration plate is taken and the image is stored using a sequence number as its filename. (e.g. "30402.jpg"). The numeric part of the filename is used as the image file id (a long integer type).

Imaging software uses Machine Learning algorithms to process the image of each vehicle registration plate and generates the registration number as a String. e.g. "201D1234". (All spaces are removed). This registration, along with the associated image file id and a timestamp is made available to the software system.

A **TollEvent** object is generated for each vehicle with three items of data:

- Vehicle Registration (String)
- Image ID (long int)
- Timestamp (Instant type object generated by *Instant.now()*]

e.g. `[“201D1234”, 30402, ‘2020-02-14 10:15:30.123Z’]`

The toll system must maintain a persistent record of these "Toll Events" in a database. The timestamp should be stored in the database using an SQL Timestamp type. This data will be later used for calculating customer bills.

Processing a Toll Event

Each time a Toll Event is generated and made available, the recording system must first check that the registration provided is valid. As this must be done quickly, vehicle registrations are stored in a 'look-up table' data structure that provides the fastest look up times. The data structure is initially populated from the registered vehicles database table. (Use the data in the "vehicles.csv" for populating your vehicles database).

If a registration is not found in the look-up table, then it should be added to a list of invalid registrations; to be dealt with later. (Registrations may be invalid due to a 'false' number plate or an imaging processing mistake, etc.).

If a registration is found in the look-up table - and thus is valid - then the system will record the Toll Event. In this system, these events are to be stored in a data structure that associates a registration with a list of Toll Events. (A registration can have multiple toll events). At some time when the system is not busy (e.g. 4am each day), all the Toll Events are batch written from the store of Toll Events to a **TollEvent** database table.

Functionality

1. Create a **menu** to support the following operations.
2. Load registration numbers from the registered **Vehicles** database table into the 'look-up table' data structure for validating registrations.
3. Process Toll Events based on the description above; validating and storing Toll Events to an appropriate data structure. (You must simulate this by reading toll events, row by row, from text file Toll-Events.csv).
4. Write the Toll Events from the data structure into the **TollEvent** database table as a batch job.

All Database access is to be implemented using Data Access Objects (DAOs), and Data Transfer Objects (DTOs) e.g. MySqlTollEventDao.

Stage 1 - Part B (40 marks)

Develop the MySqlTollEventDao DAO to provide additional functionality.

Functionality

Write appropriate DAO functions to achieve and test the following:

1. Get all Toll Event details
2. Get all Toll Event details given a vehicle registration
3. Get all Toll Event details that happened since a specified date-time
4. Get all Toll Event details that happened between
   a start date-time and a finish date-time
5. Get all Registrations that passed through the toll. Registrations must be unique and in alphabetical order)
6. Get all Toll Event details returned as type :
   Map< registration, list of toll events>

Testing and Version Control (20 marks)

Write Unit tests for each of the DAO methods listed above.

Use Version Control to help manage your work.