University of Technology, Jamaica
School of Computing and Information Technology
Advanced Programming Using JAVA
Term Project

Given: Week of February 20, 2012
Due: Week of March 26, 2012 in Lab Session

Traffic Ticketing System

The Jamaica Constabulary Force currently issues traffic tickets manually by writing tickets. The tickets after being issued are entered in batch by data entry clerks to a database (this procedure may be done months after the ticket has been issued). Offenders may also pay the fine of the ticket at the tax office months before the entry has been made. This has created a disconnection in proper record keeping for the government. You are to create a system that captures all offenders in a database and keep track of all tickets and payments.

Police Officers' Module:

The officer laying the charge will enter a first time offender into the database, and assign a ticket to him. A second time offender should already be in the database and the officer laying the charge (fine and license points deduction) should be able to pull his information and view all tickets for traffic offenses and verify if they have been paid. The officer should also be able to use the existing personal information of the offender to generate the ticket for the new offense.

Tax Offices' Module:

The tax office should be able to view all tickets for an offender with the associated traffic offenses. They should also be able to accept payment for the offense and adjust the points on the drivers' licenses.

Database Design

Police_Division - (Station#**PK**, Station_Address1, Station_Address2, Station_Tel)

Police Table – (Badge#**PK**, First_Name, Middle_Initial, Last_Name, DOB, Address1, Address2)
NB. All police officers belong to a division, please resolve foreign keys
Offender – (TRN#**PK**, First_Name, Middle_Initial, Last_Name, DOB, Address1, Address2, Type_of_License, Points, Expiry_Date)
Tickets – (Ticket#**PK**, Offense_Code, Date_of_Offense, Place_of_Offense, Descrption, Fine, Points, Payment_Status)

Offenses – (Offense_Code**PK**, Offense_Name)
NB. Resolve all foreign keys and relationships. An officer may issue several tickets; an offender may receive several tickets. Pay close attention to many-to-many relationships and 3Rd normal form normalization before building your classes

Design a JAVA application that will meet the requirements. Your application should be built using the ***Client/Server*** architecture. You are also required to provide administrative functions on the server to control access to the server as well as the database. Please take note of the fact that some tickets may be challenged in the court and therefore must be addressed appropriately by the system.
Your Database should be located on the server machine where the client will make requests to the server over the network. The server will grant the corresponding request from the client. Your application should be developed with a **graphical user interface** that will aid the users in the system for their respective data entry and processing. **Please include a GUI to interact with the data for all database tables.**

Specific Programmer Requirements

- The system should use a client/server approach
- The client should be able to add a new offender to the database
- The client should be able to update the database as appropriate for role
- The server should keep a record of all tables used in the system
- Use appropriate GUIs to facilitate ease of use of the system
- Use UML class diagrams to show all classes and their relationships
- Use ER diagrams for the database

Assessment

Marks will be awarded as follows:

- User Interface (**20%**)
- Functionality (**45%**)
- Design and use of classes (**15%**)
- User Interaction (**10%**)
- Documentation (**10%**)

Extra Credit 1 (5%)

In order to accommodate multiple users at the same time, the server thus needs to be threaded to accommodate multiple clients that will connect to it to interact with the database.

Extra Credit 2 (5%)

Your application should be able to print the traffic offense ticket in a presentable format. For this you may export the ticket to a PDF document. The marks for this section may be awarded at the discretion and satisfaction of you lab tutor. **This does not entail a screenshot of your Java GUI application form.**

Please stick to programming conventions such as **proper indentation**. Include **comments** (one to briefly describe the function and any other special/important lines).

Classes begin with **capital** letters (ex: **Person**), functions begin with **common** letters followed by **initial** caps (**ex: getName( )**)

Variables should have **meaningful** names. Refrain from using 'x' or any other single letter variables. Variables also begin with **common** letters followed by **initial** caps. Underscores are also permitted. (**ex: regStatus, date_of_birth**)

You are not limited to the database fields given above for the database tables. Feel free to add other fields or develop your own database schema.

Assignments are due on the day of the week when you have your lab. Late assignments will attract a penalty of 10% per day late. **Assignments more than five (5) days late will not collected.**