WESTERN SYDNEY UNIVERSITY

SCHOOL OF COMPUTING, ENGINEERING & MATHEMATICS
300581: Programming Techniques

Spring 2019: Programming Project

Due by 7:00pm on Friday 18th October 2019 **Assessment Weight: 40%**

A. Requirements

a) ALL instructions given in this document MUST be followed in order to be **eligible** for full marks for the assignment. This document has six (6) pages.
b) This assignment is **NOT** a group assignment; collusion, plagiarism, cheating of any kind is not acceptable. As part of your submission you MUST certify that all work submitted is your own. If you cannot honestly certify that the work is your own then do not submit the assignment. Breaches of the Misconduct Rule will be dealt with according to the university Rule (see the learning guide for more information).
c) All assignment submissions will be checked for academic misconduct by the use of the MOSS program from Stanford University. Details on MOSS can be obtained from the MOSS web site http://theory.stanford.edu/~aiken/moss/

For the problem definition described in section B you must

d) include your student id at the end of all filenames for all java code files. Two classes have been identified in section B as being required as part of your solution. Do not modify the names of these classes except for adding your student id to the end of the filename. Other Java files will be needed as part of your solution. All Java code files that are used in this assignment MUST have your student id appended to the filename. For example, Driver_#########.java;
e) include your authorship details at the top of **each** file in code comments (see item 3.1 in Section C of this document for details);
f) adhere to the coding standard as identified in the Google Java Style Guide (see Section C of this document for details);
g) ensure that standard console Input/Output are used in all code segments, **do not** use Swing;
h) ensure that your java code is appropriately modularised for the given problem definition. That is, you need to write appropriate classes and methods to solve the problem;
i) **reference** all sources that you used for inspiration of your solution as per Section D of this document;
j) Ensure that your java code compiles and runs in Eclipse installed in the SCEM labs.

B. Project Details

B(i) - Background information and description

The Police regularly patrol the roads to deter and detect dangerous driving. There are heavy penalties, including demerit points, fines and on-the-spot licence suspensions, for drivers who break the law[1]. As an unrestricted-licence driver in NSW you start with zero demerit points. If you commit an offence that has a demerit point penalty, they are then added to your driving record. If you reach or exceed the maximum demerit points allowable for your licence within a 3-year period, your licence will be suspended[2]. Unrestricted NSW driver licence holders who reach or exceed 13 demerit points are sent a Notice of Suspension, or are refused a licence if they apply to renew one[3]. Driving offences include, but are not limited to, alcohol and drug offences, heavy vehicle offences, motorcycle offences, seat belt and child restraint offences, speeding, street racing, unlicensed driving[4].

In this assignment, you will create an object-oriented, menu-driven Java program that implements a simplified range of functionalities relating to infringements for speeding offences and the resulting demerit point calculations. The Java program will retrieve data from secondary storage, store the data in memory in appropriate data structures using objects, sort and search the data, generate on-screen and file-based reports, and save data to secondary storage. The specific functional requirements for the assignment are described in section B(ii) of this document. The text files that are to be used for this assignment are described in section B(iii). The classes that **must** be used as a minimum are described in section B(iv).

[1] https://www.rms.nsw.gov.au/roads/safety-rules/demerits-offences/index.html

[2] https://www.service.nsw.gov.au/transaction/check-your-demerit-points

[3] https://www.service.nsw.gov.au/transaction/professional-drivers-demerit-points-threshold

[4] https://www.rms.nsw.gov.au/roads/safety-rules/demerits-offences/index.html

B(ii) - Program Requirements/Functionality

The Java program **must**

a) be object-oriented, utilising the classes described in section B (iv) **as a minimum**. Other classes may also be needed to solve the program requirements;

b) be menu-driven. The main menu must have the following menu items:

1. Display Drivers
2. Import Infringement File
3. Generate Suspension Report
4. Save Driver Records
5. Exit Program

c) be able to process the defined text files. The text files and their formats are described in section B (iii).

Program Start Up

When the java program starts it must perform the following file related operations:

d) Read the data from the **Driver.txt** file into computer memory into an **appropriate array of objects** (see section B (iii) for a description of the *Driver* text file and section B (iv) for a description of the *Driver* class). If the Driver file does not exist then the user should be informed of this and given the opportunity to provide an alternative filename that contains the Driver data;

After processing the Driver.txt file the program should display the main menu identified in section B(ii)(b) above.

Main Menu Item Functionality

The required functionality for each menu item is described as follows:

1. Display Drivers – when this menu option is selected the following actions should be performed by the program:

- Display an on-screen list of all drivers that were loaded into memory. The on-screen list must include the Licence Number, First Name, Last Name, Suburb, Demerit Points, and Licence Status for each driver. Ensure that the list is displayed in a clear and logical format with suitable headings.

  Extension Activity: As an extension to the stated on-screen list, provide the user with the option to display the report in descending-order of the Demerit Points.

2. Import Infringement File - when this menu option is selected the following actions should be performed by the program:

- Prompt the user for the name of the infringement file that is to be imported (see section B(iii) for an explanation about the file naming and sequencing of the infringement files).
- Import the chosen infringement file into memory
- As each infringement is read from the file, apply the penalties (demerit points, fine, licence suspension) based upon the Excess Speed, to the relevant driver licence record **in memory**. The penalties are defined by the SpeedingPenalty class which is described in section B(iv)(b). Make sure that when applying the penalties to the driver record your program considers the maximum demerit points allowed as described in section B(i).
- If a licence number in the infringement file doesn't match any of the existing driver licences held in memory then the penalties for the infringement cannot be applied and should simply be ignored.
- After processing all records in the chosen infringement file display an on-screen summary report that shows the total number of infringements correctly applied to drivers, the total revenue from fines correctly applied to drivers, the total number of licences suspended.

3. Generate Suspension Report - when this menu option is selected the following actions should be performed:

- generate an on-screen list of the driver licences that are currently suspended. The on-screen list must include the Licence Number, First Name, Last Name, Address, Suburb, Demerit Points for each suspended driver licence. Ensure that the list is displayed in a clear and logical format with suitable headings.

4. Save Driver Records – when this menu option is selected the following actions should be performed by the program:

- Write all driver licence records back to the driver.txt text file. Ensure that the format that is used to write back to the file matches the original file format as identified in section B(iii). By doing so, the newly created driver.txt file can now be used as the new input file for the next time the program is run.

5. Exit Program – when this menu option is selected the program must terminate. **The program should not terminate until this option is chosen**. If the driver record data has changed since the last save operation, then do not exit the program without first warning the user that changes will be lost if program exit is continued. Ask the user if they wish to continue to exit the program (without saving) or to cancel the exit. If the user chooses to cancel the exit, return program control to the main menu. If the user chooses to continue with the exit, allow the program to terminate without saving the driver file.

B(iii) - Text files to be processed

The data that is to be manipulated by your Java program for this assignment is contained in the text files **driver.txt,** and **infringements#.txt.** Examples of these text files are found in the zip file for the assignment. The data within these text files will need to be read into memory by your program so that it may be manipulated to solve many aspects of the required functionality of the assignment. The text files have been created to conform to a particular format. The format for each file is described below:

File: driver.txt

This file contains a full record of unrestricted NSW driver licence holders. Each line within the file represents an individual driver, and has the following format:

Licence Number, Licence Class, First Name, Last Name, Address, Suburb, Postcode, Demerit Points, Licence Status

where each data item is separated by a comma (,). The following table provides a brief explanation of each of these data items:

Data Item	Explanation	Extra Notes
Licence Number	the driver's licence number	
Licence Class	the class of licence	Possible values are C, R, LR, MR, HR, HC, MC[5]
First Name	the driver's first name	
Last Name	the driver's surname	
Address	the driver's home address	
Suburb	the driver's home suburb	
Postcode	the driver's home postcode	
Demerit Points	the driver's current accumulated demerit points	refer to section B(i) for description of how demerit points work in NSW
Licence Status	the status of the driver's licence	values are either "Valid" or "Suspended"

File: infringments#.txt

There are multiple infringement files which are numbered in sequence. These files contain a record of infringements for drivers that have committed a speeding offence that has a demerit point penalty. Each line within the files represent an individual infringement for a driver. A driver may possibly receive more than one infringement if they have been booked for speeding more than once. If a driver has not received an infringement then there will not be any infringements in the files (ie, not all drivers will appear in the files). Each infringement file has the following format:

Infringement Number, Licence Number, Date of Infringement, Excess Speed

where each data item is separated by a comma (,). The following table provides a brief explanation of each of these data items:

Data Item	Explanation	Extra Notes
Infringement Number	a unique numeric identifier for the infringement	
Licence Number	the drivers licence number	The licence number may not match any of the drivers in the driver records. See note 2 below for further detail
Date of Infringement	the date when the infringement occurred	
Excess Speed	the number of km/h above the speed limit	

Multiple infringements.txt files will be provided over time in vUWS. Each file represents a different set of infringements and can be applied to the driver record sequentially. That is, infringements1.txt corresponds to infringements that were issued in period 1, infringements2.txt corresponds to infringements that were issued in period 2, infringements3.txt corresponds

[5] https://www.rms.nsw.gov.au/roads/licence/driver/licence-classes.html

to infringements that were issued in period 3, and so on. These three files would be applied to the driver records separately but in the sequence 1, 2, 3 based upon the program operator's choice of file to process.

Notes: 1. As indicated in Program Start Up, when reading the Driver text file into memory the data should be read into **appropriate** array of objects. The classes for these objects are briefly outlined in section B(iv).
2. As indicated in the table above, the licence number in the infringement file may not match any of the drivers in the loaded driver records. This may occur for different reasons such as the driver holding a non-NSW licence, or an error in recording of the licence number.
3. We will use different sets of data for marking; the number of lines of data, and the data values in the text files will be different from those supplied with the assignment. This is to ensure that your solution has not relied upon specific data values or the number of lines in the text files to work. You should therefore test your program with different data files of varying length and varying data before submission.

B(iv) - Required Classes

To write your solution for this assignment it is a **requirement** that you write appropriate code for **at least** the following java Classes:

a) Driver
b) SpeedingPenalty

These classes are described in general terms as follows:

a) **Driver class**: The Driver class represents *an individual* NSW licence holder (driver). The Driver class needs data fields for
   - the Licence Number of the driver (a string)
   - Licence Class of the driver (a string)
   - First name of the driver (a string)
   - Last name of the driver (a string)
   - Address of the driver (a string)
   - Suburb of the driver (a string)
   - Demerit Points (a whole number)
   - Licence Status (a string)

   The Driver class needs ***appropriate*** constructors, accessors, and mutators ***where necessary*** and other **appropriate** methods based upon the general requirements of the assignment specification – that is, you will need to identify if the Driver class is required to perform any other actions and implement the necessary methods in the class.

b) **SpeedingPenalty class**: The SpeedingPenalty class represents the different penalties that are applicable for speeding offences in NSW for unrestricted licence drivers (note: this class does not cover all possible speeding offences for all types of driver types (e.g., not Learner or Provisional licence holders, etc)). A partially complete SpeedingPenalty class has been provided in the assignment zip file. The class contains data identifying the Excess Speed, the corresponding Demerit Point penalty, corresponding monetary fine, and whether the penalty includes automatic suspension of licence.
   The data fields provided in the SpeedingPenalty class are static and are:
   - ExcessSpeed – an array of excess speed values that represent the **boundary conditions** for applying penalties for speeding offences (this is further explained below in the table and by examples)
   - DemeritPoints – an array of demerit points applicable for the different speed excesses
   - Fine – an array of monetary fines applicable for the different speed excesses
   - LicenceSuspension – an array indicating whether **automatic** suspension applies for the different speed excesses

   The following is a tabular representation of the data in the four arrays in the SpeedingPenalty class that may assist you to understand the relationship between the arrays and the data:

Excess Speed (km/h)	**Demerit Point Penalty**	**Fine**	**Automatic Licence Suspension**
Exceed speed limit by **more than** 0 km/h	1	$360.00	No (false)
Exceed speed limit by **more than** 10 km/h	3	$481.00	No (false)
Exceed speed limit by **more than** 20 km/h	4	$599.00	No (false)
Exceed speed limit by **more than** 30 km/h	5	$1441.00	Yes (true)
Exceed speed limit by **more than** 45 km/h	6	$3762.00	Yes (true)

   Each of the arrays in the class have been initialised with the values shown in the table above; **do not change the arrays in any way**. Each of the arrays run in parallel to the others in the class. That is, if we know the excess

speed for an infringement then we can determine the demerit points, fine, and whether the licence is automatically suspended by looking at the **corresponding element** in the other arrays. The following examples illustrate this idea.

Example 1: if a driver is booked driving at 10 km/h over the speed limit then the penalties would be loss of 1 demerit point and a fine of $360.
Example 2: If a driver was booked driving at 22 km/h over the speed limit then the penalties would be loss of 4 demerit points and a fine of $599.
Example 3: If a driver was booked driving at 46 km/h over the speed limit then the penalties would be loss of 6 demerit points, a fine of $3762 and automatic suspension of licence.

Since the SpeedingPenalty class is meant to be a utility class and its data fields and methods are static, constructors and mutators are not needed. However, you will need to implement appropriate accessors **where necessary** and other **appropriate** static methods for this class based upon the general requirements of the assignment specification. A method called **findPenaltyIndex** has been supplied in the class; **do not change this method in anyway**. This method finds the index of the Excess Speed value in the ExcessSpeed array. This value can then be used by other methods to determine the corresponding penalties for the infringement.

Apart from the two classes identified above it is possible that you **may also need** to write other classes depending upon your solution method.

C. Google Java Style Guide

The submission in this assignment must adhere to the following listed coding standards as defined in the Google Java Style Guide that is found at https://google.github.io/styleguide/javaguide.html

Style Guide Item Number	Changes to	Modification
2.1 to 2.3	**2.1 modified**	2.1 - File Name: The source file name consists of the case-sensitive name of the top-level class it contains as identified in the question, plus an underscore, plus the student ID, plus the .java extension Example: Driver_12345678.java
3.1 to 3.4.1	**3.1 modified**	3.1 – License Info is replaced by **Authorship** information All java source files must contain the authorship information as follows: **Student ID:** **Name:** **Campus:** **Tutor Name:** **Class Day:** **Class Time:**
4.1 to 4.7, 4.8.2.1 to 4.8.2.3, 4.8.4 to 4.8.4.3, 4.8.6	**NIL**	
5, 5.1 to 5.3	**NIL**	

D. Referencing

Referencing must follow the guidelines given on page 8 in Section 2.5.1 of the unit Learning Guide. An example implementation of this referencing style can be found in the FAQ in the Programming Techniques vUWS site.

E. Project Submission Procedure

To submit your **assignment** you must do the following by the due date and time specified on page 1 of this document:

1. Create a zip file which contains
    a. your **complete Java project** including the Java source code file(s).

    Note: The zip file must be named according to the naming convention

    *studentid*_*StudentName*_300581_Project_*TutorsName*.zip

    where *studentid* is your student id, and *StudentName* is your full name, *TutorsName* is the name of your tutor.
2. Upload the above zip file in vUWS in the **Project Submission** link provided.

F. Marking Criteria and Standards

The marking criteria and standards for the assignment are published on **pages 11-14 of section 2.5.2 in the Learning Guide** and will be used to assess your assignment submission according to the specific weightings identified in the table below

Code Functionality/Correctness:	55%	Code Documentation:	5%
Class Construction	15%	Identifier Use:	5%
Algorithm Selection:	15%	Code Readability:	5%