



Traffic Violation System

Assignment 2

CPCS203 Programming-II **Term**

Assigned Date: 11/10

Delivery Date: 31/10

Instructions

- This program must ONLY be submitted on the Blackboard!
- This project worth 6% of the overall module marks (100%).
- NO assignment will be accepted after 11:59 pm for any reason
- Students can submit their assignment between 11 and 11:59 PM but in this case it will be consider as late submission.
- For discussion schedule, check the captain name, date and time on the BlackBoard.
- *Further information is provided in the course syllabus.*

Objectives

- Performing procedure on Objects and classes.
- Learn how to use and implement Class and Object concepts.
- Learn to use File I/O (Reading/Writing from/to files).

How to submit your assignment?

- Submit your assignment on the Blackboard ONLY.
- Make sure to add your names / IDs / Section / Your name / Assignment number at the beginning of your program

Files provided with assignment

- Input file samples:
 - **inputTraffic.txt**: which contains all Car, Violation, and Driver details that needs to be registered into the system.
 - **inputViolations.txt**: contains all the commands to issue traffic violation tickets. These commands are read from the file and processed by the system.
- Output files:
 - **TrafficDB.txt**: This output file displays all the registered record for the cars, violations and driver (The information in this file is read from **inputTraffic.txt**).
 - **TrafficTickets.txt**: This output file contains a log of all issued traffic violation tickets with full details. This file also have a report at the end to show total number of violations against each registered driver in the system.
 -

Note: Please check the format of each of these files and make sure you follow this format in your assignment solution.

1.1 The Traffic Violation System Description

The previous project Vehicle Mileage Guide, that you had successfully implemented helped the consumers in choosing fuel efficient vehicles. However, this results in a new issue for the traffic department to handle the unexpected increase in traffic volume. You are now required to develop a software system for the traffic department to register all the vehicles, drivers, and violation details. This information will be used while issuing traffic violation ticket.

The system you are required to develop is called **Traffic Violation System** and is expected to assist the traffic officer in issuing violation ticket. At the beginning, your system will register all the available vehicle, driver and violation details from **inputTraffic.txt**. Information read from **inputTraffic.txt** are written with all the details into the vehicle information file, called **TrafficDB.txt**.

After vehicle registration, the system will be ready to issue violation ticket for any registered vehicle. Since the data is already stored in the system, the traffic officer will capture minimum data while issuing a ticket. For each violation ticket, the vehicle's number plate, violator's national ID, violation code, violation location, violation date, vehicle insurance status and fahas status is read from the input file **inputViolation.txt**. After processing each violation ticket, the details is written to the output file **TrafficTickets.txt**. Ticket will include all the details and violation amount will be calculated as per violation type. Additional penalty will be added automatically in case of expired vehicle insurance, invalid fahas, and under age driver.

For a more detailed description of the system and commands, please follow the next three steps which will explain how to develop the **Traffic Violation System**.

Step 1: Registering Vehicle, Driver and Violation Details

All vehicles, driver and violation details will be registered into the system before issuing any violation ticket. The first line read from **inputViolation.txt** contains three integers, which determine the number of registered vehicles, drivers, and violations. For example, the total number of registered vehicles is (20), the total number of drivers is (10), and the total number of violations is (10). In the following, we describe the format of each command.

1.1 Command: AddVehicle

This command is used to add all information of the registered vehicles. Vehicle information includes plate number (such as ANG-3456), vehicle type (such as car, CUV, pickup, bus etc), vehicle brand (such as Toyota, Mercedes, Nissan etc), model name (such as Corolla, Civic, Fortuner etc), color and manufacture year. Check the following example and table.

Command Example
AddVehicle ANG-3456 Car Toyota Corolla White 2017

Field name	Type	Example
Number Plate	String	ANG-3456
Type	String	Car
Brand	String	Toyota
Model	String	Corolla
Color	String	Whiie
Year	int	2017

1.2 Command: AddDriver

This command will add driver details into the system.

Command Example
AddDriver 2021236524 Ahmed Shamrani 1989 8 11

Field name	Type	Example
National ID	String	2021236524
First Name	String	Ahmed
Last Name	String	Shamrani
Date of Birth	Date	1989 8 11

1.3 Command: AddViolation

This command will add violation details into the system.

Command Example
AddViolation 1000 Red_Light 3000

Field name	Type	Example
Violation Code	String	1000
Violation Description	String	Red_Light
Penalty Amount	double	3000

1.4 Command: Quit

The command quit will exit the process of entering the registration information.

Step 2: Issue Violation Ticket

The ticket details are provided in *InputViolation.txt*. The first line of this file is an integer that determines the number of tickets to be processed. For example, in the provided file *InputViolation.txt*, 20 tickets need to be processed. In the following, a more detailed description of issuing violation tickets is provided below.

2.1 Command: IssueTicket

This command is used to issue a violation ticket. This command contains the basic information of vehicle, driver and violation. For each ticket, it reads the violation code, vehicle number plate, driver's National ID number, location (city name), date, vehicle's insurance a fahas satus.

Command Example
IssueTicket 1000 OEN-4432 2342114502 Jeddah 2020 3 15 true false

Field name	Type	Example
Violation code	String	1000
Number plate	String	OEN-4432
Violator's National ID	String	2342114502
Location	String	Jeddah
Date	Date	2020 3 15
Has Valid Insurance	Boolean	true
Has Valid Fahas	Boolean	false

Consider the following notes when issuing tickets:

Important Notes
<ul style="list-style-type: none">The system will read the Violation Code as a string. You need to search for the violation object associated with the given violation code.
<ul style="list-style-type: none">The system will read the Number Plate as a string. You need to search for the vehicle object associated with the given number plate.
<ul style="list-style-type: none">The system will read the Violator's National ID as a string. You need to search for the driver object associated with the given national ID.
<ul style="list-style-type: none">For each ticket, the system should generate a unique 13 digit time stamped ticket number (Hint: Use System.currentTimeMillis() to generate it). Since it is a time stamped number, it will be different each time you run the program.

This ticket number will be printed in the *TrafficTickets.txt* (check step 3).

- **You must calculate total penalty amount**

There are three types of additional penalty that will be added to the actual violation amount.

1. Invalid Insurance
Add additional penalty of 100 SAR
2. Invalid Fahas
Add additional penalty of 150 SAR
3. Under Age
Add additional penalty of 1000 SAR if driver age is less than 22 years (Use driver's date of birth to calculate age)

1.2 Command: Quit

The command quit will exit the process of entering the ticket information.

Step 3: Print all the information

3.1 Print the registered vehicle, driver, and violation information

As mentioned earlier, the registered vehicle, driver, and violation information are read from **inputTraffic.txt** and are written to the file **TrafficDB.txt**.

3.2 Print all issued tickets and report of total violations by each driver.

The system should print a log of all processed violation tickets. The system will calculate the total penalty amount for each ticket. For example, in step 2, once the system process the following command from the *InputViolations.txt*, the following information presented in the table below is written to the file *TrafficTickets.txt*:

Command Example

IssueTicket 1000 OEN-4432 2342114502 Jeddah 2020 3 15 true false
--

Ticket details

Ticket No. 1601690726530

Violation Details

Violation Code: 1000












<p>Violation Description: Red Light</p> <p>Violation Penalty: 3000.0</p> <p>Vehicle Details</p> <p>Number Plate: OEN-4432</p> <p>Type: Car</p> <p>Brand: Lexus</p> <p>Model: LX570</p> <p>Color: White</p> <p>Built Year: 2011</p> <p>Violator Details</p> <p>National ID: 2342114502</p> <p>Full Name: Jamal Albara</p> <p>Ticket Details</p> <p>Date: 2020-4-3</p> <p>Location: Jeddah</p> <p>Total Penalty: 3150.0</p>















After printing all tickets, the system will print a report showing a list of all drivers and the number of violation by each driver as shown below:

Number Of Violations by Driver		
-----Total violation(s) by driver-----		
Driver ID	Driver Name	Total Violation(s)
2021236524	Ahmed Shamrani	0
2342114502	Jamal Albara	3
2225198408	Abdul Rehman	6
2474747448	Qasim Nasr	1
2908967685	Sahl Zahrani	0
2345347676	Abdul Rehman	2
2546834724	Malick Ghazi	3
2912367457	Bandar Omar	3
2134367222	Khaled Bara	0
2845898568	Hamid Farouk	2

1.2 UML Class Diagram for EasyRent














In addition to the main class, you should create four classes as shown in the following UML diagram. Note that you should write appropriate constructor, setter, and getter methods for all classes. (You don't need to follow the same given arguments). Be aware of the visibility (public-private) for each attribute/method.

 Violation
 - int violationCode  - String description  - double amount
 + Violation(int violationCode, String description, double amount)  + int getViolationCode()  + void setViolationCode(int violationCode)  + String getDescription()  + void setDescription(String description)  + double getAmount()  + void setAmount(double amount)

 Driver
 - String nationalID  - String First_name  - String Last_name  - Date dob
 + Driver(String nationalID, String First_name, String Last_name, Date dob)  + String getNationalID()  + void setNationalID(String nationalID)  + String getFirst_name()  + void setFirst_name(String First_name)  + String getLast_name()  + void setLast_name(String Last_name)  + Date getDob()  + void setDob(Date dob)

Vehicle

-  - String VehiclePlateNo
-  - String VehicleType
-  - String Brand
-  - String VehicleModel
-  - String VehicleColor
-  - int BuiltYear

-  + Vehicle(String plateNo, String VehicleType, String Brand, String VehicleModel, String VehicleColor, int BuiltYear)
-  + String getVehiclePlateNo()
-  + void setVehiclePlateNo(String VehiclePlateNo)
-  + String getBrand()
-  + void setBrand(String Brand)
-  + String getVehicleType()
-  + void setVehicleType(String VehicleType)
-  + String getVehicleModel()
-  + void setVehicleModel(String VehicleModel)
-  + String getVehicleColor()
-  + void setVehicleColor(String VehicleColor)
-  + int getBuiltYear()
-  + void setBuiltYear(int BuiltYear)

Ticket

 - long ticketNo
 - Violation violation
 - Vehicle vehicle
 - Driver violator
 ~String location
 ~Date violationDate
 ~Boolean hasValidInsurance
 ~Boolean hasValidFahas

◆ +Ticket(long ticketNo, Violation violation, Vehicle vehicle, Driver violator, String location, Date violationDate, Boolean hasValidInsurance, Boolean hasValidFahas)
● +long getTicketNo()
● +void setTicketNo(long ticketNo)
● +Violation getViolation()
● +void setViolation(Violation violation)
● +Vehicle getVehicle()
● +void setVehicle(Vehicle vehicle)
● +Driver getViolator()
● +void setViolator(Driver violator)
● +String getLocation()
● +void setLocation(String location)
● +Date getViolationDate()
● +void setViolationDate(Date violationDate)
● +Boolean getHasValidInsurance()
● +void setHasValidInsurance(Boolean hasValidInsurance)
● +Boolean getHasValidFahas()
● +void setHasValidFahas(Boolean hasValidFahas)
● +double CalculateFinalFine()

IssueTicket

- + static void main(String[] args)
- - static Vehicle getVehicle(Scanner input)
- - static Violation getViolation(Scanner input)
- - static Driver getDriver(Scanner input)
- - static Ticket GenerateTicket(Scanner input, Violation[] listViolation, Vehicle[] listVehicle, Driver[] listDriver, PrintWriter fWrite)
- + static void PrintTicket(Ticket tempticket, PrintWriter fWrite)
- + static void NumOfViolationsperDriver(Driver[] alldrivers, Ticket[] alltickets, PrintWriter fWrite)

Important Notes:

- Use of class & object, arrays of Object, and passing object to method
- Use of Files, Reading/Writing from/on files
- Your program output must be exactly same as given sample output files.
- Your display should be in a readable form.
- Organize your code in separated methods.
- Document your code with comments.
- Use meaningful variables.
- Use dash lines between each method.
- **Delayed submission will not be accepted and there will not be any extension of the project.**

Deliverables:

- You should submit one zip file containing all java codes:
BA1887415P2_EasyRent.java where BA is your section, 1887415 your ID and P2 is program 2.
- **NOTE: your name, ID, and section number should be included as comments in all files!**

Input and Output Format

Your program must generate output in a similar format to the sample run provided.

Sample input: See sample input file.

Sample output: See sample output files.

Good Luck!