Traffic control record system

Assignment brief:

Time 3 hours

You have been given some code that is currently being developed as part of a **Traffic Speed control** record application. Not all the requirements have been implemented. It is your task to implement these and raise the coding standards of all the code.

The data relates to vehicles monitored on **20-mile** stretch of road (fictitious) outside of Belfast. The road has two sensors, one that records the license plate number and time a vehicle enters the road and one at the other end that records when the vehicle exits. The average allowable speed for Cars is 60 mph whereas HGV is 50 mph. The road is only open between 06:00 and 22:00 per day and a vehicle may only enter the road once a day.

Create a **project solution** (named **<Your Name><Student Number>p3** **e.g. McGowanAidan3048614p3).** Create a package named **p3**. Add **StartApp.java** to the solution and the **traffic.csv. Ensure your name and student number are placed in the Javadoc comments of all the classes you create.** The StartApp has been partially written with a menu.

The application will run (start) from the StartApp.java, initially reading the data from the **traffic.csv** file and then performing a number of menu-driven operations.

Part 1 – Data mapping, storage and reading from file - 50%

Using your knowledge of OOP you should add/update the code based on the following:

1. Analyse the data in the **traffic.csv** and create a class (**Vehicle.java**). The file contains data on the Vehicle, including the time in and out of the road, type of vehicle, and owner information.

2. Create a simple unit test for the **Vehicle** class. No other validation or business rules are required so simple verification of fields being set by setters and constructors is sufficient.

3. In the **StartApp.java** class read and store the data (**traffic.csv**) in an appropriate JCF container. Some of the fields for some records may be empty, null or not have the expected data types. Any **entry time (In)** recorded before 06:00 or after 22:00 should be ignored. These records should not be included in the list of **Vehicles** in the JCF container. It has been recommended your code should attempt to identify these issues when reading the data (within the StartApp.java) as your Vehicle class will not have business rules included.

   When the data read is complete output the number of record attempts and number successfully read. e.g.

   ```
   Attempted to read vehicles data: 119
   Vehicle data read successfully: 100
   ```

[CONTINUED OVER]

Part 2 – Functions – 50%

Having read the data from the csv file complete the menu-driven functions as outlined below. An example of the expected format is shown for each function.

```
1. Display all vehicle data
2. Display all HGVs
3. Display vehicles and average speeds
4. Analyse and display driver age categories
5. Pens - Generate individual penalty notices (new thread needed)
6. Quit
Enter option ...
```

Note after each option 1-5 is complete the menu should be redisplayed on screen. Option 6 should end the application.

1. **Display all vehicle data**. Example output…

```
Enter option ...
1
All vehicle data
All Vehicles
License      : 22-M-9012
Type         : HGV
Time In      : 06:35
Time Out     : 07:04
First name   : Sinead
Last name    : Murphy
Age          : 39
Town         : Galway
email        : sineadmurphy@gmail.com
```
etc…

2. **Display all HGVs**. (sorted by last name) Example output…

```
Enter option ...
2
All HGVs

License      : AHY 3456
Type         : HGV
Time In      : 15:28
Time Out     : 16:00
First name   : Holly
Last name    : Armstrong
Age          : 49
Town         : Belfast
email        : hollyarmstrong71@hotmail.com
```

[CONTINUED OVER]

```
License      : 13-O-7890
Type         : HGV
Time In      : 09:00
Time Out     : 09:45
First name   : Aisling
Last name    : Brennan
Age          : 25
Town         : Limerick
email        : aislingbrennan@gmail.com
```

etc…

3. **Display the vehicle license and average speed**. Example output…

```
Enter option ...
3
License : 22-M-9012. Average Speed : 41 mph
License : 98-G-4567. Average Speed : 120 mph
License : 19-KE-6789. Average Speed : 109 mph
License : LPM 4567. Average Speed : 80 mph
```

The speed is calculated using **speed = distance / time**. Where the distance is **20 miles**. The time is the difference between the time in and time out of the road (note shown below this would be in minutes). The output should be in miles per hour as a whole number – rounded down.

Useful code :

```
import java.time.Duration;
import java.time.LocalTime;


// time representation using LocalTime objects
LocalTime timeIn = LocalTime.parse(TimeInAsAString);
LocalTime timeOut = LocalTime.parse(TimeOutAsAString);

// calculate the difference between the times
Duration duration = Duration.between(timeIn, timeOut);

// get the difference in minutes
long diffInMinutes = duration.toMinutes();
```

[CONTINUED OVER]

4. Group and display the age category of drivers, i.e. frequency of drivers within each category. Where age categories are :

   18 – 29 NEW
   30 – 50 OLDER
   50+ SENIOR

```
Enter option ...
4
NEW          : ***************************** (29)
OLDER        : **************************** (28)
SENIOR       : ******************************************* (43)
```

   Ordered by age range, as shown above (NEW, then OLDER then SENIOR).

5. Generate individual penalty notices. The speed limit for Cars is 60 mph whereas HGV is 50 mph. Each vehicle that is recorded at an average speed in excess of these limits should result in the system creating an individual personalised .txt file. Each .txt file should be titled <Lastname>_pen.txt and contain a layout similar to this example. Eg **Collins_pen.txt**

```
99-MH-8901
CAR

Dear MATTHEW COLLINS
You have been recorded travelling at an average speed of 109 mph.
This has resulted in a fixed penalty point notice.

Depart of Road Traffic Offences
```

This operation to create the .txt files should be in a **new separate thread**.

   Example output…

```
5. Pens - Generate individual penalty notices (new thread needed)

Generating pen notices (in new Thread).
```

When complete compress (zip) the entire ***Eclipse solution*** and upload to **Assignments** (P3 assessment) on CANVAS.

Now : check the uploads to ensure you have submitted the correct files (in the correct area).

[END]