CarTrawler Java Assessment

1.0 Introduction

The idea is to build on a simple Java application that currently lists c. 300 CarResult objects.

Each CarResult contains the following info:

- Supplier : the name of the CarAgent – e.g. Hertz, Budget, Avis.
- Description : make and model of car – Volkswagen Polo
- SIPP code : a 4 letter code expressing the type of car (ECMR = economy, MCMR = mini)
- Rental Cost : final cost of the car to the customer.
- Fuel Policy : Indicates if car is supplied full or empty and whether to be returned full or empty.

2.0 Objective

Using the supplied code as a starting point, the goal is to extend as follows:

- Remove any duplicates from the list (duplicates = same make, model, supplier, SIPP, FuelPolicy)
- Sort the list so that all corporate cars appear at the top. Note corporate cars are those supplied by AVIS, BUDGET, ENTERPRISE, FIREFLY, HERTZ, SIXT, THRIFTY.
- Within both the corporate and non-corporate groups, sort the cars into “mini”, “economy”, “compact” and “other” based on SIPP beginning with M, E, C respectively.
- Within each group sort low-to-high on price.
- Render the sorted list at the end of the process.

The outcome should look something along the lines of the following (note that data below is not from the list bundled with the app).

SIXT	Peugeot 107	MCMR	22.50	FULLFULL
AVIS	Peugeot 107	MCMR	30.05	FULLFULL
SIXT	Opel Corsa	EDMN	29.50	FULLFULL
FIREFLY	Volkswagen Golf	CDMR	48.00	FULLFULL
AVIS	Mercedes A Class	ICAV	80.00	FULLFULL
DELPASO	Volkswagen Up	MDMR	8.00	FULLEMPTY
DELPASO	Volkswagen Polo	EDMR	10.00	FULLFULL
GOLDCAR	Volkswagen Golf	CLMR	18.00	FULLEMPTY
GOLDCAR	Citroen Berlingo	CMMV	28.00	FULLFULL
RECORD	Ford Galaxy	FVAR	65.00	FULLEMPTY

3.0 Additionally

If time allows, include an optional step to remove all FuelType.FULLFULL cars that are priced above the median price within their groups. Hint: Google “Median Price”.

4.0 What are we looking for…

Once you’re done, send a ZIP file of the solution to careers@cartrawler.com. We’re hoping to see some / all of the following:

- Use of Ant, Maven, Gradle or similar to build the application.
- Simple, elegant, efficient code that solves the problem and prints the list.
- Demonstration of basic Java design patterns where appropriate.
- One or two sample unit tests.
- The ability to critique our baseline app as well as your own code at the next meeting.