We are delighted that you are considering joining us at Rentalcars.com, as part of our recruitment day we will be looking at your approach to problem solving and coding. With this in mind please could you complete the following exercise.

Please do not over engineer the solution so keep it simple, the things we are looking for are:

- High quality code

- Your test approach

- Problem solving

Please submit your code to us through sharing an open GitHub repository at least 48 hours before your interview.

During your interview we will ask you to take us through your code and technical solution so please do make sure your code compiles and works.

**Part 1**

Create a standalone Java console application, with the functionality to read the specified JSON file, parse and then perform the set tasks below on the parsed data. All tasks should be printed out to the console using System.out. You are free to use any external libraries, please include these JARs when you submit your work.

JSON file is accessible via <http://www.rentalcars.com/js/vehicles.json>

* Print a list of all the cars, in ascending price order, in the following format:

1. {Vehicle name} – {Price}
2. {Vehicle name} – {Price}

.

.

.

.

1. {Vehicle name} – {Price}

* Using the table below, calculate the specification of the vehicles based on their SIPP. Print the specification out using the following format:

1. {Vehicle name} – {SIPP} – {Car type} – {Car type/doors} – {Transmission} – {Fuel} – {Air con}
2. {Vehicle name} – {SIPP} – {Car type} – {Car type/doors} – {Transmission} – {Fuel} – {Air con}

.

.

.

1. {Vehicle name} – {SIPP} – {Car type} – {Car type/doors} – {Transmission} – {Fuel} – {Air con}

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 1st letter (car type) | | 2nd letter (doors/car type) | | 3rd letter (transmission) | | 4th letter (Fuel/Air Con) | |
| Letter | Value | Letter | Value | Letter | Value | Letter | Value |
| M | Mini | B | 2 doors | M | Manual | N | Petrol/no AC |
| E | Economy | C | 4 doors | A | Automatic | R | Petrol/AC |
| C | Compact | D | 5 doors |  |  |  |  |
| I | Intermediate | W | Estate |  |  |  |  |
| S | Standard | T | Convertible |  |  |  |  |
| F | Full size | F | SUV |  |  |  |  |
| P | Premium | P | Pick up |  |  |  |  |
| L | Luxury | V | Passenger Van |  |  |  |  |
| X | Special |  |  |  |  |  |  |

Examples:

ECMN – Economy, 4 door, manual, petrol, no air conditioning

FVAR – Full size passenger van, automatic, petrol, air conditioning

* Print out the highest rated supplier per car type, descending order, in the following format:

1. {Vehicle name} – {Car type} – {Supplier} – {Rating}
2. {Vehicle name} – {Car type} – {Supplier} – {Rating}

.

.

.

.

1. {Vehicle name} – {Car type} – {Supplier} – {Rating}

* Give each vehicle a score based on the below breakdown, then combine this score with the suppliers rating. Print out a list of vehicles, ordered by the sum of the scores in descending order, in the following format:

1. {Vehicle name} – {Vehicle score} – {Supplier rating} – {Sum of scores}
2. {Vehicle name} – {Vehicle score} – {Supplier rating} – {Sum of scores}

.

.

.

.

1. {Vehicle name} – {Vehicle score} – {Supplier rating} – {Sum of scores}

Breakdown

Manual transmission – 1 point

Automatic transmission – 5 points

Air conditioned – 2 points

**Part 2**

Using the code created in Part 1 extend the code to provide the output through a REST API using one of the following languages:

• PHP

• JavaScript (Node, Angular are also acceptable)

• Java

• Scala