

**King Abdulaziz University**  
**Faculty of Computing and Information Technology**  
**Semester: Fall 2020 (First semester- 1441-1442)**

|  |  |
| --- | --- |
| **Course Code: CPCS 203** | **Course Name: Programming II** |

**Assignment #1 (Vehicle Mileage Guide)**

|  |  |
| --- | --- |
| **Assigned Date** | Sunday 20/09/2020 |
| **Delivery Date and Time** | Saturday 10/10/2020 |

**WARNING**:

* 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 reasons.
* Student can submit their assignments between 11 and 11:59 PM but in this case it will be consider as late submission, and they will lose 2 points from the total mark of the assignment.
* For discussion schedule, check the teacher name, date and time on the blackboard. **Further information is provided in the course syllabus.**

**Objectives:**

* Learn how to use and implement multidimensional array.
* Performing procedure on array elements.
* Learn to use and implement String, File I/O (Reading/Writing from/to files).

**Description**

Due to increasing gasoline prices globally, the government want to promote fuel efficient vehicles. For this reason, they want to develop a system to guide the consumers about the vehicle fuel efficiency. This program is a Vehicle Fuel Efficiency Guide to assist consumers about vehicles’ general information (type, manufacturer, model, year and average mileage). In addition, the program can perform certain statistical operations on the vehicles’ mileage (average, maximum and minimum). Figure 1 displays a sample data on which this program performs operations.

**Note:** Please refer to Figure 2 for complete input list.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Vehicle Type | Manufacturer | Model | Engine Power | Year | Mileage (KM/L) |
| Hatchback | Toyota | Yaris | 1.5 | 2017 | 16.8 |
| Vitz | 1.0 | 2018 | 20.2 |
| KIA | Rio | 1.6 | 2019 | 19.2 |
| Picanto | 1.0 | 2019 | 23.4 |
| Hyundai | i10 | 1.5 | 2018 | 21.8 |
| Sedan | Nissan | Altima | 2.0 | 2016 | 13.8 |
| Maxima | 2.4 | 2016 | 12.5 |
| Chevrolet | Impala | 2.4 | 2018 | 13.8 |
| SUV | Ford | Explorer | 2.3 | 2017 | 14.2 |
| Jeep | Wrangler | 3.0 | 2018 | 9.3 |

Figure 1 Sample vehicle info

As shown in Table 1, you are required to make different arrays to store all the data used in the program. For example, you need a one-dimension array to store vehicle type. Each vehicle type has at least one manufacturer, therefore you need a two-dimension array to store the manufacturer names of each type. Similarly, each manufacturer has different models. The program stores the vehicle model and year in one column and mileage in another column. So, you need 2 three-dimension arrays: the first one to store the model and year as follows: **Yaris\_1.5\_2017**, while the second one to store the mileage.

**Basic Requirements**

* The program must read the data from a text file called “input.txt” that follows a specific pattern. If the file doesn’t exist, print a message to let the user know what happened.– see **Input File Pattern** section for more details.
* The program must generate a text file as the output called “output.txt” that contains the results of the commands written in the input file – see **Output File Pattern** section for more details.
* The program must load and deal with data dynamically. This means you should never write the data as hard codes. You should use loops to deal with the arrays.
* The program must follow the format of the output file as possible. So, you have to reformat strings. For example: Yaris\_1.5\_2017. You need to know how to extract and separately print the model (Yaris), engine capacity (1.5) and year (2017). For this you can use a method called “split” in the string object.

**Input File Pattern**

The input file called ‘input.txt’ is attached with this document. The pattern, which the input file follows, is explained in the following input file example:

3 //The first line in the input file: the number of the vehicle type stored in this file. In other words, the size of vehicle type array.

3 2 2 //The second line in the input file:

the number of manufacturer for each type is separated with a space. Ex: the first type has 3 manufacturers, the second type has 1 manufacturer, and the third type has 2 manufacturers.

In other words, manufacturers[0] = new String[3]; manufacturers[1] = new String[2]; manufacturers[2] = new String[2];

**add\_vehicletypes Hatchback Sedan SUV** //add\_ vehicletypes is a command that registers the names of the vehicle types. It takes a list of the types names separated with a space. Ex: vehicletypes[0] = “Hatchback”; vehicletypes[1] = “Sedan”; vehicletypes[2] = “SUV”;

**add\_manufacturers Hatchback Toyota KIA Hyundai** //add\_manufacturers is a command that registers the names of the manufacturers. It takes the name of the vehicle type as the first argument, and then the list of the manufacturers’ names separated with a space. “Hatchback” vehicle type has three manufacturers, so three names are defined for its manufacturers (Toyota, KIA, Hyundai). Note: you have to find the index of the vehicle type and the index of the manufacturer to set the names.

**add\_manufacturers Sedan Nissan Chevrolet** //” Sedan” vehicle type has two manufacturers, so two names are defined for its manufacturers (Nissan, Chevrolet).

**add\_manufacturers SUV Ford Jeep** //”SUV” vehicle type has 2 manufacturers, so two names are defined for its manufacturers (Ford, Jeep).

**add\_vehicles Hatchback Toyota 3** //”add\_vehicles” is a command that registers the vehicles of the manufacturer in the vehicle type. It takes the name of the vehicle type as the first argument, the name of the manufacturer as the second argument, the number of the vehicles as the third argument. All these arguments are separated with a space. Then it lists the vehicles’ information (Model\_EngineCapacity\_Year) and Mileage, depends on the number of the vehicles. Vehicle information and Mileage are separated with a space.

Ex: in the “Hatchback” vehicle type and “Toyota” manufacturer, there are 3 vehicles, and their information and Mileage are listed below. You need 2 three-dimension arrays; the first one to store the vehicle’s information (Model\_EngineCapacity\_Year) as a string, and the second one to store the Mileage as a double.

Yaris\_1.5\_2017 16.8

Vitz\_1.0\_2018 20.2

Passo\_1.3\_2018 17.6

For example, the input **Yaris\_1.5\_2017 16.8** contains a String (Yaris\_1.5\_2017) as vehicle information and a double value (16.8) as Mileage. You need to extract the model (Yaris), engine capacity (1.5) and year (2017) separately and store them in the first three-dimension array as a string, while mileage in the second three-dimension array as a double.

**add\_vehicles Hatchback KIA 6** //In the “Hatchback” type and “KIA” manufacturer, there are 6 vehicles, and their information and Mileage are listed below.

Rio\_1.4\_2019 19.2

Picanto\_1.0\_2019 23.4

Cerato\_1.8\_2019 15.5

Soul\_1.3\_2017 17.3

Forte\_1.5\_2018 16.5

Ceed\_1.4\_2019 17.7

**add\_vehicles Hatchback Hyundai 3** //In the “Hatchback” type and “Hyundai” manufacturer, there are 3 vehicles, and their information and Mileage are listed below.

i10\_1.0\_2019 21.8

Roadster\_1.8\_2017 16.6

Accent\_1.5\_2019 18.2

**add\_vehicles Sedan Nissan 2** //In the “Sedan” type and “Nissan” manufacturer, there are 2 vehicles, and their information and Mileage are listed below.

Altima\_2.0\_2016 13.8

Maxima\_2.4\_2016 12.5

**add\_vehicles Sedan Chevrolet 4** //In the “Sedan” type and “Chevrolet” manufacturer, there are 4 vehicles, and their information and Mileage are listed below.

Impala\_2.4\_2017 13.8

Malibu\_1.8\_2018 14.9

Cruze\_2.0\_2018 14.3

Lumina\_2.4\_2016 12.2

**add\_vehicles SUV Ford 2** //In the “SUV” type and “Ford” manufacturer, there are 2 vehicles, and their information and Mileage are listed below.

Explorer\_2.3\_2016 15.2

Flex\_3.3\_2018 10.9

**add\_vehicles SUV Jeep 1** //In the “SUV” type and “Jeep” manufacturer, there is 1 vehicle, and his information and Mileage are listed below.

Wrangler\_3.0\_2018 9.3

**print\_result Hatchback Toyota** //”print\_result” is a command that prints details about the information and Mileage of the vehicles of the specific manufacturer in the specific type. It takes the name of the type as the first argument, the name of the manufacturer as the second argument. All these arguments are separated with a space. Ex: prints the details of the manufacturer “Toyota” in the “Hatchback” type.

**Note:** For all Cost per KM calculations, use gasoline price as 1.43 SAR/liter

**print\_result Hatchback KIA** //Prints the details of the manufacturer “KIA” in the “Hatchback” type.

**print\_result Hatchback Hyundai** //Prints the details of the manufacturer “Hyundai” in the “Hatchback” type.

**print\_result Sedan Nissan** //Prints the details of the manufacturer “Chevrolet” in the “Sedan” type.

**print\_result Sedan Chevrolet** //Prints the details of the manufacturer “Nissan” in the “Sedan” type.

**print\_result SUV Ford** //Prints the details of the manufacturer “Ford” in the “SUV” type.

**print\_result SUV Jeep** //Prints the details of the manufacturer “Jeep” in the “SUV” type.

**find\_average\_mileage** //” find\_average\_mileage” is a command that finds and prints the average Mileage of vehicles in the all types.

**find\_best\_mileage** //” find\_best\_mileage” is a command that finds and prints the best vehicles in the all types.

**find\_worst\_mileage** //” find\_worst\_mileage” is a command that finds and prints the worst vehicles in the all types.

**about\_guide** //” about\_app” is a command that prints the information about the develop of the program (You).

**exit** //” exit” is a command that ends the program. After this command, the output file called “output.txt” must be generated in the same directory of the program. Also, the date and time of the generation should be printed in the output file.

|  |
| --- |
|  |

Figure 2. Input file (input.txt) example.

**Output File Pattern**

The output file should include all the results of the commands that have been read from the input file “input.txt”. The format of the output file is provided as separate file “output.txt”.

**Commands You Have to Implement**

As you see in the input and output files, there are some commands that you have to implement:

|  |  |
| --- | --- |
| Command #1 | |
| **Command** | **add\_ types** |
| **Description** | It registers the names of the types. It takes a list of the types’ names separated with a space. |
| **Example** | add\_types Hatchback Sedan SUV |
| **Output of the Example** | - Command: add\_types  + Hatchback + Sedan + SUV |

|  |  |
| --- | --- |
| Command #2 | |
| **Command** | **add\_manufacturers** |
| **Description** | It registers the names of the manufacturers. It takes the name of the type as the first argument, and then the list of the manufacturers’ names separated with a space. |
| **Example** | add\_manufacturers SUV Ford Jeep |
| **Output of the Example** | - Command: add\_manufacturers  -> Type: SUV  + Ford + Jeep |

|  |  |
| --- | --- |
| Command #3 | |
| **Command** | **add\_vehicles** |
| **Description** | It registers the vehicles of the manufacturer in the type. It takes the name of the type as the first argument, the name of the manufacturer as the second argument, the number of the vehicles as the third argument. All these arguments are separated with a space. Then it lists the vehicles’ information (Year\_Model\_EngineCapacity) and Mileage depends on the number of the vehicles. Vehicle information and Mileage are separated with a space. |
| **Example** | add\_vehicles Hatchback Toyota 3  Yaris\_1.5\_2017 16.8  Vitz\_1.0\_2018 20.2  Passo\_1.3\_2018 17.6 |
| **Output of the Example** | - Command: add\_vehicle  -> VehicleType: Hatchback  -> Manufacturer: Toyota  -> Number of vehicles: 3  ---------------------------------------------------------------------  Model Engine Capacity Year Mileage  Yaris 1.5 2017 16.80  Vitz 1.0 2018 20.20  Passo 1.3 2018 17.60  --------------------------------------------------------------------- |

|  |  |
| --- | --- |
| Command #4 | |
| **Command** | **print\_result** |
| **Description** | It prints details about the information and Mileage of the vehicles of the specific manufacturer in the specific type. It takes the name of the type as the first argument, the name of the manufacturer as the second argument. All these arguments are separated with a space. |
| **Example** | print\_result Sedan Chevrolet |
| **Output of the Example** | - Command: print\_result  -> VehicleType: Sedan  -> Manufacturer: Chevrolet  -> Number of vehicles: 4  ----------------------------------------------------------------------  Model Engine Capacity Year KM/Liter Cost per KM  ----------------------------------------------------------------------  Impala 2.4 2017 13.80 0.10 SAR  Malibu 1.8 2018 14.90 0.10 SAR  Cruze 2.0 2018 14.30 0.10 SAR  Lumina 2.4 2016 12.20 0.12 SAR  ----------------------------------------------------------------------  \* The average Mileage is 13.80  \* The best vehicle is Malibu 1.8 (2018)  \* The worst vehicle is Lumina 2.4 (2016) |
| Command #5 | |
| **Command** | **find\_average\_mileage** |
| **Description** | It finds and prints the average Mileage of vehicles in the all types. |
| **Example** | find\_average\_mileage |
| **Output of the Example** | - Command: find\_average\_mileage  \* The average Mileage for all vehicles in all vehicle types is 16.08 |

|  |  |
| --- | --- |
| Command #6 | |
| **Command** | **find\_best\_mileage** |
| **Description** | It is a command that finds and prints the best vehicles in the all types. |
| **Example** | find\_best\_mileage |
| **Output of the Example** | - Command: find\_best\_mileage  \* The best vehicle mileage in all vehicle types is:  ------------------------------------------------------------------------  Model Engine Capacity Year KM/Liter Cost per KM  ------------------------------------------------------------------------  Picanto 1.0 2019 23.40 0.06 SAR  ------------------------------------------------------------------------  In Vehicle Type: Hatchback, Manufacturer: KIA |

|  |  |
| --- | --- |
| Command #7 | |
| **Command** | **find\_worst\_mileage** |
| **Description** | It finds and prints the worst vehicles in the all types. |
| **Example** | find\_worst\_mileage |
| **Output of the Example** | - Command: find\_worst\_mileage  \* The worst vehicle mileage in all vehicle types is:  ------------------------------------------------------------------------  Model Engine Capacity Year KM/Liter Cost per KM  ------------------------------------------------------------------------  Wrangler 3.0 2018 9.30 0.15 SAR  ------------------------------------------------------------------------  In Vehicle Type: SUV, Manufacturer: Jeep |

|  |  |
| --- | --- |
| Command #8 | |
| **Command** | **about\_guide** |
| **Description** | It prints the information about the developer of the guide (student info). |
| **Example** | about\_guide |
| **Output of the Example** | - Command: about\_guide  -> Developed By: Ahmed Alqarni  -> University ID: 1909292  -> Section: CA |

|  |  |
| --- | --- |
| Command #9 | |
| **Command** | **exit** |
| **Description** | It ends the program. After this command, the output file called “output.txt” must be generated in the same directory of the program. Also, the date and time of the generation should be printed in the output file. |
| **Example** | exit |
| **Output of the Example** | Thank You! :)  Report generated on Sat Sep 12 01:59:05 AST 2020 |

**Deliverable**

You have to submit only the java file of your code. The file and the class names should be “**Project\_1\_YourFirstName\_YourLastName\_YourID**”. Where “YourFirstName“ is your first name, “YourLastName“ is your last name, “YourID“ is your type ID.

**NOTE:** your name, ID, and section number must be included as comments in the file!

**Output Format**

Your program must generate output in a similar format to the given sample output file [output.txt].

**Good Luck and Start Early!**