# Formal Languages and Compilers – JFlex & Cup

Using the JFLEX lexer generator and the CUP parser generator, realize a Java program capable of recognizing a domain language for computing the fuel consumption of different vehicles. The input file is divided into 2 sections separated by a line with "%%%" (3 percent characters).

The first section contains a non empty list of fuel prices. Each element of the list has the following fields: <fuel identifier><date><fuel price>";" where:

- <fuel identifier> is composed by a prefix (one of **methane** ,**gasoline** or **gpl**) followed by an underscore and a number (i.e. *gpl_23*)
- <date> is in the format **gg/mm/yyyy**, where **gg** is a number between 0 and 30 (included) and **mm** is a number between 1 and 12 (included).
- <fuel price> is a real number followed by the keyword **euro** (i.e. *0.86 euro*).
- Each line is ended by a semicolon **;**

The second section is composed by a list (that could be empty) of vehicles with information on the distance covered (in km) and on fuel consumption. Each element of the list comprises an header and the vehicle data. The header is formed by a keyword "**vehicle**:" followed by a <car name>; then either the keyword **"plate:"** followed by a <plate id> or the keyword **"chassis:"** followed by a <chassis id>.

- <car name> is a string composed by letters, numbers and the underscore charcter (i.e. *Panda* or *Renault_4*)
- <plate id> is a string composed by 2 letters, 3 numbers and 2 letters (i.e. *AB345GF*)
- <chassis id> is a string composed by 3 letters followed by 3 or 4 numbers, the charcter "x" or the character "y" followed by a number between 0 and 128 (i.e. *ABC321x23* or *CDE3344y100*)

The vehicle data is a non empty list of one or 2 elements with the following format:
<fuel type>":" <fuel consumption> **km/l** ":" <consumption data> ";" where:

- <fuel type> is one of **methane** ,**gasoline** or **gpl** and is followed by a semicolon ":"
- <fuel consumption> is a real number followed by the keyword "**km/l"** followed by a semicolon ":"
- <consumption data> is a non empty list of comma separated elements; each element has the format: <distance covered> **km** <fuel id>"," where:
  - <distance covered> is an integer number followed by the keyword **km**
  - <fuel id> is one of the values present in the header section
  - each element is separated by the following one by means of commas
- each vehicle data entry is terminated by a semicolon ";"

The program must check the input file against the grammar described and moreover it must compute the partial and total fuel expenses for each car. The fuel expenses can be computed as:

- <fuel expenses> = <distance covered> / <fuel consumption> * <fuel price>

The fuel prices listed in the header section must be stored in a proper symbol table (i.e. Hashtable) and retrieved for computing the fuel expenses.

**Input Example**:

*methane_001 06/06/2008 0.82 ;*
*methane_002 08/06/2008 0.78 ;*
*gasoline_001 07/06/2008 1.60 ;*
*gasoline _002 11/09/2008 1.55 ;*
*gpl_001 06/06/2008 0.68 ;*
*gpl_002 10/06/2008 0.70 ;*

*%%%*

*Vehicle: Fiat_Panda*

*Plate: av523am*
*gasoline : 15 km/l : 1000 km gasoline_001, 500 km gasoline_001, 250 km gasoline_002;*
*Vehicle: Renault_Clio*
*Chassis: abc1234y128*
*methane: 20 km/l : 1000 km methane_001, 1000 km methane_002;*
*gasoline: 16 km/l : 2000 km gasoline_001;*
*Vehicle: Ford_Focus*
*Chassis: cba123x0*
*gpl : 10 km/l : 1000 km gpl_001, 1000 km gpl_002;*
*gasoline : 16.5 km/l : 2000 km gasoline_001;*

**Output Example**:

*Vehicle: Fiat_Panda*
*gasoline: 185.83 €*
*total: 185.83 €*
*Vehicle: Renault_Clio*
*methane: 80 €*
*gasoline: 200 €*
*total: 280 €*
*Vehicle: Ford_Focus*
*Gpl: 138 €*
*gasoline: 193,9 €*
*total: 331.9 €*