COMP 348 Assignment 3 C & Ruby

Name: Hualin Bai ID: 40053833

I. Procedural programming with C:

Question 1:

```c
/*****************************************************************************
COMP 348  Assignment 3
@author: Hualin Bai
@ID: 40053833
@Date: 2020/04/02
*****************************************************************************/

#include <stdio.h>
#include <malloc.h>
#include <stdlib.h>

int row, col; //for row and colum of 2D array

/* the method is to transpose the elements of 2D array*/
void matrixTranspose( int **array){
   int **transarr;//pointer to pointer to the transarr[col][row]

   printf("...doing matrixTranspose...\n");
   //handle memory allocation for 2D array
    transarr = (int**) malloc(col * sizeof(int*));
    for(int r=0; r<col; r++){
        transarr[r] = (int*) malloc(row * sizeof(int));
    }

    //judgement whether memory is allocated
    if(transarr == NULL){
        printf("Error: out of memory.\n");
        return;
    }

```

```
    //transpose
    for(int r=0; r<col; r++){
        for(int c=0; c<row; c++){
            transarr[r][c] = array[c][r];
        }
    }

    //show the transposed 2D array
    printf("show the transposed 2D array\n");
    for(int r=0; r < col; r++){
        for(int c=0; c < row; c++){
            printf("%d\t", transarr[r][c] );
        }
        printf("\b \n");
    }

    //release transarr memory
    for(int r=0; r<col; r++){
        free(transarr[r]);
    }
    free(transarr);
}
```

```c
int main()
{
    int **array; //pointer to pointer to the array

    //set values of row and colum of 2D array
    printf("Enter values of row and colum of 2D array.\n");
    scanf("%d %d", &row, &col);
    if(row <=0 || col <=0){
        printf("row and col should be a positive integer!");
        return -1;
    }

    //handle memory allocation for 2D array
    array = (int**) malloc(row * sizeof(int*));
    for(int r=0; r<row; r++){
        array[r] = (int*) malloc(col * sizeof(int));
    }
    //judgement whether memory is allocated
    if(array == NULL){
        printf("Error: out of memory.\n");
        return 1;
    }

    //Enter value of each element of 2D array
    printf("Enter value of each element of 2D array\n");
    for(int r=0; r<row; r++){
        for(int c=0; c<col; c++){
            printf("set array[%d][%d]\n", r, c);
            scanf("%d", &array[r][c]);
        }
    }

```

```
86      //show the 2D array
87      printf("show the 2D array\n");
88      for(int r=0; r < row; r++){
89          for(int c=0; c < col; c++){
90              printf("%d\t", array[r][c] );
91          }
92          printf("\b \n");
93      }
94
95      //recall matrixTranspose
96      matrixTranspose(array);
97
98      //release array memory
99      for(int r=0; r<row; r++){
100         free(array[r]);
101     }
102     free(array);
103
104     return 0;
105 }
```

Output:

```
Enter values of row and colum of 2D array.
2
3
Enter value of each element of 2D array
set array[0][0]
1
set array[0][1]
2
set array[0][2]
3
set array[1][0]
4
set array[1][1]
5
set array[1][2]
6
show the 2D array
1       2       3
4       5       6
...doing matrixTranspose...
show the transposed 2D array
1       4
2       5
3       6
```

Question 2:

main.c saving...

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <malloc.h>

//node for maker-level structures
struct node {
    char *maker; //pointer to car maker
    struct node *next; //pointer to next maker
    struct model *below; //pointer to model structure
};

//node for model-level structures, include 12 attributes
struct model {
    char *car_maker;
    char *car_model;
    char *trim;
    char *km;
    char *year;
    char *type;
    char *drivetrain;
    char *transmission;
    char *stock;
    char *status;
    char *fuel_eco;
    char *set_of_features;
    char *attrs;//for show all attributes
    struct model *next;//pointer to next model
};
```

```
30  //this method is to get all features of vehicle
31  void getfeatures(char *line, struct model *head_car_model )
32  {
33    // firstly split {set_features} of line, then split other attributes
34    const char s[] = "{";
35    char *arr_f; //for store set_features {features}
36    char *arr_o; //store first split other attributes
37    //store split other 11 attributes of car
38    char *arr0, *arr1, *arr2, *arr3, *arr4, *arr5, *arr6, *arr7, *arr8, *arr9, *arr10;
39    char *token;
40    //set attrs of struct model
41    char attributes[3000];//store all atttrs, to keep the data, not use pointer.
42    strcpy(attributes, line);
43    ((head_car_model)->attrs) = attributes;
44    //printf(" %s\n", head_car_model->attrs);
45
46    //use strtok to split {set_features} of line, store in att_f[]
47    token = strtok(line, s);
48    arr_o = token;
49    while(token != NULL){
50      arr_f =  token;
51      token = strtok(NULL, s);
52    }
53    //move all elements in arr_f to arr_ff
54    char arr_ff[strlen(arr_f)+1];
55    for(int i = 0; i<strlen(arr_f)-1; i++){
56      arr_ff[i+1] = arr_f[i];
57    }
58    //add '{' before features
59    arr_ff[0] = '{';
60    //assign set_features to arr_f
61    arr_f = arr_ff;
62    //split other attributes in arr_o
63    //need to split ',' in arr_o
64    int n = 0;
65    while( n < 11){
66      token = strtok(arr_o, ",");
67      //assign other attributes to arr[0-10] in order of output file
68      switch(n){
69        case 1:
70          arr1 = token;
71          break;
72        case 2:
73          arr2 = token;
74          break;
```

```
75          case 3:
76              arr3 = token;
77              break;
78          case 4:
79              arr4 = token;
80              break;
81          case 5:
82              arr5 = token;
83              break;
84          case 6:
85              arr6 = token;
86              break;
87          case 7:
88              arr7 = token;
89              break;
90          case 8:
91              arr8 = token;
92              break;
93          case 9:
94              arr9 = token;
95              break;
96          case 10:
97              arr10 = token;
98              break;
99          default:
100             arr0 = token;
101             break;
102     }
103     while(token != NULL){
104         arr_o = token;
105         token = strtok(NULL, s);
106     }
107     n += 1;
108 }
109     //assign attributes to structures
110     head_car_model->car_maker = arr0;
111     head_car_model->car_model = arr1;
112     head_car_model->trim = arr2;
113     head_car_model->km = arr3;
114     head_car_model->year = arr4;
115     head_car_model->type = arr5;
116     head_car_model->drivetrain = arr6;
117     head_car_model->transmission = arr7;
118     head_car_model->stock = arr8;
119     head_car_model->status = arr9;
```

```
    head_car_model->fuel_eco = arr10;
    head_car_model->set_of_features = arr_f;

}

//this method is to given a reference (pointer to pointer) to the head of a list and a char, inserts a new node on the front of the list.
void push( struct node** head_ref, char *new_data)
{
  struct node *new_maker = malloc(sizeof(struct node));
  new_maker->maker = new_data;
  new_maker->next = (*head_ref);
  (*head_ref) = new_maker;
  new_maker->below = NULL;
  printf("Create the maker node is %s \n", ((new_maker)->maker));
}
//this method is to insert model
void insertModel( struct model *prev_model, struct model *model_below)
{
  model_below->next = prev_model->next;
  prev_model = model_below;
}


//this method is to insert a new model node in given maker node
void pushModel(struct node** prev_ref,  struct model *new_below)
{
  if(prev_ref == NULL){
    printf("the given previous node cannot be null");
    return;
  }
  if(((*prev_ref)->below) == NULL){
    ((*prev_ref)->below) = new_below;
    new_below->next = NULL;
  }
  else{
    new_below->next = ((*prev_ref)->below);
    ((*prev_ref)->below) = new_below;
  }
  printf("Show multi linked list:\n");
  printf("the node of (%s) (maker->below) is: %s\n\n", ((*prev_ref)->maker) , (*prev_ref)->below->car_model);

}
```

```c

//this method is searchInventory, using traverse linked list to find all matched key and value
void searchInventory(char *str1, char *str2, struct node *head_search)
{
  //printf(" %s\n", (head_search->below->car_maker));
  //printf("test  %d\n", (head_search->next) == 0);
  printf("Find all matched data:\n ");

  while((head_search->next) != 0){
    if(strcmp((head_search->maker), str2) == 0){
      printf("match maker is: %s\n", (head_search->maker));
        while((head_search->below) != 0){
            printf("%s\n", (head_search->below->attrs));
            head_search->below = head_search->below->next;
        }
        break;
      }
      head_search = (head_search->next);//find next node

  }
}


//this method is to savecatalogue2file( head_maker )
void savecatalogue2file( struct node *head_maker_save)
{
  //create and write an output file
  FILE *ft = NULL;
  ft = fopen("./output.txt", "a+");
  //printf("oooo  %s\n", head_maker_save->maker);
  while(head_maker_save->next != NULL){
    //printf("%s\n", head_maker_save->maker );
    //create file output
    while((head_maker_save->below) != NULL){
      printf("%s\n", head_maker_save->below->attrs );
      fputs(head_maker_save->below->attrs,ft);
      head_maker_save->below = head_maker_save->below->next;
    }
    head_maker_save = head_maker_save->next;
  }
//close file
  fclose(ft);

}

```

```c
//this method is to add2Inventory(data)
void add2Inventory(char* data, struct node *insert_maker)
{
  //create a model node
  //printf("%s\n", insert_maker->maker);
  struct model *add_model = NULL;
  add_model = malloc(sizeof(struct model));
  if (add_model == NULL) { printf("fail to allocate memory!");}
  getfeatures(data, add_model);
  //printf("%s\n", add_model->car_maker);

  //push into linked list
  while((insert_maker->next) != 0){
    if(strcmp((insert_maker->maker), (add_model->car_maker)) == 0){
      pushModel(&insert_maker, add_model);
      break;
    }
    insert_maker = (insert_maker->next);//find next node

  }
  printf("this data has already added into linked list:\n\n");
  printf("%s\n", add_model->attrs);

}
```

```
230
231 int main(void) {
232   // read data file
233   FILE *fp = NULL; //file pointer
234   fp = fopen("./car_data.txt", "r");
235   char buff[1024];
236
237   if(fp == NULL)
238   {
239     perror("Fail to read!");
240     return -1;
241   }
242   //create struct node of 5 car makers
243   struct node *head_maker = NULL;
244   struct node *head_node = NULL; //for additionally head pointer to node of list
245   char *data = "Bmw";
246   push(&head_maker, data);
247   data = "Lexus";
248   push(&head_maker, data);
249   data = "Toyota";
250   push(&head_maker, data);
251   data = "Mercedec";
252   push(&head_maker, data);
253   data = "Honda";
254   push(&head_maker, data);
255   head_node = malloc(sizeof(struct node));
256   head_node->next = head_maker;
257
258   printf("Already created maker linked list.\n\n");
259   printf("creating multi linked list...\n\n");
260   //read ordered output file
261   while (fgets(buff, sizeof(buff), fp) != NULL)
262   {
263     struct model *head_model = NULL;
264     head_model = malloc(sizeof(struct model));
265     if (head_model == NULL) { printf("fail to allocate memory!");}
266     //create multi linked list
267     getfeatures(buff, head_model);
268
```

```
    //use head_maker to traverse all node maker, to find same maker, then node->below->new model node.
    while((head_maker->next) != 0){
      if(strcmp((head_maker->maker), (head_model->car_maker)) == 0){
        pushModel(&head_maker, head_model);
        break;
      }
      head_maker = (head_maker->next);//find next node
    }
    //recall savecatalogue2file()
    savecatalogue2file(head_maker);

  }

  char add_data[500] = "Bmw, X6, SS, 10km, 2018, SUV, AWD, auto, 22AA999A, new, 10L/100km, {Heated
  Seats, Heated Mirrors, Keyless Entry}";
  // pointer is back to head of the node list
  head_maker = head_node->next;
  add2Inventory(add_data, head_maker);

  //recall searchInventory(str_key, str_value, head_maker)
  char *str_key, *str_value;
  str_value = "Mercedec";
  str_key = "car_maker";
  searchInventory(str_key, str_value, head_maker);

  fclose(fp); //close file
  //free malloc
  // free(head_node);
  // free(head_model);

  return 0;

}
```

Output:

```
clang version 7.0.0-3~ubuntu0.18.04.1 (tags/RELEASE_700/final)
> clang-7 -pthread -lm -o main main.c
> ./main
Create the maker node is Bmw
Create the maker node is Lexus
Create the maker node is Toyota
Create the maker node is Mercedec
Create the maker node is Honda
Already created maker linked list.

creating multi linked list...

Show multi linked list:
the node of (Honda) (maker->below) is:  CRV

Show multi linked list:
the node of (Mercedec) (maker->below) is:  CLK

Show multi linked list:
the node of (Toyota) (maker->below) is:  camry

>
```

```
creating multi linked list...

Show multi linked list:
the node of (Honda) (maker->below) is:  CRV

Honda, CRV, LE, 0km, 2018, SUV, AWD, auto, 19BF723A, new, 8L/100km, {Heat
ed Seats, Heated Mirrors, Keyless Entry}

Show multi linked list:
the node of (Mercedec) (maker->below) is:  CLK

Mercedec, CLK, LX, 1100km, 2017, coupe, RWD, auto, 18FO724A, Used, 6L/100
km, {AC, Heated Seats, Heated Mirrors, Keyless Entry, Power seats}

Show multi linked list:
the node of (Toyota) (maker->below) is:  camry

Toyota, camry, SE, 65101km, 2010, Sedan, FWD, Manual, 18131A, Used, 5.5L/
100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry}
```

```
Find all matched data:
 match maker is: Mercedec
Mercedec, CLK, LX, 1100km, 2017, coupe, RWD, auto, 18FO724A, Used, 6L/100
km, {AC, Heated Seats, Heated Mirrors, Keyless Entry, Power seats}
```

```
this data has already added into linked list:

Bmw, X6, SS, 10km, 2018, SUV, AWD, auto, 22AA999A, new, 10L/100km, {Heate
d Seats, Heated Mirrors, Keyless Entry}
```

II. Object Oriented Programming with Ruby:

Question 3:

```ruby
def charcount(array)
  array.sort.each {|x| puts "#{x}, ch_count= #{x.length} "}
end

array = ["Adam", "Eve", "Mark", "Franklin", "John"]
charcount(array)
```

```
Adam, ch_count= 4
Eve, ch_count= 3
Franklin, ch_count= 8
John, ch_count= 4
Mark, ch_count= 4
=> ["Adam", "Eve", "Franklin", "John", "Mark"]
```

Question 4:

```ruby
def calcARI(filename)
  char_num = word_num = sent_num = 0
  File.open(filename).each{|line|
    # scan (\w) to count character number
    char_num = line.scan(/\w/).length + 1
    # scan (\s) to count word number, plus 1 since omit last space
    word_num = line.scan(/\s/).length + 1
    #scan (. ? !) to determine whether is a sentence.
    sent_num = line.scan(/\.|\?|\!/).length
  }

  # calculate ARI
  # use to_f to change type integer to float or use Rational(char_num, word_num)
  ari = ((4.71*(char_num.to_f/word_num)) + (0.5 * (word_num.to_f/sent_num))) -21.43

  puts "Total # of characters: #{char_num}"
  puts "Total # of words: #{word_num}"
  puts "Total # of sentences: #{sent_num}"
  puts "Automated Readability Index: #{ari.round(1)}"
  # determine grade level
  case ari
    # range is [1,2)
    when 1...2
      puts "Grade level: 5-6 (Kindergarten)"
    when 2...3
      puts "Grade level: 6-7 (First/Second Grade)"
    when 3...4
      puts "Grade level: 7-9 (Third Grade)"
    when 4...5
      puts "Grade level: 9-10 (Fourth Grade)"
    when 5...6
      puts "Grade level: 10-11 (Fifth Grade)"
    when 6...7
      puts "Grade level: 11-12 (Sixth Grade)"
    when 7...8
      puts "Grade level: 12-13 (Seventh Grade)"
    when 8...9
      puts "Grade level: 13-14 (Eighth Grade)"
    when 9...10
      puts "Grade level: 14-15 (Ninth Grade)"
    when 10...11
      puts "Grade level: 15-16 (Tenth Grade)"
    when 11...12
      puts "Grade level: 16-17 (Eleventh Grade)"
```

```ruby
        when 12...13
            puts "Grade level: 17-18 (Twelfth grade)"
        when 13...14
            puts "Grade level: 18-24 (College student)"
        when 14
            puts "Grade level: 24+ (Professor)"
        else
            puts "Error Grade!"
        end

    end

    filename = "paragrah.txt"
    calcARI(filename)
```

Output:

```
Total # of characters: 474
Total # of words: 96
Total # of sentences: 4
Automated Readability Index: 13.8
Grade level: 18-24 (College student)
```

Question 5:

Main.rb

main.rb saving...

```ruby
# an inventory showroom system
#for import and export car informations list.
require "./CarModel"

WELCOME = "*****************************\n"\
          "***CAR INVENTORY SHOWROOM***\n"\
          "*****************************\n\n"

MENU = "Choose an option:\n"\
       " 1. convertListings2Catalougue\n"\
       " 2. searchInventory\n"\
       " 3. Add2Inventory\n"\
       " 4. saveCatalogue2File\n"\
       " 5. show_stocknumber\n"\
       " 6. Exit\n" \
       ">>>>>>>>>>>>>>>>>>>>"

#display welcome and menu
puts WELCOME, MENU

# create new variable
a_car = CarModel.new("car_data.txt")

# process option selection of menu
leave = false
until leave
  # get option from input
  option = gets
  case option[0]
    when "1"
      puts "processing convertListings2Catalougue\n"
      a_car.convertListings2Catalougue
    when "2"
      puts "processing searchInventory"
      a_car.searchInventory({"car_maker" => "Mercedec"})
    when "3"
      puts "please input a car information:\n"
      input = gets
      puts "processing add2Inventory\n"
      a_car.add2Inventory(input)
    when "4"
      puts "processing saveCatalogue2File\n"
      a_car.saveCatalogue2File
```

```
44      when "5"
45        puts "processing show_stocknumber"
46        a_car.show_stocknumber
47      when "6"
48        leave = true
49        puts ">>>Exit inventory showroom<<<"
50      else
51        puts "please input an option number from 1 to 5."
52    end
53  end
54
```

CarModel.rb

CarModel.rb saving...

```ruby
# class Car_model
# to store all the listing information
# to store listing fearures
require "./CarMaker"
class CarModel

  # 12 listing features: KM, TYPE_CAR, TRANSMISSION, STOCK_NUM, DRIVETRAIN, STATUS,
  #FUEL_ECONOMY, CAR_MAKER, YEAR, TRIM, SET_FEATURES, Model(others)
  # patterns for car attributes match
  # use freeze() method on constants to make them immutable
  # Use pos/neg lookahead/lookbehind in a regex to specify what chars shoud follow the patterns.
  eg.(?<!exp)
  # i for ignoreCase, x for ignorePatternWhitespace

  # match #km that kilometers followed by "km".
  KM = /(\d*(^[L\/]?)\d+km(?!\w+))/i.freeze

  # match Type with range in {Sedan, coupe, hatchback, station, SUV}
  # can use (^sedan$) or ((?<!\w)sedan(?!\w+))
  TYPE_CAR = /
      (^Sedan$)|
      (^coupe$)|
      (^hatchback$)|
      (^station$)|
      (^SUV$)/ix.freeze

  # match Transmission including {Auto, manual, steptronic}
  TRANSMISSION = /
      (^Auto(?!\w+))|
      (^manual(?!\w+))|
      (^steptronic(?!\w+))/ix.freeze

  # match Stock# that Combination of letters and numbers NOT ending with “km”
  STOCK_NUM = /^\d{2,}[A-Z]+\d*[A-Z]*(?!km)$/i.freeze

  # match DRIVETRAIN in {FWD, RWD, AWD}
  DRIVETRAIN = /
      (^fwd(?!\w+))|
      (^rwd(?!\w+))|
      (^awd(?!\w+))/ix.freeze

  # match STATUS in {Used, new}
  STATUS = /
        ^Used(?!\w+)|
        ^new(?!\w+)/ix.freeze
```

```ruby

# match Fuel Economy that similar to: 5.5L/100km format
FUEL_ECONOMY = /
    (\d*(\.?)\d*[L](\/)\d+km$)/ix.freeze

# match car_maker in {Honda, Toyota, Mercedes, BMW, Lexus}
CAR_MAKER = /
    ^Honda(?!\w+)|
    ^Toyota(?!\w+)|
    ^Mercede[sc](?!\w+)|
    ^BMW(?!\w+)|
    ^Lexus(?!\w+)/ix.freeze

# match Model that Any text that doesn't match any of the other criteria in this table, we can just use if...else to set.
MODEL = ""

# match Year
YEAR = /^\d{4}(?!\w+)/ix.freeze

# match Trim that any two letters acronym
TRIM = /^[[:alpha:]]{2}(?!\w+)/ix.freeze

# match set of features that Any set of features inside curly parenthesis
SET_FEATURES = /(?<=\{).*(?=\})/ix.freeze

#attr_accessor
attr_accessor :file_name, :car_inventory, :car_attributes

#construstor
def initialize(file_name)
  @file_name = file_name
  # car_inventory array
  @car_inventory = [] #an array
  @car_attributes = {} #a hash
end

# This method is convertListings2Catalougue
def convertListings2Catalougue()

  puts "the file does not exist!" unless File.exist?(@file_name)

```

```ruby
    # reads the listing file line by line
    # and correctly recognizes and extracts different listing features
    File.open(@file_name).each{ |line|
      add2Inventory(line)
    }
    puts "Already add all car objects from listing in car_inventory."

end

# method Add2Inventory
# that accepts a new listing, then add the line to the original file
# and add an appropriate object to the catalogue based on the listing features.
def add2Inventory(line)

  # firstly extract set_features, since it specify includes {}.
    if line =~ SET_FEATURES
      car_attributes[:set_features] = line.match(SET_FEATURES)[0].split(/\s*\,+\s*/)
      #since already extracted features, then remove it to avoid workload.
      #puts line.match(SET_FEATURES)[0].split(/\s*\,+\s*/)
      line.slice!(SET_FEATURES)
      line.slice!("{}")
    end

    # then ceck each word of line
    # store them into hash (car_attributes)
    # 12 listing features: KM, TYPE_CAR, TRANSMISSION, STOCK_NUM, DRIVETRAIN, STATUS,
    FUEL_ECONOMY, CAR_MAKER, YEAR, TRIM, SET_FEATURES, Model(others)
    words = line.split(/\s*\,+\s*/)
    words.each{|word|
          if  (w = word.match(KM))
            car_attributes[:km] = w[0]
          elsif (w = word.match(TYPE_CAR))
            car_attributes[:type_car] = w[0]
          elsif (w = word.match(TRANSMISSION))
            car_attributes[:transmission] = w[0]
          elsif (w = word.match(STOCK_NUM))
            car_attributes[:stock_num] = w[0]
          elsif (w = word.match(DRIVETRAIN) )
            car_attributes[:drivetrain] = w[0]
          elsif (w = word.match(STATUS))
            car_attributes[:status] = w[0]
          elsif w = word.match(FUEL_ECONOMY)
            car_attributes[:fuel_economy] = w[0]
          elsif w = word.match(CAR_MAKER)
            car_attributes[:car_maker] = w[0]
```

```
129                 car_attributes[:car_maker] = w[0]
130               elsif w = word.match(YEAR)
131                 car_attributes[:year] = w[0]
132               elsif w = word.match(TRIM)
133                 car_attributes[:trim] = w[0]
134               else
135                 car_attributes[:model] = word
136             end
137           }
138           #recall update_car_inventory method
139           # to create the appropriate object and add it into the #array of car_inventory
140           car = update_car_inventory(car_attributes)
141
142           # check whether this line has alread exist in data arry
143           # -1 for same, 1 for no exist
144            check = 0
145            if(!car_inventory.empty?)
146              check = check_same(car)
147            end
148
149
150           if(check == -1)
151             puts "this car has exist in inventory!"
152           elsif((car != 0 && check == 0) || (car != 0 && check == 1))
153             @car_inventory.push(car)
154             #puts "adding car object in car_inventory"
155           else
156             puts "please input correct car_maker!"
157           end
158
159       end
160
161       # this method is to update car inventory from the given hash of attributes
162       private
163       def update_car_inventory(car_attributes)
164         # generate proporite objects based on :car_maker attribute
165         # car_maker in {Honda, Toyota, Mercedes, BMW, Lexus}
166         case car_attributes[:car_maker].downcase
167         when "honda"
168           ahoda = Honda.new(car_attributes)
169           #puts ahoda.to_s
170           #puts ahoda.gettotal
171         when "toyota"
172           Toyota.new(car_attributes)
173           #puts Toyota.new(car_attributes).to_s
```

```ruby
    when "mercedec", "mercedes"
      car_attributes[:car_maker] = "Mercedec"
      Mercedes.new(car_attributes)
      #puts Mercedes.new(car_attributes).to_s
    when "bmw"
      Bmw.new(car_attributes)
      #puts Bmw.new(car_attributes).to_s
    when "lexus"
      Lexus.new(car_attributes)
      #puts Lexus.new(car_attributes).to_s
    else
      puts "incorrect car_maker!"
      0
    end

  end

# the method is to check if the car object is exist in data
# transfer line to an object, then compare it with each car object in array
private
def check_same(car)
  car_inventory.each{|item|
    return -1 if (car.to_s == item.to_s )
  }
  return 1
end

# The method is to store all created catalogue objects to an output file alphabetically
according to their maker name.
public
def saveCatalogue2File
  # create a output file
  output_file = File.open("./output.txt", "w+")
  # sort all car objects in car_inventory array
  # Comparison are based on alphabetical order of car_maker
  @car_inventory.sort!{|a, b| a.car_maker <=> b.car_maker}

  # output each object of file
  @car_inventory.each{|car|
    output_file.puts car.to_s
    puts car.to_s
  }
  # close file
  output_file.close
end
```

```ruby

  # the method is to show the number of listings of makers in stock
  def show_stocknumber
    car_inventory.each{|item|
      item.gettotal
    }
    #show total number in stock
    car_inventory[0].getinformation

  end

  # the method is to perform an advanced search for all vehicles in stock based on the combonation of hash key-value pairs
  def searchInventory(ahash)
    car_search = []
    count = 0
    #since I use symbol[:symbol] as hash key, I need transfer symbol to # string or transfer string to symbol to compare if the keys and #values are same.
    # use method .to_s or .to_sym for unify type of keys and values of #ahash and car_attrs
    # for test type
    #puts ahash.has_key?("car_maker")
    #puts car_inventory[0].car_attrs.has_key?("car_maker".to_sym)

    car_inventory.each{|car|
      car.car_attrs.each{|key, value|
        if(ahash.has_key?(key.to_s) && ahash.has_value?(value.to_s))
          count += 1
        end
       }
      if (count == ahash.length)
          car_search.push(car)
      end
      count = 0
    }
    puts "complete searchInventory:"
    puts "show all matched cars:"
    car_search.each{|car| puts car.to_s}

  end

end

```

CarMaker.rb

CarMaker.rb saving...

```ruby
# super class CarMaker
# and its hierarchy class
class CarMaker
  # 12 listing features: KM, TYPE_CAR, TRANSMISSION, STOCK_NUM, DRIVETRAIN, STATUS, FUEL_ECONOMY, CAR_MAKER, YEAR, TRIM, SET_FEATURES, Model(others)
  attr_accessor :car_maker, :trim, :model, :km, :type_car, :transmission, :stock_num, :drivetrain, :status, :fuel_economy, :year, :set_features, :car_attrs
  @@total = 0


  # constructor
  # create a car listing with the given attributes
  # required car_maker, model, and trim lines
  def initialize(attrs)
    @car_attrs = attrs.clone #use clone to avoid change data of hash
    @car_maker = attrs[:car_maker]
    @model = attrs[:model]
    @trim = attrs[:trim]
    @km = attrs[:km]
    @type_car = attrs[:type_car]
    @transmission = attrs[:transmission]
    @stock_num = attrs[:stock_num]
    @drivetrain = attrs[:drivetrain]
    @status = attrs[:status]
    @fuel_economy = attrs[:fuel_economy]
    @year = attrs[:year]
    @set_features = attrs[:set_features]
    @@total += 1
  end

  # listing features
  def getinformation
    puts "total inventory car number is: #{@@total}\n"
  end
  # create a catalogue
  def to_s
    # use gsub! to replace [] to {}, and delete "" in string line
      set_features = @set_features.to_s.gsub!("[", "{").gsub!("]","}").gsub!("\"","")
      "#{@car_maker}, #{@model}, #{@trim}, #{@km}, #{@year}, #{@type_car}, #{@drivetrain}, #{@transmission}, #{@stock_num}, #{@status}, #{@fuel_economy}, #{set_features}\n\n"
    end

end
```

```ruby

# define some subclasses associated with car_maker
# car_maker in {Honda, Toyota, Mercedes, BMW, Lexus}
class Honda < CarMaker
    # constructor
    # create Honda car with the given attributes
    @@total_honda = 0
    def initialize(attrs)
      super(attrs)
      @@total_honda += 1
    end

    # get total number
    def gettotal
      puts "Honda total: #{@@total_honda}\n"
    end
end

class Toyota < CarMaker
    # constructor
    # create Honda car with the given attributes
    @@total_toyota = 0
    def initialize(attrs)
      super(attrs)
      @@total_toyota += 1
    end

    # get total number
    def gettotal
      puts "Toyota total: #{@@total_toyota}\n"
    end
end
```

```ruby

class Mercedes < CarMaker
    # constructor
    # create Honda car with the given attributes
    @@total_mercedes = 0
    def initialize(attrs)
      super(attrs)
      @@total_mercedes += 1
    end

    # get total number
    def gettotal
      puts "Mercedes total: #{@@total_mercedes}\n"
    end
end

class Bmw < CarMaker
    # constructor
    # create Honda car with the given attributes
    @@total_bmw = 0
    def initialize(attrs)
      super(attrs)
      @@total_bmw += 1
    end

    # get total number
    def gettotal
      puts "BMW total: #{@@total_bmw}\n"
    end
end

class Lexus < CarMaker
    # constructor
    # create Honda car with the given attributes
    @@total_bmw = 0
    def initialize(attrs)
      super(attrs)
      @@total_lexus += 1
    end

    # get total number
    def gettotal
      puts "BMW total: #{@@total_lexus}\n"
    end
end
```

Tests:

```
 /run_dir/run.sh main.rb
****************************
***CAR INVENTORY SHOWROOM***
****************************

Choose an option:
 1. convertListings2Catalougue
 2. searchInventory
 3. Add2Inventory
 4. saveCatalogue2File
 5. show_stocknumber
 6. Exit
>>>>>>>>>>>>>>>>>>>>
```

```
1
processing convertListings2Catalougue
Already add all car objects from listing in car_inventory.
```

```
2
processing searchInventory
complete searchInventory:
show all matched cars:
Mercedec, CLK, LX, 1100km, 2017, coupe, RWD, auto, 18F0724A, Used,
6L/100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry, Power
seats}
```

```
3
please input a car information:
 Toyota, camry, SE, 65101km, 2010, Sedan, FWD, Manual, 18131A, Used, 5.5L/100km, {AC, Heated
Seats, Heated Mirrors, Keyless Entry}
Toyota, camry, SE, 65101km, 2010, Sedan, FWD, Manual, 18131A, Used, 5.5L/100km, {AC, Heated
Seats, Heated Mirrors, Keyless Entry}
processing add2Inventory
this car has exist in inventory!
```

```
3
please input a car information:
 Bmw, X8, LE, 9000km, 2020, SUV, FWD, auto, 20AB987C, Used, 20.5L/100km, {AC, Heated Seats,
Heated Mirrors, Keyless Entry}
Bmw, X8, LE, 9000km, 2020, SUV, FWD, auto, 20AB987C, Used, 20.5L/100km, {AC, Heated Seats,
Heated Mirrors, Keyless Entry}
processing add2Inventory
```

```
4
processing saveCatalogue2File
Bmw, X8, LE, 9000km, 2020, SUV, FWD, auto, 20AB987C, Used, 20.5L/100km, {AC, Heated Seats,
Heated Mirrors, Keyless Entry}

Honda, CRV, LE, 0km, 2018, SUV, AWD, auto, 19BF723A, new, 8L/100km, {Heated Seats, Heated
Mirrors, Keyless Entry}

Mercedec, CLK, LX, 1100km, 2017, coupe, RWD, auto, 18FO724A, Used, 6L/100km, {AC, Heated
Seats, Heated Mirrors, Keyless Entry, Power seats}

Toyota, camry, SE, 65101km, 2010, Sedan, FWD, Manual, 18131A, Used, 5.5L/100km, {AC, Heated
Seats, Heated Mirrors, Keyless Entry}
```

```
5
processing show_stocknumber
BMW total: 1
Honda total: 1
Mercedes total: 1
Toyota total: 1
total inventory car number is: 4
```

```
6
>>>Exit inventory showroom<<<
```

Output file:

I tried to add2Inventory method to test sort by alphabetically based on car maker name.

output.txt saving...

```
1  Bmw, X8, LE, 9000km, 2020, SUV, FWD, auto, 20AB987C, Used,
   20.5L/100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry}
2
3  Honda, CRV, LE, 0km, 2018, SUV, AWD, auto, 19BF723A, new, 8L/100km,
   {Heated Seats, Heated Mirrors, Keyless Entry}
4
5  Mercedec, CLK, LX, 1100km, 2017, coupe, RWD, auto, 18FO724A, Used,
   6L/100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry, Power
   seats}
6
7  Toyota, camry, SE, 65101km, 2010, Sedan, FWD, Manual, 18131A, Used,
   5.5L/100km, {AC, Heated Seats, Heated Mirrors, Keyless Entry}
```