Welcome to AnyCar Manufacturing Plant

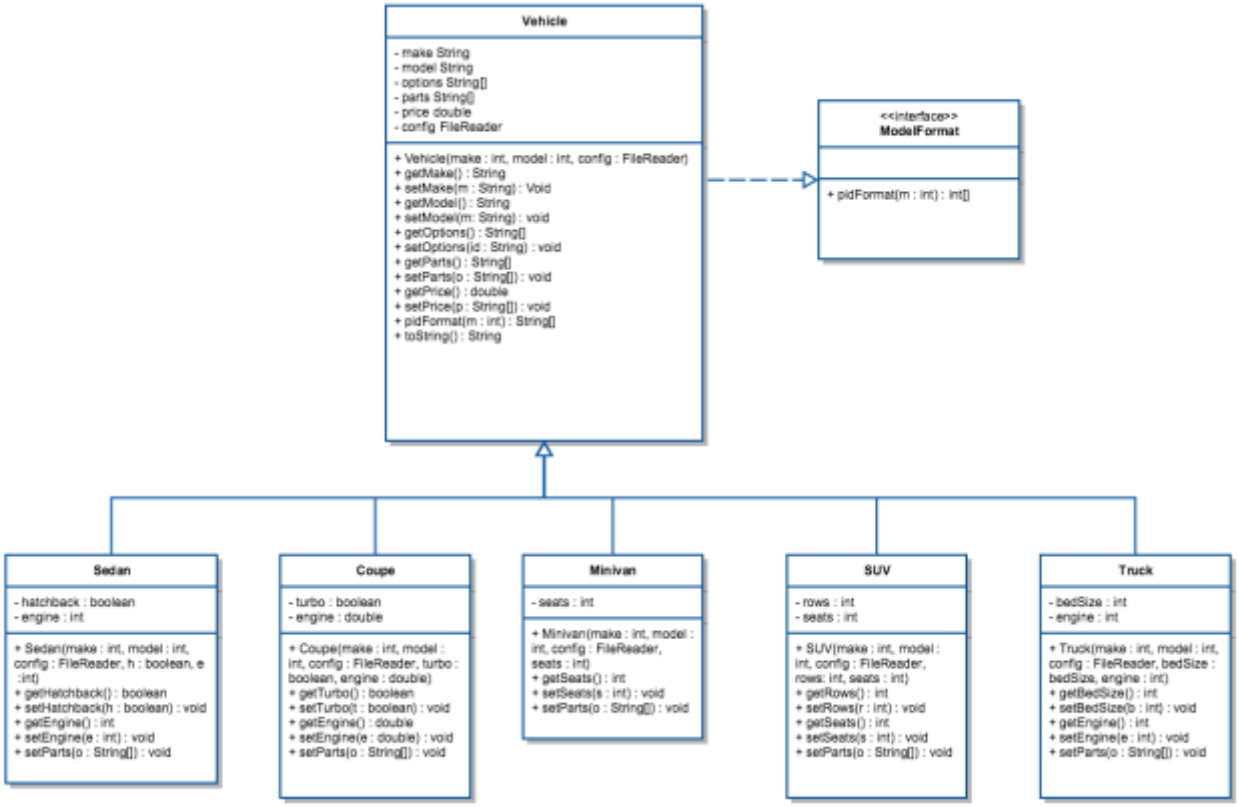
Dear Student,

Great work on AnyCarMain shell application. We have expanded our manufacturing customers to three car companies ThatAuto, ThisAuto, and OtherAuto. So, we must make some drastic changes to your application. You will have to use the 4 pillars of OOP and create different vehicle objects according to customer specs. Look at the Application Requirements and make the changes accordingly. If you have any questions let me know.

Thank you,
Your Supervisor
AnyCar
Manufacturing Technical Manager

Application Requirements

Each company has its own configuration file information needed to decipher all required information is within ‘ThisCar.config’, ‘ThatCar.config’, and ‘OtherCar.config’. The following is UML diagram for the Vehicle object and all Child objects as well as the ModelFormat interface:



AnyCarManufactor.java (was AnyCarMain.java)

AnyCarManufactor is the main file of your application. Main will initiate the application.

Global Variables:
`None`

Main

Type: Public Static
Input: String Array args – Command line arguments
Return: Void
Description: `Main` Method for initiating AnyCar application. When called will call `request` to initiate the interaction with the user.

request

Type: Public Static
Input: None
Return: Void
Description: `request` user to input PID(String) and calls `buildVehicle` to start building each vehicle. `Request` will continue to prompt the user until the user inputs -1, then request will return.

buildVehicle

Type: Public Static
Input: String pid - product ID number
Return: Void
Description: Creates a Vehicle object and call getters and setters for make, model, options, parts, and price, and print the Vehicle object using it's toString method, and return. The first two digits defines the make and the vehicle type and exhibit the following behavior:

- The first digit is the make
  - 1: ThisAuto
  - 2: ThatAuto
  - 3: OtherAuto
- The second digit is vehicle type
  - 1: Sedan
  - 2: Coupe
  - 3: Minivan
  - 4: SUV
  - 5: Truck

Vehicle.java

Abstract Vehicle object that implements ModelFormat

Private Variables:

- `String Make`
- `String Model`
- `String[] Options`
- `String[] Parts`
- `double Price`
- `FileReader Config`

Constructor

Input: String make – Vehicle Make's, String model – Vehicle's Model, FileReader config – FileReader object of config file

getMake/setMake

Getter: returns private string Make
Setter:

- Input: string make
- Operations: assigns private string make with input value

getModel/setModel

Getter: returns private string Model
Setter:

- Input: string model
- Operations: assigns private string model with input value

getOptions/setOptions

Getter: returns private String Array options
Setter:

Input: String pid - 12-digit product ID number
Return: void
Description: Use pidFormat to parse the pid and identifies the options within config. If the number is zero then the option is None. Each option is appended to an array with the option name and type as string example: "Exterior Color: White" once all options are defined then return the array of options.

getParts/setParts

Getter: returns private String Array parts
Abstract Setter:

- Input: String Array Options
- Operations: assigns private String Array parts value

getPrice/setPrice

Getter: returns private double price
Setter:

Input: String Array parts – List of parts from getParts
Return: void
Description: Iterate through parts and combine the cost of all parts and assign object variable price

pidFormat

Type: Public
Input: int m – id of manufacture
Return: Integer Array of pid Format
Description: Exhibits the following format:

m = 1. ThisAuto
a. Returns [4,5,6,7,8,9,10,11,12]
index 0. First 4 digits is the model
index 1. Digit 5 Exterior Color
index 2. Digit 6 Interior Color
index 3. Digit 7 Powertrain
index 4. Digit 8 Seat Type
index 5. Digit 9 Radio Type
index 6. Digit 10 Tire Size
index 7. Digit 11 Rim Size
index 8. Digit 12 Miscellaneous
m = 2. ThatAuto
a. Returns [6,7,8,9,10,11,12,13,15]
index 0. First 6 digits is the model
index 1. Digit 7 Exterior Color
index 2. Digit 8 Interior Color
index 3. Digit 9 Powertrain
index 4. Digit 10 Seat Type
index 5. Digit 11 Radio Type
index 6. Digit 12 Tire Size
index 7. Digit 13 Rim Size

index 8. Digits 14 and 15 Miscellaneous

m = 3. OtherAuto

a. Returns [4,5,6,7,8,9,10,11,16]

index 0. First 4 digits is the model

index 1. Digit 5 Exterior Color

index 2. Digit 6 Interior Color

index 3. Digit 7 Powertrain

index 4. Digit 8 Seat Type

index 5. Digit 9 Radio Type

index 6. Digit 10 Tire Size

index 7. Digit 11 Rim Size

index 8. Digit 12 thru 16 Miscellaneous

toString

Same as Lab 1 (Car.java)

Sadan.java

Sedan extends Vehicle. See UML Diagram

setParts
Type: Public
Input: String Array o – String Array of options
Return: Void
Description: Reads config variable to identify the parts specific to Sedans and parts specific to the options array.

Coupe.java

Coupe extends Vehicle. See UML Diagram

setParts
Type: Public
Input: String Array o – String Array of options
Return: Void
Description: Reads config variable to identify the parts specific to Coupes and parts specific to the options array.

Minivan.java

Minivan extends Vehicle. See UML Diagram

setParts
Type: Public
Input: String Array o – String Array of options
Return: Void
Description: Reads config variable to identify the parts specific to Minivans and parts specific to the options array.

SUV.java

SUV extends Vehicle. See UML Diagram

setParts
Type: Public
Input: String Array o – String Array of options
Return: Void
Description: Reads config variable to identify the parts specific to Sedans and parts specific to the options array.

Truck.java

Truck extends Vehicle. See UML Diagram

setParts
Type: Public
Input: String Array o – String Array of options
Return: Void
Description: Reads config variable to identify the parts specific to Trucks and parts specific to the options array.

ModelFormat.java

Interface between Manufactures to identify Product Identification formats.

pidFormat
Type: Public
Input: int m – id of manufacture
Return: Integer Array of pid Format
Description: Must be implements