Welcome to AnyCar Manufacturing Plant

Dear Student,

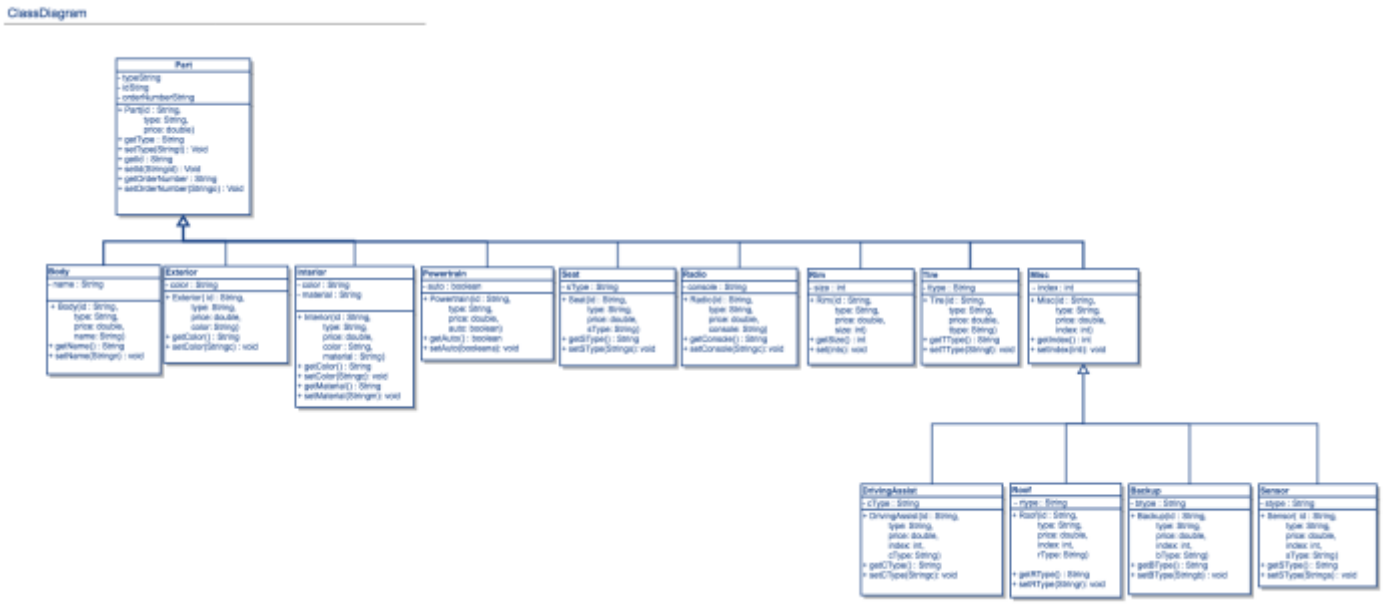
Wow! Great work your AnyCarManufactor application works and looks great! We are at the point now where users are utilizing your application to process orders using each individual car company's configuration. We now need your help in another area within our company. We are having issues with our supply department. We need an application that can track parts, assign parts to orders, and notify users when parts are needed to be ordered. We already have most of the application developed, but what we need from you is the application that processes each request.

Thank you,
Your Supervisor
AnyCar
Manufacturing Technical Manager

Application Requirements

Parts

Parts has already been implemented and must be utilized throughout your application where applicable. YOU WILL NOT MANIPULATE PARTS CLASSES AND METHODS IN ANYWAY. See UML for class format and view code comments for use.



SupplyDump.java

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

Attribute:
`Stock: <T>`

Main

Type: Public Static
Input: String Array args – Command line arguments
Return: Void
Description: `Main` Method for initiating Supply Dump application. Start by calling ingestStock, then call request.

request

Type: Public Static
Input: None
Return: Void
Description: `request` user to input order number and calls `buildOrder` to start building each order. `Request` will continue to prompt the user until the user inputs -1, then request will return.

buildOrder

Type: Public Static
Input: String oid - order ID number
Return: Void
Description: BuildOrder creates the invoice with each order. The Order ID (oid) has the first 4 digits being the order number, and the proceeding numbers be the Product ID (pid). Use pidFormat from lab2 to define the pid format. Call parts available to identify if each part is available and if ALL PARTS ARE AVAIALABLE then call requestParts. Finally call createInvoice and show Invoice with the results of requestParts.

pidFormat

See Lab2 for Requirements

Exceptions:

- CarCompanyNotFoundException – “Car Company with ID ” + first_index + “ not identified within our records”
- InvalidPIDException – “PID “ + pid + ” is not valid (check your length).

partAvailable

Type: Public Static
Input: String p – part String (Body, Exterior….), String id – request id
Return: Boolean – If part is available
Description: Check SupplyDump attribute Stock to see if the part p with id is available. If the part is available return True, if the part is out of stuck return False.
Exceptions:

- PartNotDefinedException – “Part ” + part_name + “ is not defined within our stock”
- PartIdNotDefinedException – “Part ” + part_name + “ with id “ + part_id + “ is not defined within our stock”

requestParts

Type: Public Static
Input: String pid – Product Id
Return: ArrayList – all part objects
Description: Removes all items from SupplyDump attribute Stock and loads all parts into an ArrayList. Once all Items are pulled from stock the ArrayList is returned.
Exceptions:

- OutOfStockException – “Part ” + part_name + “ is out of stock. This was not caught during the parts available checks.”

createInvoice

Type: Public Static
Input: ArrayList order – All Parts within Order
Return: Void
Description: Creates and writes to file the order number, the product Id, list each part and the total price (See Example_Invoice.txt). The filename will be the car company underscore (_) the order id as a text file (.txt)

showInvoice

Type: Public Static
Input: String oid - order ID number
Return: Void
Description: Writes to console the order number, the product Id, list each part and the total price (See example output out.jpg).

ingestStock

Type: Public Static
Input: None
Return: Void
Description: Reads from stock.xml file and builds SupplyDump attribute Stock with the current Stock levels.
Exceptions:

- StockFileNotFoundException – “Stock file stock.xml is not found.”
- PartNotFoundException – “Part ” + attribute + “ is not defined within SupplyDump.”

Note: You can use whatever data structure you would like from the following: queue, stack, ArrayList, or HashMap. The XML functionality is already in place you must have ingestStock add each item to your data structure. So, if Body has a quantity of 10 ThatAutoSuperDuperSadanB then there should be 10 Body objects within your data structure with id= 210111, type= body, price= 11000.00, and name= ThatAutoSuperDuperSadanB.

SupplyDumpException.java

See exceptions defined in SupplyDump.java