Welcome to AnyCar Manufacturing Plant

Dear Student,

Now that we have the parts identification and processing portions setup for your Supply Dump Application we now need you to develop a processing application for manufacturing vehicles. We currently have twelve manufacturing stations in our production line. We have 4 body stations, paint booth, seats assembly and a station for powertrain, seats, radio, tires, and rims installation stations, and lastly a single station for processing miscellaneous options cruise control, roof, backup system, and sensor system if applicable. The amount of time a required for each installation will be defined within the updated stock.xml, which you will use to schedule vehicle manufacturing. The manufacturing application Manufactoring.java will be a multi-threaded application that processes each order in the most efficient manor and cannot have scheduling conflicts.

Thank you,
Your Supervisor
AnyCar
Manufacturing Technical Manager

Application Requirements

Manufactoring.java will process the results of SupplyDump.java requests and create multiple threads of Station.java objects. Manufacturing will schedule and facilitate the building of each vehicle part by part. THE BODY MUST BE THE FIRST PART BUILT. After the Body is built your application must schedule available stations to complete the processing of each vehicle. There cannot be more then one vehicle at each station and each vehicle must visit all applicable stations.

Parts.java

Add the following attribute and getters and setters:

- Installed: Boolean – if part has been installed (default: False)
- Cycles: Integer – number of cycles to complete

SupplyDump.java

Changes must be made to request and the cycles attributes must be assigned from the new cycles tag in the stock.xml file.

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 text file of order IDs and calls line by line `buildOrder` to build each order and store the results of returnParts in an ArrayList. Each request should still print to console the results and save the invoice file. Lastly when each OID is processed ArrayList is sent

over to Manufactoring.java. `Request` will continue to prompt the user until the user inputs -1, then request will return.

Order.java

An order will consist of an ArrayList of parts and represents each order being processed.

Attribute:
time: Timestamp – if still processing time equals the start time and if completed processing the total time in seconds to complete.
(https://www.mkyong.com/java/how-to-get-current-timestamps-in-java/)
parts: Arraylist<Parts> - ArrayList of all the parts involved with the build
oid: String - Order Id

Order
Type: public
Input: parts: ArraList<Parts>, oid: String
Return: Order object
Description: Order object initialization

getPart
Type: public
Input: p: String – of part number
Return: Part needed to be processed
Description: Uses a String to retrieve the part needing to be processed

getStationsLeft
Type: public
Input: None
Return: String[] or parts needed to be installed
Description: Returns a list of all parts with installed equal to false

complete
Type: public
Input: None
Return: Boolean if all parts in parts if installed
Description: Returns true the time variable will calculate the total time of processing in seconds and returns true else return false

toString
Type: public
Input: None
Return: String of Order ID the number of cycles required and the total time it r
Description: return if complete string: “Order ID: oid Total time: time Total Number of cycles: cycle_total”
If not complete string: “Order ID: oid Start time: time Stations not complete: stations”

Station.java

Implements Runnable or Extends Thread

Attribute:
Name: private String – Name of the Station
completed: private Stack<Order> – completed orders
queued: private Stack<Order> – orders needing to be processed
Num: private Integer – Number of installs at this station
Kill: private Boolean – When False the thread will exit

Station
Type: public
Input: name: String - name of station, station_type: String - the type of station
Return: Station object
Description: Station object initialization

waitList
Type: public
Input: None
Return: Integer – length of queued
Description: returns the number of orders in the queue

process
Type: private
Input: None
Return: Void

Description: Pops order from queued and accesses the part within the order. Creates a for loop from the number of cycles the part requires, increments num, sets the Part as complete, and pushes the order to the completed Stack.

run
Type: public
Input: None
Return: void
Description: Creates while not Kill that continually checks the queued stack for new orders to process. If there is no new orders then wait until there are more orders.

interrupt
Type: public
Input: None
Return: void
Description: Prints to console: “Station: Name completed Num installations and is now shutting down” then sets Kill to True.

getCompletedOrders
Type: public
Input: None
Return: - Stack completed
Description: Returns the completed stack of all completed orders

addOrder
Type: public
Input: o: Order
Return: void
Description: pushes the order onto the queued stack.

Manufactoring.java

Manufactoring will receive an ArrayLIsts of orders and process each vehicles build. You must Create and start Stations.java objects Body1, Body2, Body3, Body4, Exterior Color, Interior Color, Powertrain, Seat, Radio, Tire, Rim, Miscellaneous. Next iterate through the Arraylist of orders(ArrayList of parts) and schedule each station which MUST START WITH BODY. If no station is available then the order be added to the station with the shortest queue. Once all orders are complete interrupt each thread and exit Manufactoring.java. Each completed order will print to console the vehicles Order ID and the total

Attributes:

Schedule: private ArrayList<Stations> - Represents each station

Manufactoring

Type: public
Input: None
Return: - Manufacturing Instance
Description: Manufactoring.java initilization

Start

Type: public
Input: orders: ArrayList<ArrayList<Parts>> - orders from SupplyDump to be processed
Return: Void
Description: Turns the ArrayList<Parts> into orders, starts processing orders, when an order is complete print the toString of the order object, and when all orders are complete kill all threads and return to SupplyDump

Create_Stations

Type: public
Input: None
Return: Void
Description: Populates Schedule and initiates the threads for each station.

Schedule_Stations

Type: public
Input: None
Return: - Stack completed
Description: Returns the completed stack of all completed orders