James Hendry: 2170283: Programming 3 assignment 1 Documentation

The scenario that I chose to implement for this assignment that will use multithreading is a car factory that makes cars on demand to the specifications of the customer. It will make these cars on factory line that can make as many cars as possible. These threads will be independent from the user interface which will place orders based on the specifications. When the specifications and have been confirmed It will make a car object that has the user who ordered it as its owner.

The factory line will then take build the car. This will run on its own thread. There can be as many threads as the machine allows for.

When the car is built it will be moved to an arraylist <car> object until the user collects it. Where the user will collect all cars that it belongs too.

This program will achieve multithreading by having as many factory Floor threads which are started by invoking the makeCar( Car unbuiltCar ) method in the User Interface class. Allowing for different users to make many cars at the same time.

I implemented multithreading in order to allow multiple users to make their own cars at the same time. It also allows new users and more car orders to be placed at the same time without needing the previous car order to be finished. The car collect method allows users to collect their cars even if they had made an order themselves.

**Classes:**

**User:** Is the user who orders and receives the car. This is the person buy the car.

**Fields:**

ArrayList <Car> uncollectedCars: A car that the user has ordered and built but not collected.

ArrayList <Car> carz: The cars the user has (has collected)

Name: The Users name.

**Methods:**

+collectCar( Car car2BeCollected): adds a car to the carz arraylist.

**User Interface:** Is the class containing the main method and its related methods.

**Fields:**

all fields are private static

ArrayList<User> users: The users on the system. (could be in a database in later versions)

User currentUser: The current User.

String strChoice: the stored input for string-based choices.

Int intChoice: the stored input for int based choices.

**Methods:**

+Main():

First asks for a name for the first user then invokes displayOptions

+displayOptions():

Displays a set of options in a while loop and then invokes methods based on what number is entered in.

-collectCar():

Checks to see if the user has any cars in the currentUser’s uncollectedCars ArrayList then it add them all to the currenUser’s carz ArrayList. If there is nothing in currentUser’s uncollectedCars ArrayList then it states that it has not cars to collect.

-changeUser():

Displays a lists of user and changes the value of currentUser based on the user input

-makeNyuUser():

Asks for a string imput then makes a new user with the input as a name then asks the user if it wants to change to the new user.

-placeCarOrder():

Uses an instance of PartsCatagory and uses it’s array to ask the user what parts they want their car to have then asks for final confirmation and then makes a car object with the currentUser as it’s user and then invokes makeCar(Car unbuiltCar) which starts a factoryFloor Thread that will build the car.

+displayArrayOfOptions( String questions, object[] array ):

Displays a question and an array of options

+displayArrayOfOptions( String questions , ArrayList<user>):

Displays a question and an ArryListOfQuestions.

+buildCar( Car unbuiltCar ):

Starts a factoryFloor thread that builds the car then adds the car its user’s uncollectedCars ArrayList

**Car:** The car that will get built.

**Fields:**

-String carType: What type of car it is.

-String Color: What color the car is.

-ArrayList<String> parts: Bits that the car has, and the factory gets them from an inventory (that in this case is infinite and then puts on the car)

-ArrayList<String> orderdParts: The parts the car has but have yet too be placed on the frame.

Boolean built: Whether the car is made yet.

**Constructors:**

+Car ( ArrayList<String>orderedbits , String typeOCar , String colorOfCar , User theNyuOwner): done for custom built cars.

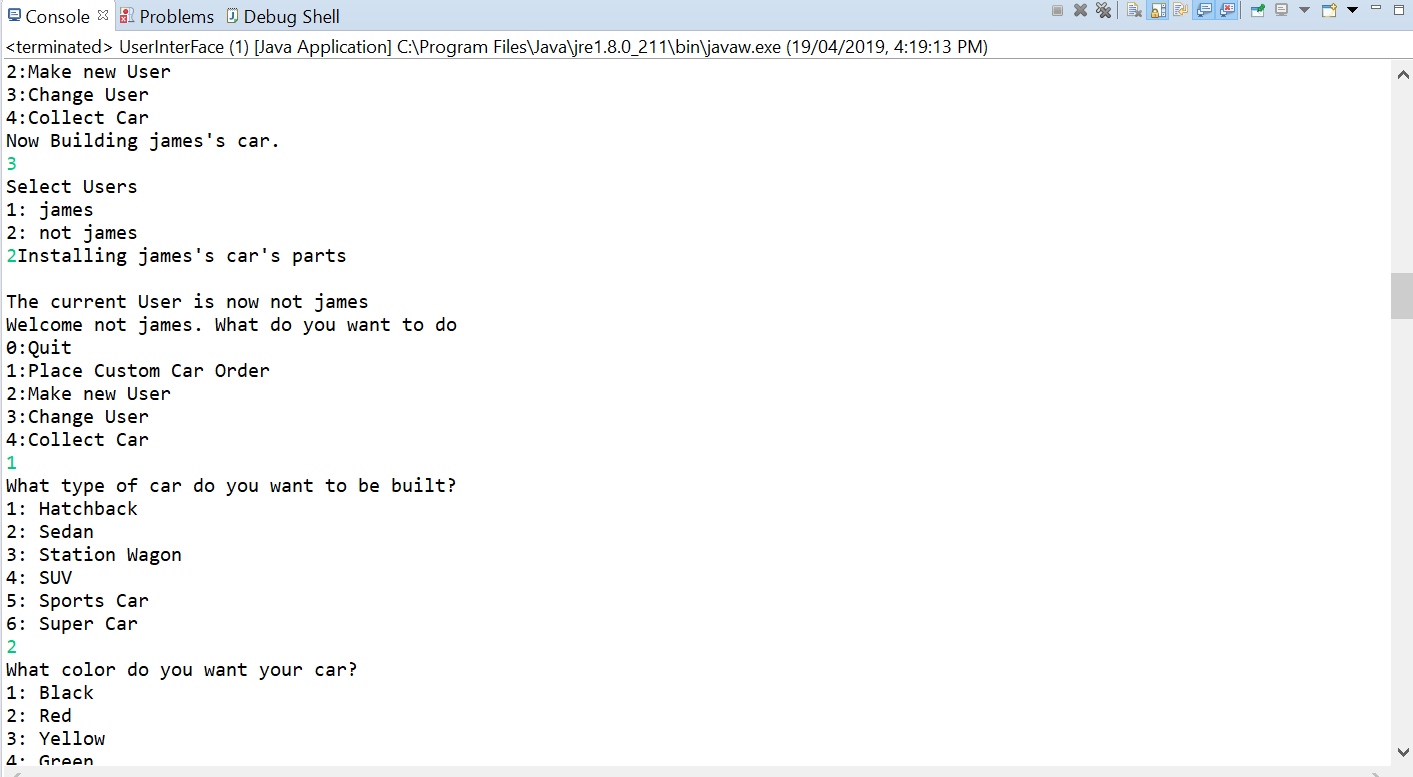
**Methods:**

+discribeCar(): describes the properties of the car.

**PartsCatagory:** A quick way to store all the available options a car can have.

-array String avalilbeParts[]: an array of available parts.

-array String carType[]: an array of available car types.



The program running showing how it is possible to switch users even with a FactoryFloor Thread Running