**CIS 311 Assignment 2**

Kustom Karz, a new start automobile manufacturing company has recently moved from producing orders of single quantities to going after fleet holders. In any case, the existing entry system that allows a user to select the options for their vehicle(s) is atrocious and you have been asked to write a new system.

After discussing the process of how a customer orders a vehicle, you have built up the following tables of information:

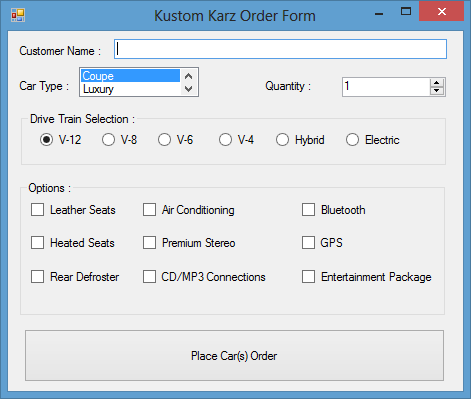
|  |  |
| --- | --- |
| ***Base Car Body Type*** | ***Cost*** |
| Coupe | $10,000 |
| Luxury | $20,000 |
| Sedan | $17,000 |
| Sports Edition | $25,000 |
| SUV | $27,000 |

|  |  |
| --- | --- |
| ***Powertrain Options (available on every base car type)*** | ***Cost*** |
| V-12 | $7,500 |
| V-8 | $2,500 |
| V-6 | $1,000 |
| V-4 | $500 |
| Hybrid | $3,000 |
| Electric | $6,000 |

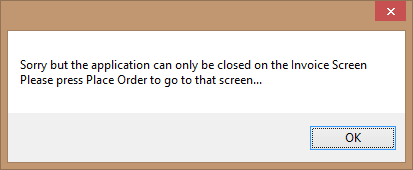
Kustom Karz also makes the following options available, from which a customer can choose as few or many as he/she chooses – each option selected costs an additional $750.00:

* Leather Seats
* Heated Seats
* Rear Defroster
* Air Conditioning
* Premium Stereo
* CD/MP3 Connections
* Bluetooth
* GPS
* Entertainment Package

About the only real limitation that Kustom Karz places on a customer is that if they want to order a fleet of vehicles (e.g. multiple), all vehicles must be configured exactly the same (hence the term fleet). So after some consideration, you have decided that you want to build a custom order entry screen that looks like the following:

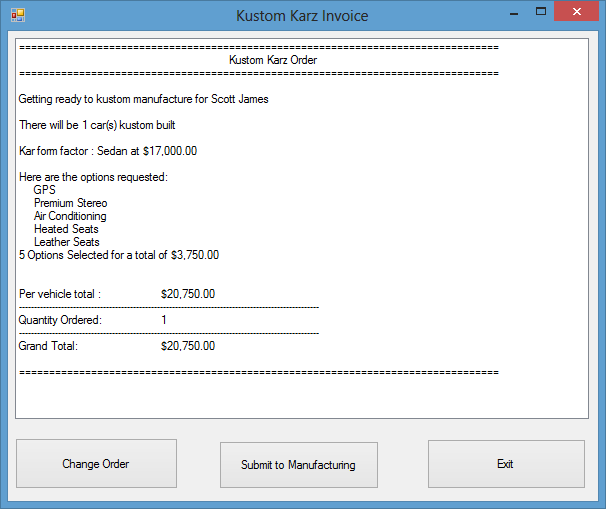


Kustom Karz found out that the previous software system that they used caused a lot of frustration for their customers. They want users to be placed initially on the order entry screen – no user should be able to quit the application from this form! If the user tries to close the order entry screen, he/she will receive the following message:

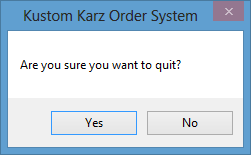


Once this message is dispatched, the user is placed back on the existing order entry form.

If the user presses the “Place Car(s) Order” button, he or she is taken to the second form, which is an invoice form:



This is a complete invoice of the order. If the user examines the invoice and realizes that there are any changes that need to be made, he/she can press “Change Order” and be taken back to the first form, which will still have the information initially entered sitting in it. Changes can be made and then the user can press “Place Car(s) Order” again to return to this screen. If everything is okay, then the user can press the “Submit to Manufacturing” button which would send the order to the manufacturing floor to start the actual building (you don’t need to do anything about this), then the application returns back to the Order Entry screen form which has been completely reset to let another customer interact with it. When the user is on the invoice summary screen and presses either the “Exit” button or the X in the upper corner of the Invoice screen, he/she should be prompted as to whether or not the application should really exit:



If “No” is pressed, the application continues on from where it was prior to the button being pressed. If “Yes” is pressed, the application ends.

A couple of things to think about: (1) there is no calculate button for the invoice – as soon as the user sees the form, the values should be there for them – you cannot use the Load handler since it will only execute the first time the form is loaded, e.g. no ability to jump back and forth between the two forms; (2) since the user exits on the invoice form, it will close, but the original form will still be running and hidden – you will need to figure out how to stop the original form at that point to get your application to actually shut down; (3) the powertrain type has not been calculated into the shown cost – you need to include it in your program! Have fun – think and play about how the two forms have to interact! This will aid in your understanding of Visual Basic’s messaging queue.

Complete your assignment and place your entire solution in a zip file, which you will upload to Canvas. Turn in a cover sheet, your program source code and screenshots of your program’s execution stapled together in that order in class.