Lab Assignment 6: Car Invoice System

This lab focuses on database operations and advanced Form features. Please look at the sample application for an example of this applications functionality. The database and connection string file are included in the zip file.

Preliminaries

Create and add the following objects to your project. This will include all the skeleton objects that you need to complete your project.

1. Create a project called CarInvoice.
2. Rename Form1 to frmMain.
3. Add a Form called frmGetCustInfo.
4. Add a Form called frmGetCustID.
5. Add a Class called FormConfig.
6. Add a Class called CarDB.
7. Add a Class called Customer.
8. Add a Class called Invoice.
9. Add a Class called Car.
10. Add a Class called Model and another called ModelList that inherits a List of Model.
11. Add a Class called Package and another called PackageList that inherits a List of Package.
12. Add a Class called Feature and another called FeatureList that inherits a List of Feature.
13. Add a Class called Color and another called ColorList that inherits a List of Color.
14. Add a Class called Interior and another called InteriorList that inherits a List of Interior.

Code Business Classes

The business classes include classes that mirror the database tables to ensure compatibility between your application and the database. I suggest using auto-implemented properties for all primitive data types. Make sure to add sufficient comments to describe your code.

Code the Customer Class

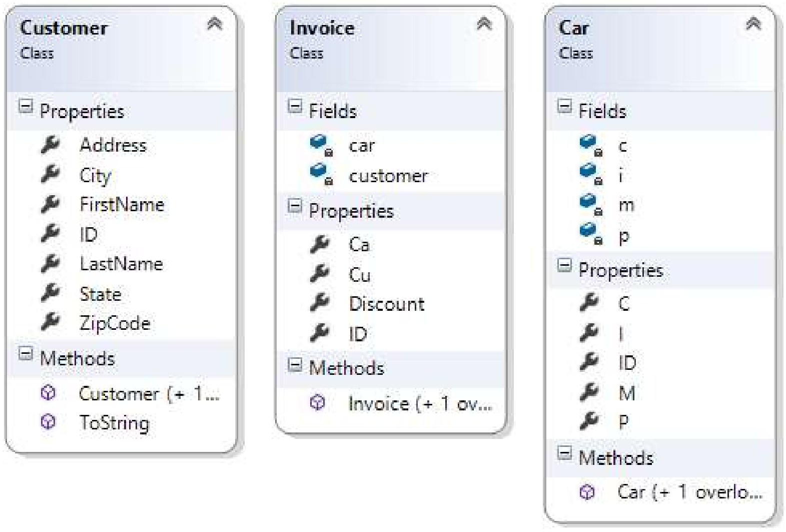
Add fields, default and parameterized constructors, properties for all fields and a ToString method to print each field on a new line (make sure to use "\r\n" for TextBox compatibility). The fields and data types are defined in the database and shown in the figure below.

Code the Invoice Class

Add a Car object, Customer object, integer ID and double Discount as data to the Invoice class. Add default and parameterized constructors and properties (the object properties just get and set the objects). The fields and data types are defined in the database and shown in the figure below.

Code the Car Class

The Car class has 4 objects and an integer as fields. Add the Model, Package, Color and Interior objects as fields, an integer ID and default and parameterized constructors, and properties. The fields and data types are defined in the database and shown in the figure below.



Code the Model and and ModelList Classes

Add the fields for the Model, default and parameterized constructors, properties and a ToString method to print all the Model data on one line. The coding for the ModelList is complete because it inherits List of Model. The fields and data types are defined in the database and shown in the figure below.

Code the Package and PackageList Classes

Add the fields for the Package, default and parameterized constructors, properties and a ToString method to print all the Package data on one line. The coding for the PackagelList is complete because it inherits List of Package. The fields and data types are defined in the database and shown in the figure below.

Code the Feature and FeatureList Classes

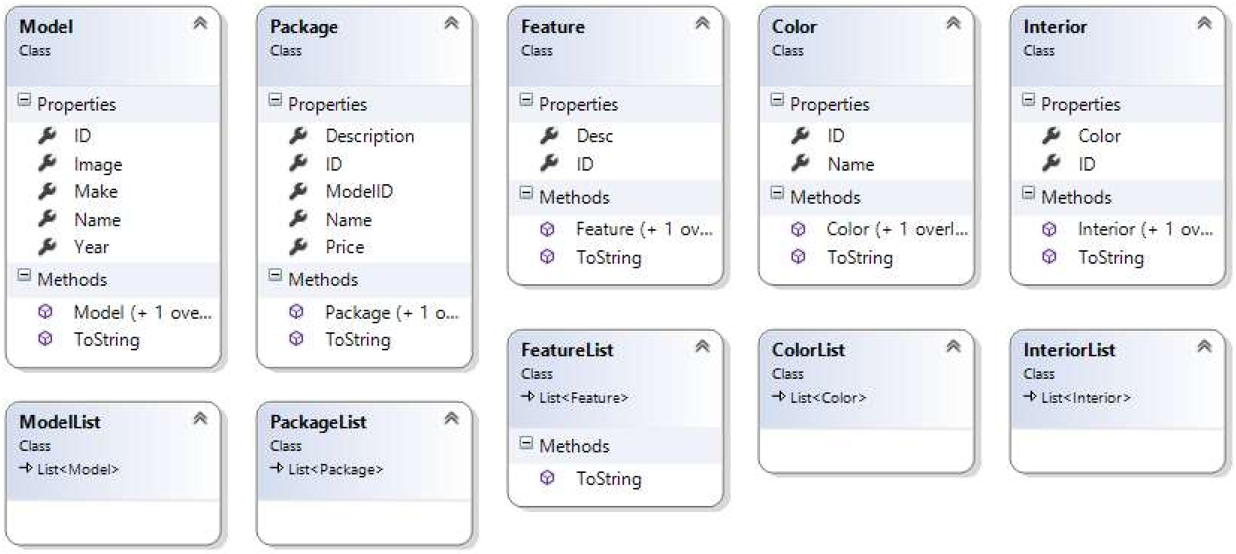
Add the fields for the Feature, default and parameterized constructors, properties and a ToString method to print all the Feature data on one line. The coding for the FeatureList is mostly complete because it inherits List of Model. It only needs a ToString method that prints all Features in the List. The fields and data types are defined in the database and shown in the figure below.

Code the Color and ColorList Classes

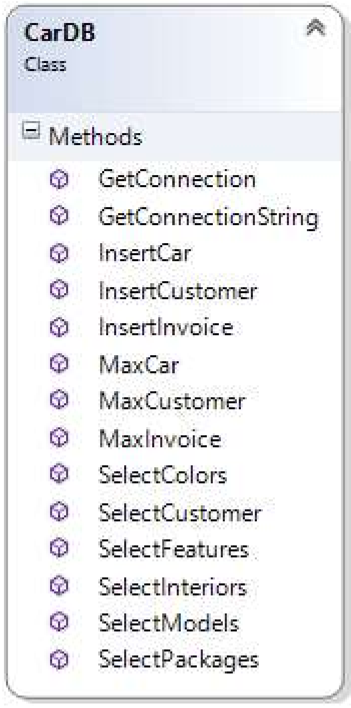
Add the fields for the Color, default and parameterized constructors, properties and a ToString method to print all the Color data on one line. The coding for the ColorList is complete because it inherits List of Color. The fields and data types are defined in the database and shown in the figure below.

Code the Interior and InteriorList Classes

Add the fields for the Interior, default and parameterized constructors, properties and a ToString method to print all the Interior data on one line. The coding for the InteriorList is complete because it inherits List of Interior. The fields and data types are defined in the database and shown in the figure below.



Code the CarDB Class



The CarDB class contains all the database code to connect to, read and write data to the database. The figure on the left shows the methods for this class. I suggest writing one at a time and testing the method for correct output. Make a temporary form to test your database code.

The GetConnectionString method opens the database.txt file, reads all the text in the file into a string and retirns the string.

The GetConnection method gets the connection string from GetConnectionString, uses the connection string to create a connection object and returns the connection object.

The InsertCustomer method accepts a Customer and inserts the Customer's data into the database. Remember not to insert the Customer ID because it is an autonumber field in the database.

The SelectCustomer method accepts a integer Customer ID, queries the data from the database, and returns a Customer object containing the data.

The MaxCustomer method queries the max Customer ID from the Customers table and returns it as an integer. Make sure to use the EcecuteScalar method to query this value from the database.

The SelectModels method returns a ModelList of all Models in the Models table. Try copying the SelectCustomer method and modify the code to return a ModelList.

The SelectPackages method accepts a Model ID and returns a PackageList containing all Packages for the Model ID. Try copying the SelectModels and modify the code to accept the integer ID and query the required data.

The SelectFeatures method accepts a Packagel ID and returns a FeatureList containing all Features for the PackageID. Try copying the SelectPackages and modify the code query the required data.

The SelectColorss method accepts a Modell ID and returns a ColorList containing all Colors for the Model ID. Try copying the SelectFeatures and modify the code query the required data.

The SelectInteriorss method returns a InteriorList of all Interiors in the Interiors table. Try copying the SelectModels method and modify the code to return a InteriorList.

The InsertCar method accepts a Car object and inserts the Car's data into the database. Remember not to insert the Car ID because it is an autonumber field in the database. Try copying the InsertCustomer and modify it to insert a Car's data into the database.

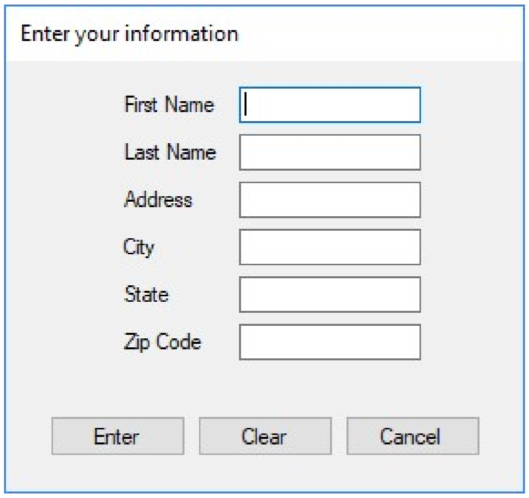
The MaxCar method queries the max Car ID from the Cars table and returns it as an integer. Try copying the MaxCusromer method and modify to return the max Car ID.

The InsertInvoice method accepts an Invoice object and inserts the Invooice's data into the database. Remember not to insert Invoice ID because it is an autonumber field in the database. Try copying the InsertCustomer and modify it to insert a Car's data into the database.

The MaxInvoice method queries the max Invoice ID from the Invoicess table and returns it as an integer. Try copying the MaxCusromer method and modify to return the max Invoice ID.

The frmGetCustInfo and frmGetCustID Forms and Code

The frmGetCustInfo and frmGetCustID Forms get data from a user. The frmGetCustInfo gets the Customer's name , address, etc. The frmGetCustID gets the Customer's ID to include in the Invoice for the Car. You can drag-and-drop components from the Toolbox and use the Properties box to set properties for these forms.



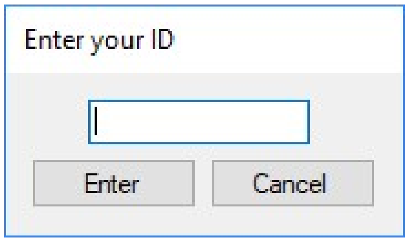
Draw the frmGetCustInfo Form as shown to the left.

The event handler for btnEnter should store a DialogResult OK for the Form, instantiate a Customer, copy the data from the Form to the Customer object, copy the Customer object to the Form's Tag property and close the Form. Set btnEnter to the Form's AcceptButton.

The btnClear event handler should clear all the TextBoxes for the Form. Set btnClear to clear the TextBoxes when Alt-c is pressed.

The btnCancel event handler should store DialogResult Cancel and Close the Form. Set btnCancel to the Form's CancelButton.

Set txtFurstName as the Forms' ActiveControl and the tab order to from txtFirstName to txtZipCode and from btnEnter to btnCancel. Remove the ControlBox and set the BorderStyle to FixedDialog.



Draw the frmGetCustID Form as shown to the left. The btnEnter event handler should store a DialogResult OK for the form, convert the string from txtCustID to an integer, copy the integer to the Form's Tag property and close the Form. Set btnEnter to the Form's AcceptButton

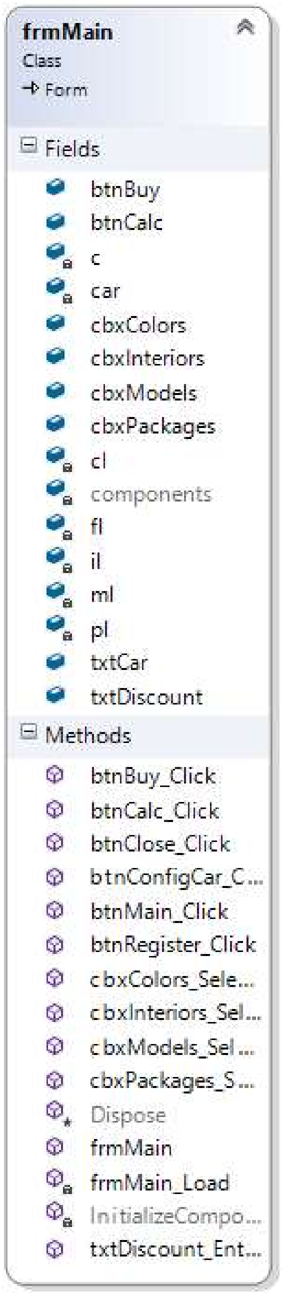
The btnCancel should store a DialogResult Cancel and close the Form. Set btnCancel to the Form's CancelButton.

Set the Form's ActiveControl to txtCustID and the tab order to go to btnEnter, then to btnCancel. Remove the ControlBox and set the BorderStyle to FixedDialog.

The FormConfig and frmMain Code

This class contains two methods. The DrawMainForm method redraws frmMain into the Main Form and the DrawConfigCarForm method redraws frmMain into the Congig Car Form. You can be creative with this or try to mimic the example Forms. The main requirement is that all Form objects must be declared, instantiated, configured and placed on the Form with code and not using the drag-and-drop method. This skill facilitates the use of dynamic Forms, which is good to know for Web Form development. Both forms should be full-screen and have a background image and the Buttons on the Main Form must have background images too. All of the event handler code will have to be on frmMain. The event wiring will have to be for some objects will be in FormConfig and others will be in the frmMain load event handler.

The DrawMainForm Code



The frmMain class will need fields for the Lists and Form objects. If Form objects are not affected by methods on the form, they can be coded in the FormConfig class. Because the Form rendered by the DrawConfigCarForm method has objects that are affected by the code in event handlers, these objects must be declared and instantiated on frmMain, and have their respective event handlers added in the Form load event. The exit and Form navigation Buttons event handlers do not affect objects on the Form and can be declared, instantiated and have events added in FormConfig. The frmMain objects are listed below. The Customer is used on both Form renderings (DrawMainForm and DrawConfigCarForm) and the other objects are used on DrawConfigCarForm.

Add the following objects to frmMain (do not need to instantiate these)

1. ModelList ml
2. PackageList pl
3. ColorList cl
4. InteriorList il
5. FeatureList fl
6. Customer c
7. Car car

Add and instantiate the following Form objects to frmMain. These objects must be instantiated as fields because they are manipulated by event handlers coded in frmMain.

1. ComboBox cbxModels
2. ComboBox cbxPackages
3. ComboBox cbxColors
4. ComboBox cbxInteriors
5. TextBox txtCar
6. TextBox txtDiscount
7. Button btnCalc
8. Button btnBuy

Create a load event for frmMain and call DrawMainForm to render the Main Form. Remember to pass frmMain as this. The main Form has three Buttons that are declared and instantiated in the DrawMainForm method: btnClose to close the Form, btnRegister to open frmGetCustInfo and btnConfigCar to open the form rendered by the DrawConfigCarForm method. Write event handlers for each of these Buttons. They will be wired in DrawConfigCarForm.

The code for btnClose_Click should close the current Form.

The code for btnRegister_Click should:

1. Instantiate and open a frmGetCustInfo
2. If the DialogResult is OK
    a. Copy the Customer from the frmGetCustInfo Tag property (remember to cast)
    b. Insert the Customer into the DB
    c. Get the max ID from the Customers table
    d. Copy the ID to the frmMain Customer
    e. Show the Customer in a MessageBox (need to show the ID so Customer can get to the DrawConfigCarForm).

The code for btnConfigCar should:

1. Instantiate and open a frmGetCustID
2. If the DialogResult is OK
    a. Query the Customer's data from the datebase and copy it to the frmMain Customer
    b. If the Customer is not null
        i. Clear the items in cbxModels
        ii. Get all models from the database and copy to ml
        iii. If ml is not null
            1. Copy all Models strings into cbxModels items.

Write event handlers for the instantiated Form objects on frmMain.

cbxModels_SelectedIndexChanged code should:

1. Change the background image for the form to the correct Car image (it should be one of the fields in your Model class)
2. Insert the Car's Model information into the txtCar TextBox
3. Clear the items in cbxPackages
4. Query the Packages for the current Model from the database and copy to pl
    a. If query is successful
        i. Copy the Package's strings to cbxPackages

Make sure to put default text into all necessary Form Objects and disable subsequent Form objects. cbxPackages should be enabled.

cbxPackages_SelectedIndexChanged code should:

1. Insert the Car's Package information into the txtCar TextBox
2. Query all the Package's Features from the database and copy to fl
3. If Query is successful
    a. Insert all of the Packages Features from fl into the tctCar TextBox

4. Clear the items in cbxColors
5. Query the Colors for the current Model from the database and copy to cl
6. If query is successful
    a. Copy the Color's strings to cbxColors

Make sure to put default text into all necessary Form Objects and disable subsequent Form objects. cbxColors should be enabled.

cbxColors_SelectedIndexChanged code should:

1. Insert the Car's Color information into the txtCar TextBox
2. Clear the items in cbxInteriors
3. Query the Interiors from the database and copy to il
4. If query is successful
    a. Copy the Interior's strings to cbxInteriors
    b. cbxInteriors should be enabled

Make sure to put default text into all necessary Form Objects and disable subsequent Form objects.

cbxInteriors_SelectedIndexChanged code should:

1. Insert the Car's Color information into the txtCar TextBox
2. txtDiscount read-only should be disabled

Make sure to disable subsequent Form objects.

txtDiscount_Enter code should:

1. Clear the txtDiscount TextBox
2. btnCalc should be enabled

Make sure to disable the subsequent Form object.

btnCalc_Click code should:

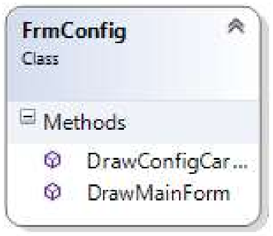
1. Convert the string in txtDiscount to double
2. Insert the discount into the tctCar TextBox
3. Insert the total price before taxes into the txtCar TextBox
4. btnBuy should be enabled

btnBuy_Click code should:

1. Instantiate a Car object and copy the Model, Package, Color and Interior IDs to it
2. Insert the Car into the database
3. Get the max ID from the Cars table and copy it to Car.ID
4. Instantiate an Invoice object and copy the Car and Customer into it
5. Insert the Invoice into the database

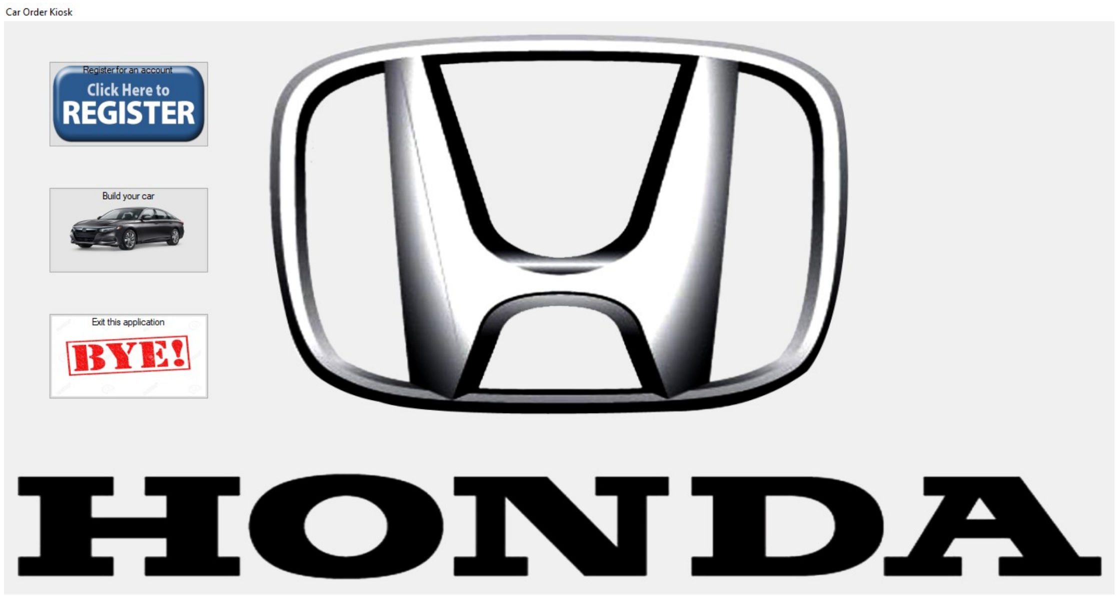
Make sure to put default text into all necessary Form Objects and disable all Form objects. cbxModels and cbxPackages should be enabled.

The DrawConfigCarForm Code

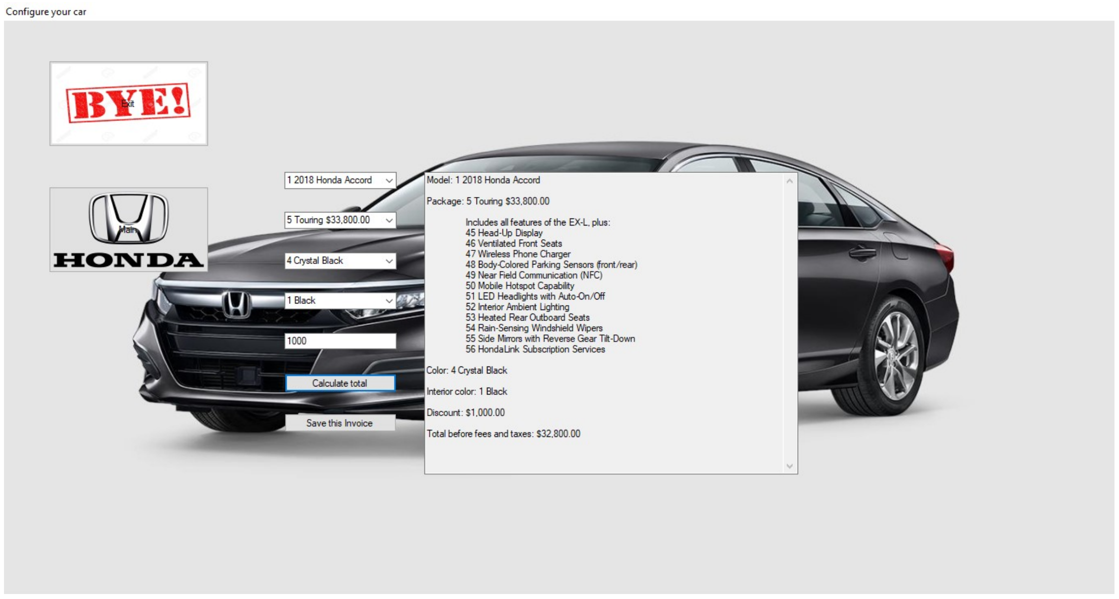


This is where you can be creative. Write the necessary code to draw both forms. Look at the exe in the example I uploaded in the zip file for suggestion on how the interface should work.

Sample Main screen



Sample Car Config screen



```
Model: 1 2018 Honda Accord

Package: 5 Touring $33,800.00

        Includes all features of the EX-L, plus:
        45 Head-Up Display
        46 Ventilated Front Seats
        47 Wireless Phone Charger
        48 Body-Colored Parking Sensors (front/rear)
        49 Near Field Communication (NFC)
        50 Mobile Hotspot Capability
        51 LED Headlights with Auto-On/Off
        52 Interior Ambient Lighting
        53 Heated Rear Outboard Seats
        54 Rain-Sensing Windshield Wipers
        55 Side Mirrors with Reverse Gear Tilt-Down
        56 HondaLink Subscription Services

Color: 4 Crystal Black

Interior color: 1 Black

Discount: $1,000.00

Total before fees and taxes: $32,800.00
```