**CIS 163 Project 3 – A Car Dealer program**

**THIS IS A GROUP (2 members) PROJECT.**

**Due Date**

* See schedule at the end of the syllabus

**Why are we doing this project.**

**The instructor will fill in the details; here are some bullet points**

* Whether autos are being bought/sold is irrelevant, most companies have an inventory of items to track.
* In most projects, partial source code will have been developed, so working with old code is important.
* In most projects today that are small (mid) size, client involvement is a must. In some projects, involvement could be weekly, while other projects could be less frequent. For us in a classroom setting, I expect questions from groups each class period, so I know (as the client) work is being done from the start of the project. (Being completely open here, if questions are not being asked, I will assume work is not getting done, and I would be very concerned if I was paying for this project)
* There are 3 major requirements that are difficult (save/load text, using streams, multi-table display), I am happy to help any group that comes to the office or ask questions in class.
* **This project is challenging**, and is typical for what you will be doing in a company; not an inventory system as in our case, but something that starts with existing code, where your boss (or an actual client) has a set of goals for you to achieve.

**Before Starting the Project**

* Review Chapters 8 - 10 and Chapters 12, 13, 15, 18 of the CIS163 book
* Read this entire project description before starting, if you have any question please ask the instructor

**Learning Objectives**

After completing this project you should be able to:

* Use inheritance and polymorphism
* Use advanced Swing components like JTable and AbstractTableModel
* Save and restore objects using the **Serialization/text** files
* Use Collections.sort
* Using simple Date and GregorianCalendar classes
* Using streams and Lamda functions

**Program description:** Your assignment is to create a program that helps Car Dealers manage their inventory. You should be able to buy and sell cars and trucks from your program. A full description follows.

**A completed program must have the following functionality:**

* Save, load the rental database with serialized files using JFileChooser
* Save, load the rental database with text files using JFileChooser
* Buy/Sell a Car or Truck with a bought date, sell date, name of buyer, price buy/sell name
* Complete error checking
* **And much more! DETAILS BELOW**

**Step 0:** I have provided a partially completed program that will get you started on this project. The intent of the code is to help you understand the different techniques that will be used in your final project. This sample program has many issues, such as, no error checking, it is poorly designed, incorrect results, incomplete results and most importantly does not implement most of the functionality that is required in your project. **In other words, there is much that must be changed on the sample code; the hope is, it will get your team thinking about how best to proceed.**

**Step 1: Create an Eclipse/Intellij project named “RentalPrj”**

* Create a package named: project3
* Create a class named: Auto implements Serializable
* Create a class named: Car that extends Auto
* Create a class named: Truck that extends Auto
* Create a class named: GUICarDealer extends JFrame implements ActionListener
* Create a class named: BoughtCarDialog extends JDialog
* Create a class named: BoughtTruckDialog extends JDialog
* Create a class named: SoldOnDialog extends JDialog
* Create a class named: ListEngine extends AbstractTableModel

**Step 2: Implement the Auto (base class) and using the following:**

*public class Auto implements Serializable {*

*// What is the purpose of this variable (search google)*

*private static final long serialVersionUID = 1L;*

*/\*\* The date the Auto was bought on \*/*

*protected GregorianCalendar bought;*

*/\*\* The Name of the Auto e.g., Ford F150, Pontiac Vibe \*/*

*protected String autoName;*

*/\*\* The price of the auto, i.e., what the dealer paid \*/*

*protected double boughtPrice;*

*/\*\* The trim package of the auto, i.e., lx, ex, sl \*/*

*Protected String Trim;*

*// Create more as needed*

*// add constructor*

*// add getter, setter methods*

**Step 3a: Car is a derived class by extending Auto and using the following:**

*public class Car extends Auto {*

*private boolean turbo;*

*// add constructor*

*// add getter, setter methods*

*// Hint: You will need to override the getSoldBoughtCost function here (more info on this later in this document)*

**Step 3b: Truck is a derived class by extending Auto and using the following:**

*public class Truck extends Auto {*

*private boolean fourByFour;*

*// add constructor*

*// add getter, setter methods*

*// Hint: You will need to override the getSoldBoughtCost function here (more info on this later in this document)*

**Step 4: Implement the class GUICarDealer using the following:**

*public class GUICarDealer extends JFrame implements ActionListener{*

*// declare GUI components (menu items, buttons, etc.) needed*

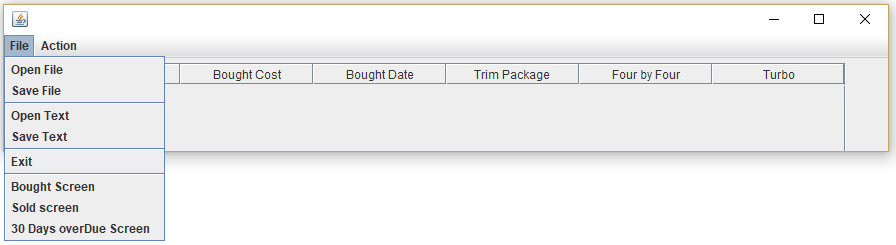
*// constructor method that prepares the GUI*

*// event handlers and other methods needed to build the GUI*

***NOTE: Again, I have provided GUI code (Step 0) to help with all of these sections.***

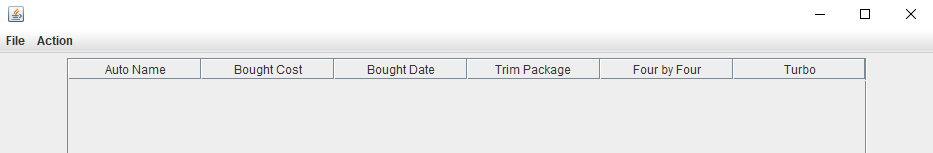
* The GUICarDealer class is the class that displays the GUI to the user and allows the user to buy a truck or car, and allows the user to sell a car or truck. In addition, the GUI allows the users to save and load the database using serialized **and text files**. The GUICarDealer must handle the following operations shown below. The first screen shot shows the main GUI screen:

***This screen shows the JMenu items (Open file, Save file, Open Text …)***

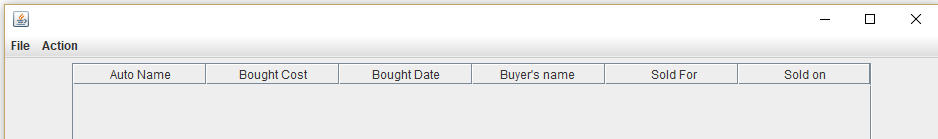


* (File Menu: Open/Save) have read and write operations to save or load the database to/from a Serializable and text files **(THIS IS DIFFICULT).** You must use a JFileChooser to select the file during read/load and write/save operations.
* (File Menu: Screen) Your program has three different table layouts (Bought Screen, Sold Screen, 90Day Screen), shown below: More information regarding these screens later in the document.

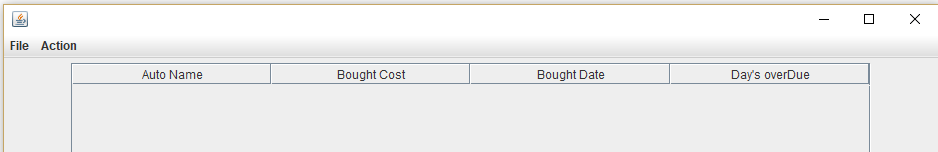
***This screen the Bought Screen:***



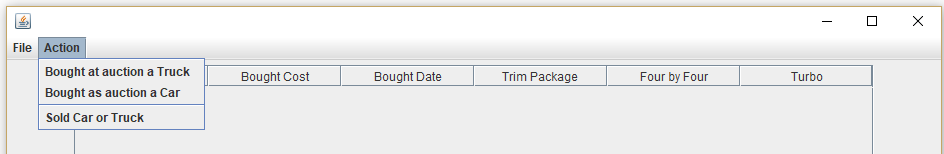
***This screen the Sold Screen:***



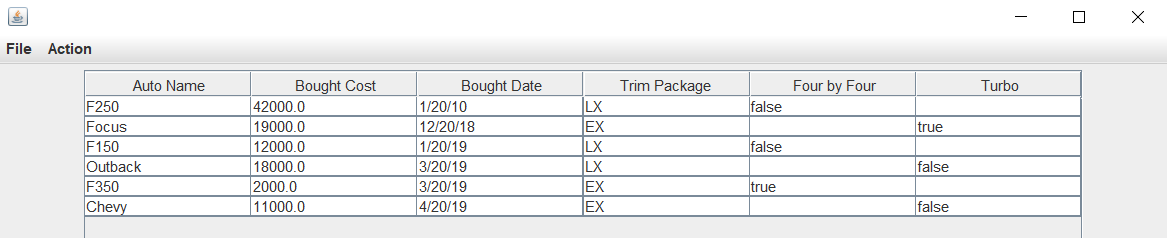
***This screen the Over Due Screen:***



* This Action menu allows the user to sell/buy cars and trucks (step 5, 7 for buying and selling)



**Finally, here an example Screen with some data attached.**

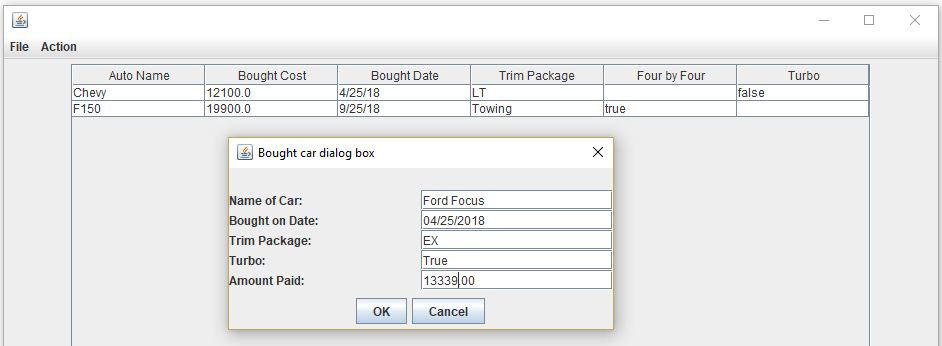


**Step 5: Implement the JDialog classes (BoughtCarDialog, BoughtTruckDialog) to buy a car or truck using the following:**

These dialog boxes are invoked when the user selects buy Truck or Car (see above). The only difference between the two dialog boxes turbo for a car and fourbyfour for a truck. When a JDialog box appears, have today’s date in the JTextField for bought on. The following is some coding help with the GregorianCalendar and Date classes.

*Date date = Calendar.getInstance().getTime(); // Today’s date*

***NOTE: Again, I have provided code to help with this section.***

 Here is a sample screen shot of a boughtCarDialog: (The data is just an example)

**Step 6: Implement the class ListEngine using the following:**

This class is used for storing the auto units (Car and Truck) into an ArrayList<Auto>. (Note: Auto is the base class and review chapter 9 of your book). The functionality of this class is similar in concept to code presented in chapter 9, specifically, the staffList array. The main difference is that this class must handle all the operations from the GUI class. That is, buy a car or truck, sell a car or truck, save and load, etc.

Note: The following code is just a start; to fully understand how to create the ListEngine see the class notes. Examples of a ListEngine class will be presented in class.

**public class** ListEngine **extends** AbstractTableModel {  
  
 **private** ArrayList<Auto> **listAutos**;

**private** String[] **columnNamesBought** = {**"Auto Name"**, **"Bought Cost"**,  
 **"Bought Date"**, **"Trim Package "**, **"Four by Four"**, **"Turbo"**};

@Override  
**public int** getColumnCount() {

@Override  
**public int** getRowCount() {

@Override  
**public** Object getValueAt(**int** row, **int** col) {

@Override  
**public** String getColumnName(**int** col) {

*// add methods to add, delete, and update.*

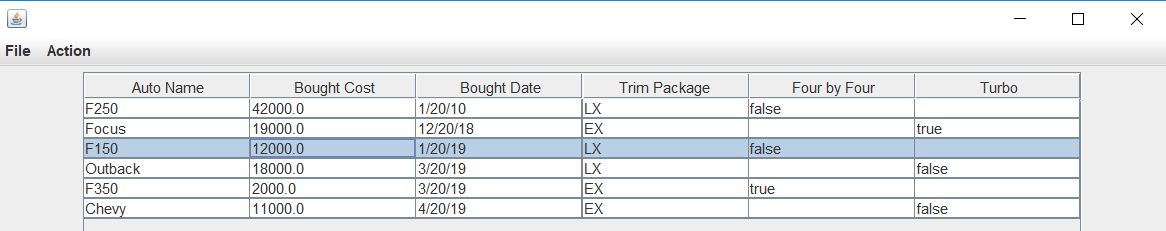
*// add methods to load/save accounts from/to a binary/text file*

*// add other methods as needed*

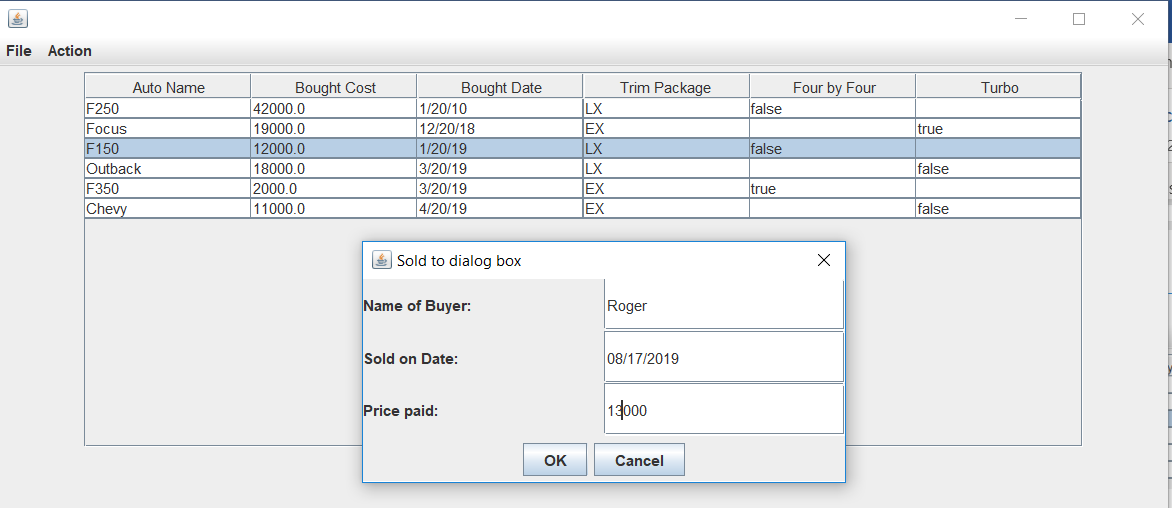
**Step 7: Implementing the sell of a car or truck function:**

***There are 3 steps to sell a auto. 1) select the auto, see below, then (2) a dialog box appears, and final (3) a information dialog box appears.***

***1) First select the auto to sell***

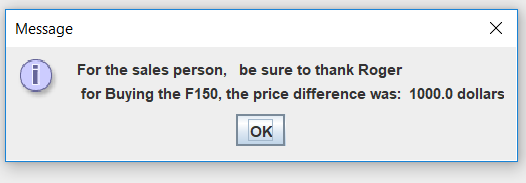


***2) A sell dialog box appear***



***3) A resulting informational dialog box appears***

* Thank you dialog box with the following information.



* Use a JOptionPane.showMessageDialog to output this box.

Important: to accomplish this, create a method (GetSoldBoughtCost )in the Auto base class called

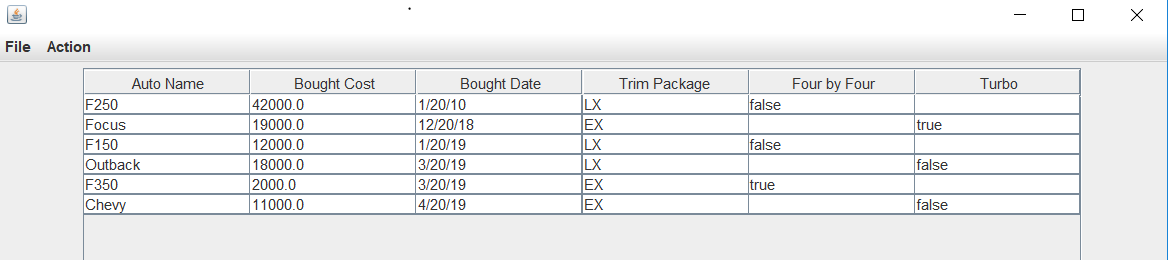
**public abstract double** getSoldBoughtCost(GregorianCalendar SoldDate, **double** SoldCost); which returns the difference between the bought and sold price.

**Step 9: TOTALLY error checked your program (the whole program). For example: Sold Date was before bought date; improper date such as: “abc/abc/abc”; etc.**

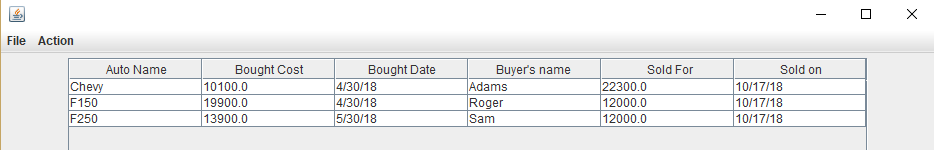
**Step 10: More information regarding Bought, Sold, and Overdue screens.**

* Each screen should be sorted, and the user can buy or sell an auto only on the bought Screen.
* For example:

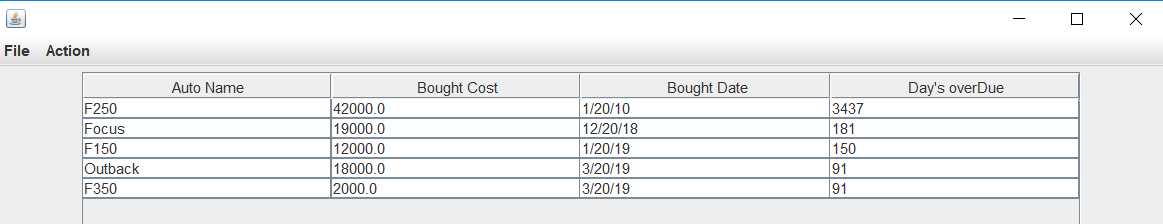
**Bought Screen (sorted on Bought using a Lambda function): In your code use a yellow highlighter to show your lambda function. Note: the table columns**



**Sold Screen (sorted on Buyer’s Name using a Generic function): In your code use a yellow highlighter to show your Generic function. Note: the table columns**



**OverDue Screen, list all the auto (sorted by overdue days) that have not sold, and were bought 90 or more days ago. (sorted on days over due), For example: (uses today’s date)**



**Solve this using any technique, use a yellow highlighter to show your work. Note: the table columns**

***Important: There is always just one dialog box at a time, that is, each screen replaces the other. See demo!***

**Important, I will have test data to be used for this project during demonstrations using data found at the bottom of the list engine.**

--------------------------- YOU’RE DONE ☺ -------------------------------

**CIS 163 – Computer Science II**

**Project 3: “Car Dealer” Program**

|  |  |
| --- | --- |
| Student Name |  |
| Date Submitted, Days Late, Late Penalty |  |

|  |  |  |
| --- | --- | --- |
| **Graded Item** | **Pts** | **Points Secured / Comments** |
| Javadoc Comments and Coding Style/Technique  (<http://www.cis.gvsu.edu/studentsupport/javaguide>)   * Code Indentation (auto format source code in IDE) * Naming Conventions (see Java style guide) * Proper access modifiers for fields and methods * Use of helper (private) methods * Using good variable names * Header/class comments * Every method uses @param and @return (1 sentence after) * Every method uses a /\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* separator * Overall layout, readability, **No text wrap** * Using /\*\* … / for each Instance variable * Has many inner “inner” comments | 10 |  |
| **Steps 1 – 7: Basic Functionality**   * Sell an Auto functionality * Buying a truck or car functionality * Error checking * Load and Save using text files | 10  10  10  15 |  |
| **Step 8 – 10:**   * Sorting for each screen * One Screen | 18  12 |  |
| **Cleaning up the existing code that you used in your project. In other words, have a good design, no wasted lines of code, No extra code. etc.**  **Misc. issues found.** | **10**  **5** |  |
| **Total** | **100** |  |

**Comments: (extra credit)**