**CIS 163**

**Project 4 – Modifying existing code**

**(A common first job as a developer)**

**Due Date**

* See schedule at the end of the syllabus

**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:

* Single Linked list implementation
* Understand how to code: Generics
* The ability to examine and figure out existing code and modify that code.

**Program description:** Your first task as a new programmer in a company will typically be modifying existing code. With that in mind, our assignment is to change the Car Dealer demo program (provided, named: GUICarDealer) so the program uses a Linked list and other new functionality. **You cannot add any additional instance variables to the classes provided without the instructor’s permission, specifically, the Node.java and MySingleLinkedList.java**

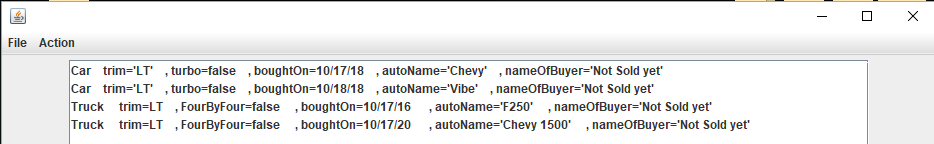
**Step 0: Start with the code provided and figure out how this code functions (take an hour or two)**

**Step 1: The existing code uses an ArrayList, change the following:**

1. In the list Engine, change All Arraylist to MySingleLinkedList. **(USE a single linked list with a tail).** Once you have made this change, you will see syntax errors in every place an ArrayList method was being used.
2. To remove these syntax errors, complete the methods found in MySingleLinkedList class and any additional method you need. You cannot add any additional instance variables to the classes provided, that is, the Node, and/or the MySingleLinkedList, classes.

* **Requirement:** When you re-write the add method, you are required to sort by Cars first and by truck second, in the Car section sort by date bought, and same is true with the truck section.

For example:



1. Create JUnits that show your MySinleLinkedlisL is functioning correctly with 100% coverage on MySingleLinkedList. The file mySingleLinkedListTest.java has been provided as a starting point.

A very small example has been included in this test file to help.

1. In this step, create an **undo** function (place a JButton on the panel). The undo button should take the user all the way back to the start of your program, that is, multi-undos. Further, the undo only needs to undo inserts (buys) and removes (Sells) and not load (or save.)
2. Final step, this is a very difficult step and you will need to see the instructor before attempting this step.
   1. Make your linked list a total generic linked list.
   2. Allow the undo function to undo a load operation in one step. Again, see the instructor regarding this final step.

**CIS 163 – Computer Science II**

**Project 4: “modify existing code” Project**

|  |  |
| --- | --- |
| 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 |  |
| Functionality of your program   * Step 1, MySingleLinkedList * Step 2, add method that sorts by car, truck * Step 3, Junit testing. * Step 4, Simple undo * Step 5, More Complex undo, and generic MySingleLinkedList (must see instructor regarding this step) | 20  20  20  20  10 |  |
| **Total** | **100** |  |

**Comments:**