**CIS 163**

**Project 3 – 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

**Why are you doing this assignment**

* This project will help you develop complex methods, that is, the add method will be very complex to implement.

**Learning Objectives**

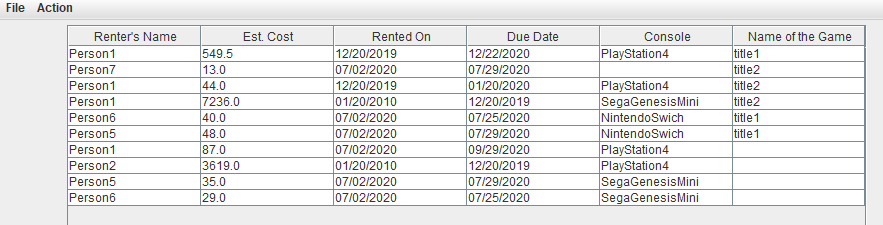
After completing this project, you should be able to:

* Implement a Double Linked list
* 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 Rental System program, 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 MyDoubleWithOutTailLinkedList.java**

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

**Sample output:**

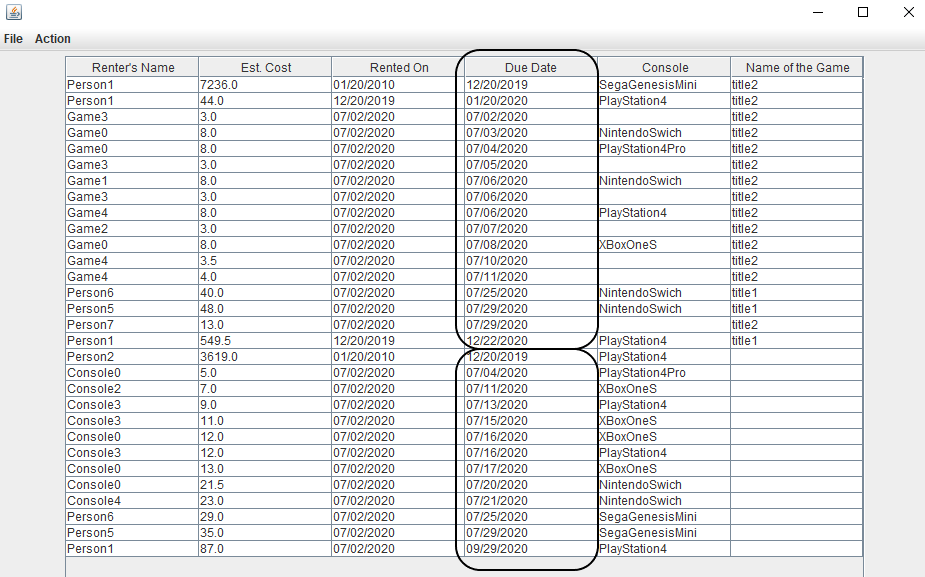


**Step 1: The existing code uses an ArrayList, change the following:**

1. In the list Model, change All Arraylist<Rental> to MyDoubleWithOutTailLinkedList (USE a Double linked list without a tail). Once you have made these changes, you will see NO syntax errors in every place an ArrayList method was being used. **However, you program will not function and your task is to complete steps 2, and 3. From this point on, you are not permitted to change the ListModel class.**
2. Complete all the methods found in MyDoubleWithOutTailLinkedList class and any additional methods you need. You cannot add any additional instance variables to the classes provided, that is, the DNode, and/or the MyDoubleWithOutTailLinkedList classes. You are NOT permitted to change the GUI and ListModel class.

**Requirements for step 2:** When you write the add method, you are required to sort by Games first (ordered by dueDate) and by Consoles second (ordered by dueDate). For this step 1, you need not worry about two dueDates dates being equals. (See the final step regarding a change in this requirement).

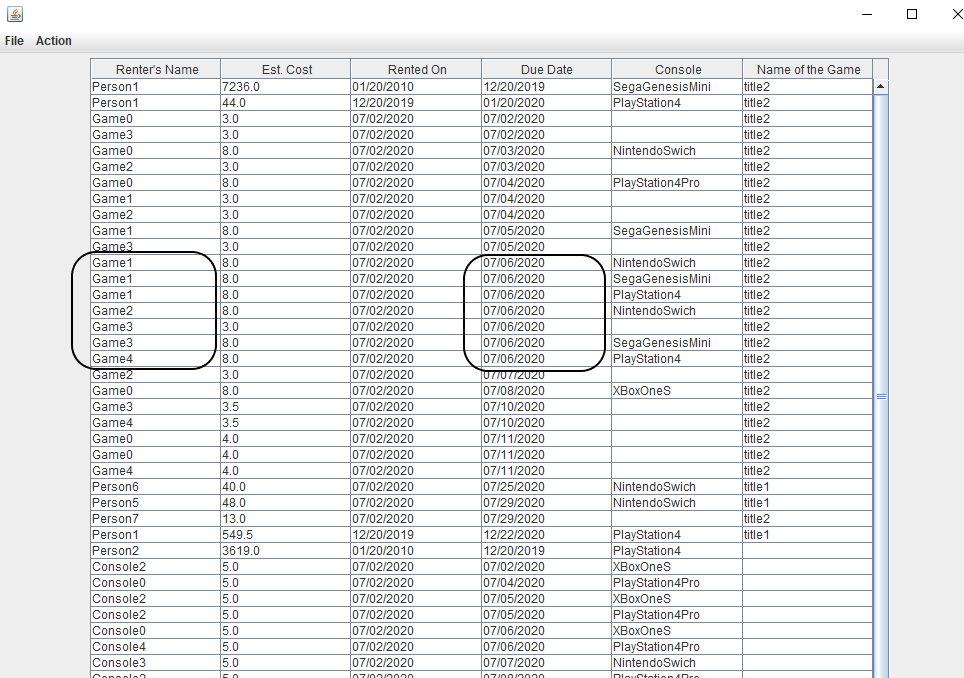
For example:



1. Final step, this is a difficult step and you will need to see the instructor before attempting this step.
   1. Handle equals cases in the add method. In other words, if two dates are the same then sort by renter’s name.

Important Notes: If your GUI does not work after you have completed the add, remove, and get methods… then it means your methods are not working properly! I suggest trying this in the createList Method in ListModel, by replacing the lines from (approx.) 272 to 274 with:

add(game1);  
add(game4);  
add(console1);  
listOfRentals.remove(0);  
add(console4);  
add(game5);  
add(game2);  
listOfRentals.remove(listOfRentals.size()-1);  
listOfRentals.remove(2);  
add(game3);  
add(console5);  
add(game6);  
add(console3);  
listOfRentals.remove(listOfRentals.size()-1);  
add(game7);  
add(console2);  
for (int i = 0; i < listOfRentals.size(); i++)  
 System.*out*.println(listOfRentals.get(i).toString());



**CIS 163 – Computer Science II**

**Project 3: “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 2, MyDoubleWithOutTailLinkedList   Remove  get   * Step 3, Add with Duplicates   (partial credit for this step.) | 35  20  15  20 |  |
| **Total** | **100** |  |

**Comments:**

**Each test case below gets more and more difficult and will be done in sequence.**

Level 1: Testing

1. Debugger shows the correct list but GUI does not:
2. Count is set to 0 at line approx. 290 (while < 0) to see if 4 rentals are displayed on GUI.
3. Count is set to 30 at line approx. 290 (while < 30) to see if all rentals are displayed.

Level 2: Testing

1. Count is set to 30 at line approx. 290 (while < 30) to see if all rentals are displayed. (several attempts)

Level 3: Testing

1. Count is set to 30 at line approx. 290 (while < 30) to see if all rentals are displayed and checkout dates are equal and sorted by name.
2. Count is set to 30 at line approx. 290 (while < 300) to see if all rentals are displayed and checkout dates are equal and sorted by name. (several random attempts)