**Test your program from Homework 2, using the class from Homework 1.**

**In your documentation answer, at what point does your program performance begin to degrade?**

/\*\*

\* This class will implement a generic double linked list, and then use it to test my program

\* from homework 1.

\*

\* Name: Nabeel Hussain

\* Class: CMSC350

\* Professor: Didier Vergamini

\* Homework 2

\* Date: 11/12/2016

\*

\* **@author** Nabeel Hussain

\*/

My program from Homework 1 will now use the **LinkedListUMUC <T>** class I implemented for this homework, to hold the employees that are added to the company. It will then sort the employees onto a GUI output text area by different key elements (first name, last name, position, pay, etc…).

My program now has four classes. The first class is the data structure class LinkedListUMUC <T>, which will implement a generic doubly linked list. This class will be tested using my homework 1 program, where the ArrayLists I was using to store the employees, first names, last names, job titles, ages, etc… will now be stored using Linked Lists.

The second class is the data element class Employee, which holds the information required for each employee of the company. This includes an employee’s ID, first name, last name, job title, hourly pay, number of years worked, age, and gender.

The third class is the data manager class, which holds the methods that will be used by my GUI class to perform the different operations necessary by the application. This will include reading in a file and extracting employee information from it. It will also include writing the current list of employees of the company into a file. It will also include, a method to make random instances of the employee class. This class will use the LinkedListUMUC <T> class to manage and store the employees.

The fourth class is used for the GUI, which will contain the buttons and a text area output to meet the functional requirements for this assignment.

Text files to be used to run my **program:**

Read in File: employeeReadFile.txt

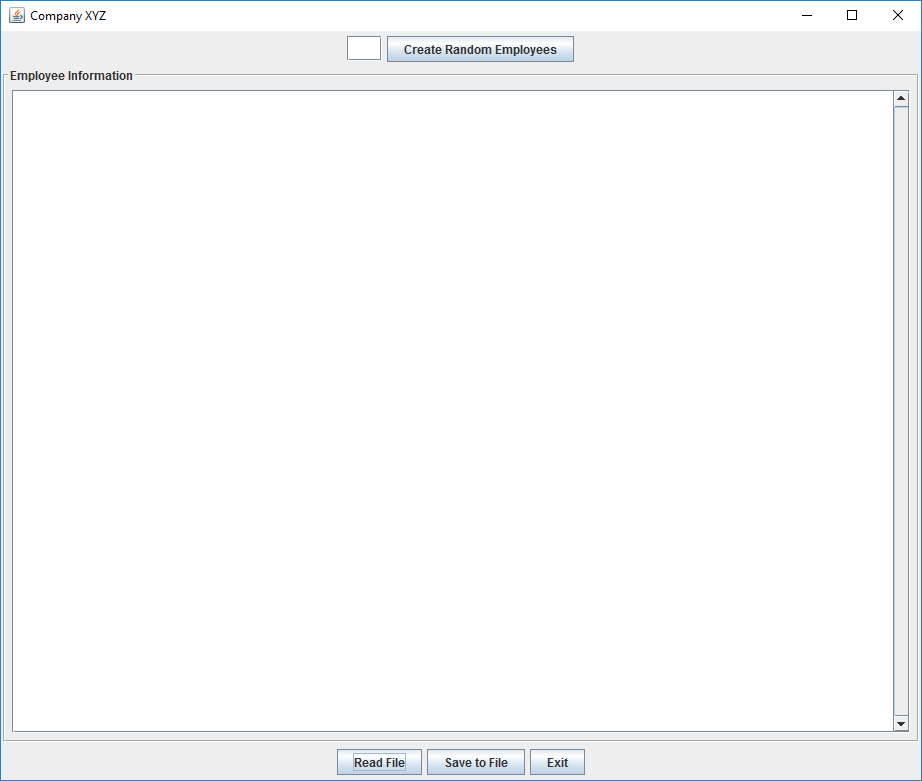
Save to File: employeeWriteFile.txt

Text files to be used with my **Junit tests:**

Read in File: TestReadFile.txt

Save to File: TestWriteFile.txt

The application must always read in a file first, in order to create a pool of names and employee positions, for when the “Create Random Employees” button is selected.

My GUI:

**Test Plan:**

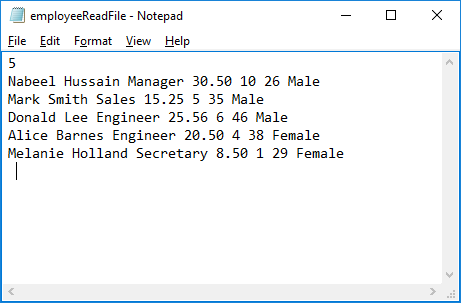
|  |  |  |  |
| --- | --- | --- | --- |
| **Test Case** | **Input** | **Expected Output** | **Did Test Pass?** |
| 1 | Reads in the .txt file: employeeReadFile.txt | A string that has been formatted to display all the employee’s information in the GUI text output area. | Y |
| 2 | Reads in the .txt file: employeeReadFile.txt  Create Random Employees: 20 | 20 randomly created employees, that will be added to the current list of employees. All the employees will be display in the text area box of the GUI. | Y |
| 3 | Reads in the .txt file: employeeReadFile.txt  Create Random Employees: 20  Writes to the .txt file: employeeWriteFile.txt | The full list of employees written into the selected file. The first line should contain the number of employees in the list. The following lines will include the information of each employee per line.  Example of how the file should look like:  25  Nabeel Hussain Manager 30.50 10 26 Male  Mark Smith Sales 15.25 5 35 Male  Donald Lee Engineer 25.56 6 46 Male  Alice Barnes Engineer 20.50 4 38 Female  Melanie Holland Secretary 8.50 1 29 Female  Etc… | Y |
| 4 | No File initially read in.  Create Random Employees: 5 | JOption Pane Show Message Dialog:  “You must read in a file of employees first, in order to create new random ones.” | Y |
| 5 | Press “Exit” Button | Close GUI Application | Y |

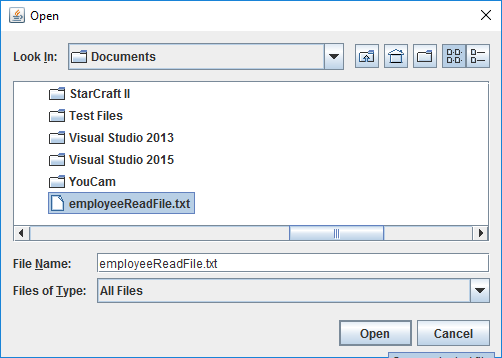
\*Additional test cases are in my **LinkedListUMUCTest.java** and **EmployeeManagerTest.java** Junit test files

**At what point does your program performance begin to degrade?**

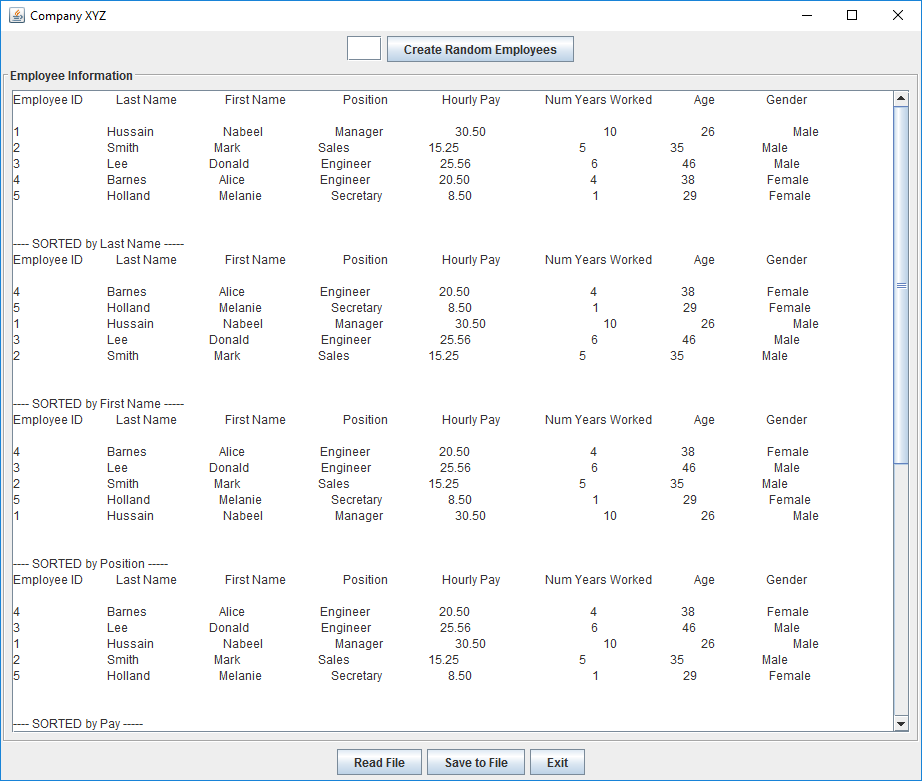
Using my LinkedListUMUC<T> class, the performance will begin to degrade when I try to create more and more random instances of my employees, which will have to be displayed in the output text area in sorted order. This is due to the fact that when my program needs to generate random names, positions, ages, etc.. and also sort the employees by first name, last name, position, etc… then the time it takes to iterate through the list and re-arrange the employees will take O(n) time, which will slow down the performance as n gets larger. I noticed that it started to slow down when I tried to added more than 250 random employees in at the same time.

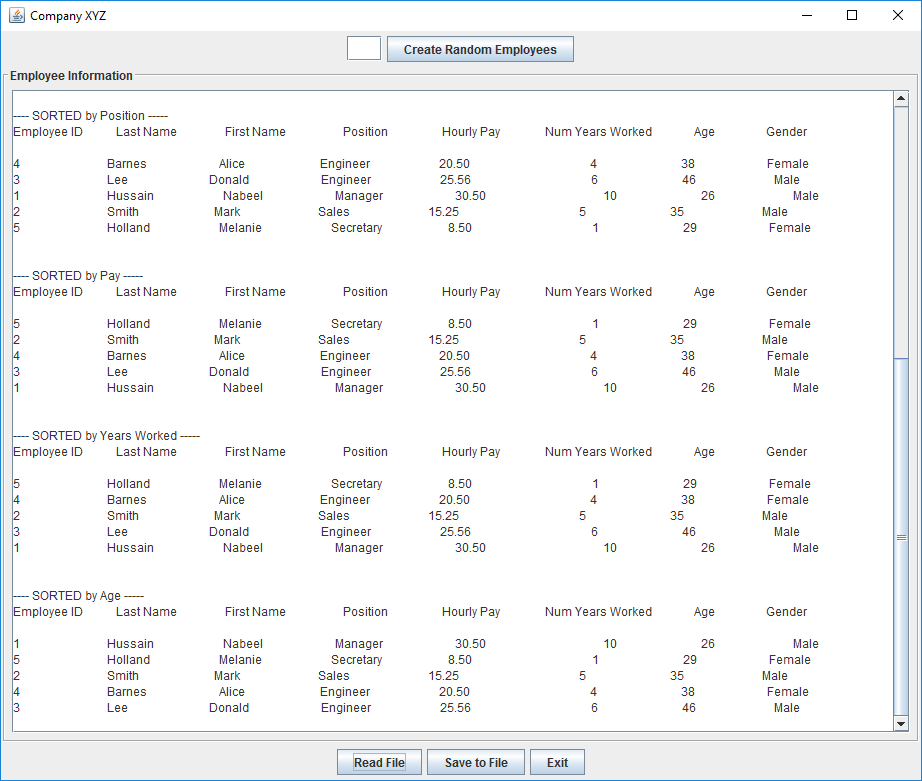
**Screen shots of successful compilation and running for all test cases**





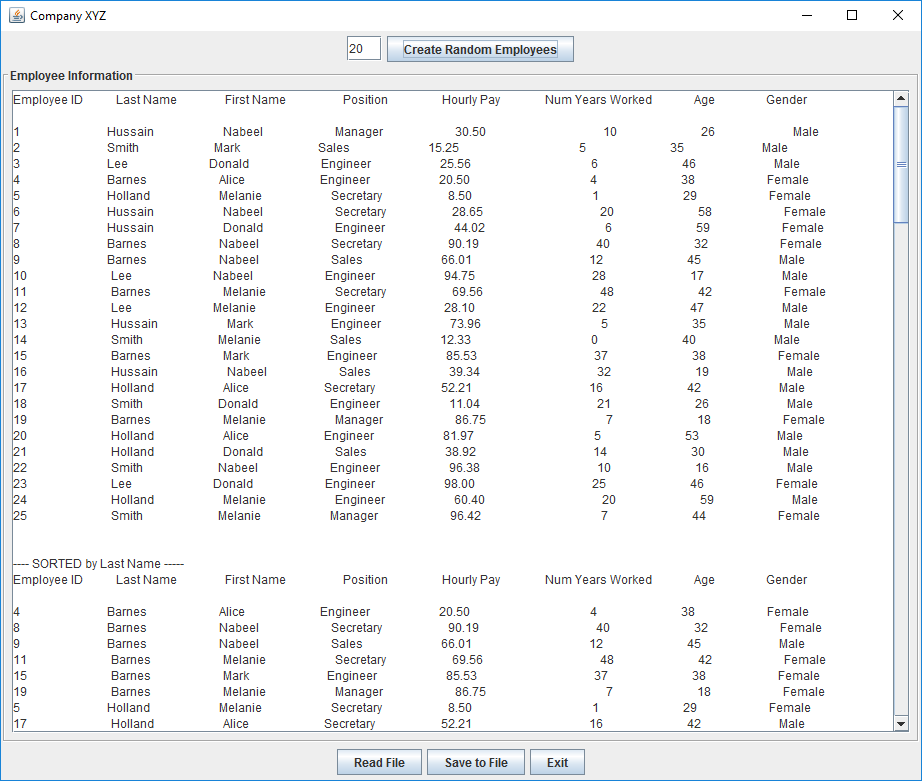
**Displays employee info from the file onto the text Area of the GUI, in various sorted orders.**

****



**Create 20 random employees, which will be added to the current list of employees:**

**The output will show a total of 25 employees now, which includes the 5 from the file read in, and the 20 random ones created.**



**Save the current list of employees by writing them into a selected file.**

