CIS211: Data Structures

Delaware Tech, CT Department

Wilmington

(Instructor: Lee Hsu)

**Project #6: Merge Files**

1. **Introduction:**

In this project, we extend the concepts of sorting and merging into a more practical real world application. Since the Java Collection API provides various built-in sorting capabilities, so we will focus on the application of the concepts instead of the implementation.

The input file that you are going to use is the tab-delimited text file that we used for project 1: **p1arts.txt**. And the output file that you are going to produce is also the tab-delimited text file called **p6sortedArts.txt** which is sorted ascendingly on **artistID** and then **artID**. Example follows:

|  |  |  |  |
| --- | --- | --- | --- |
| ArtistID | ArtID | Title | Appraised Value |
| 1 | 1038 | Spring Flowers | 800 |
| 1 | 1050 | Cattle Ranch | 10000 |
| 1 | 1103 | Trail End | 8000 |
| 2 | 1042 | Coffee on the Trail | 7544 |
| 3 | 1013 | Superstitions | 78000 |
| 3 | 1021 | Bead Wall | 14000 |
| 3 | 1034 | Beaver Pole Jumble | 28000 |
| 3 | 1063 | Asleep in the Garden | 110000 |

1. **Description:**

In business, it is very common to have a huge file that cannot fit into memory entirely. As you have learned through various sorting algorithms that, to sort the data, the data need to be in memory; so in this project, you are going to assume that the maximum size of the memory you can use for sorting data is only ¼ of the size of the file (i.e., there are 116 records in the input file and your RAM can only accommodate 30 records.) And, you may assume that you have more than enough hard drive space.

1. **Assignments:**
   1. Theory: Know how it works -- Spreadsheet
      1. Attached is a spreadsheet file named “p6MergeSort(Start-Up).xlsx. This file is similar to what you did for project 2. The description there is self-explanatory and you are going to complete the entire worksheet.
      2. Change the file name to “**p6MergeSort(yourLastName, yourFirstName).xlsx**”.
   2. Programming Practice:
2. Create a class called **Art** that implements **Comparable** interface.
3. Read part of the file into an ArrayList of Art, use Collectons.sort() to sort the array and then write them to a temporarily file.
4. Read some more records from the file, sort them, and merge them with the sorted file on the disk.
5. Repeat the above step until done.