CS201 – Final Project (Session 2, Fall 2023)

(25% of overall points)

- **Date to release**: Mon, 4th Dec 2022
- **Deadline to submit**: 10:00pm (Beijing Time), Saturday, 9th Dec 2022
- **Format**: a group project, with 2 students in each group

- **Description of the project**

In this project, you are required to develop a java project to process a database file about people's information. The file, "*people.txt*", is downloadable from Sakai. Opening it will show that each line in the file contains a person's record as:

given name; family name; company name; address; city; county; state; zip; phone1; phone2; email; web; birthday

Each field is split by ';'. For example,

Kiley;Caldarera;Feiner Bros;25 E 75th St #69;Los Angeles;Los Angeles;CA;90034;310-498-5651;310-254-3084;kiley.caldarera@aol.com;http://www.feinerbros.com;03/25/1956

1. create in a new java project, "***CS201 Final Project***" and add the following classes:
2. a non-public class, "***PeopleRecord***", a class for people's record as in the file "people.txt". Note that the instances of this class are going to be added (as a node) into a tree structure defined below. Define necessary methods to make it happen.
3. a non-public class, "***MyBST***", a binary search tree class for the people's records. The class consists of a few methods:
    a. ***getInfo*** that returns the total number of nodes and the height of the tree
    b. ***insert*** that adds a node into the tree
    c. ***search*** that takes first/given name and family name as parameters, and returns all nodes/records that have the same names
4. a non-public class, "***MyHeap***", a heap for people's records. Define methods such as insert and remove to add/push a node into the tree or delete/pop a node out of the tree.
5. a non-public class, "***MyHashmap***", a hashtable based map for storing people's records. You must use quadratic probing approach for key collision handling. Define common methods for map such as put, get, search and delete.
6. a public class, "***DatabaseProcessing***"'. The class has a few methods:
    a. ***loadData*** that takes a file's name (with the type String) and loads all data/records into an instance of the class MyBST
    b. ***search*** that takes a person's first/given name and family name as parameters, uses the method search in the class MyBST and returns all records that match the names
    c. ***sort*** that gets all the records out of the binary search tree (MyBST) and insert them into an instance of the class 'MyHeap'. Once the heap is created, use the idea of heap sort to order (sort) all records and a proper data structure to hold the result (after sorting).
    d. ***getMostFrequentWords*** that takes an integer '*count*' and '*len*' as parameters. It returns the top '*count*' most frequently appearing words (whose length is no less than

the '*len*') in the file '*people.txt'* and the frequency of each word. **Note that** (1) the 'word' here only includes those from the fields "*given name; family name; company name; address; city; county; state*" and it must not contain nothing but 26 alphabets; (2) If the parameter '*len*' is smaller than 3, the method prints an error message and throws a customized exception "*ShortLengthException*"; (3) You must use the methods defined in the class '*MyHashmap*'.

e. public static void ***main(String[])*** that tests all methods above using the file "*people.txt*". Print the relevant information to prove that all methods you've developed above give the results as expected such as searching and sorting.

- **Note that:**
  1. You are **NOT allowed** to use java built-in classes LinkedList and PriorityQueue, but you may use other data structures such as ArrayList and Stack.
  2. You may define attributes and the parameters of methods for if they are not specified. Add other methods or attributes for the classes above, or even new classes if necessary.
  3. For the heap class, you may use either the idea of the linked list or array to implement the structure. You may **NOT copy codes** directly from the java built-in class PriorityQueue but borrow ideas from there.
  4. Overall, you have quite a degree of freedom to finish the project as long as the requirements are met.

- **Submission:**
  1. In the Sakai, submit the following files **by only ONE member** of the group:
  2. The java source file, "DatabaseProcessing.java"
  3. A PDF file and/or a video file to show:
     a. What you have completed and what you have not completed (if any), such as classes, methods, testing, etc., including those features/functions that are not specified above.
     b. What testing has been done and the results such as searching and sorting
     c. Individual contributions from each member of the group, and overall ratio (such as 50-50 or 60-40)
     d. The video should be NO more than 90 seconds long, with voice and/or text over

- **Marking criteria:**
  1. *Completeness of the project*: have all tasks been finished? Have the PDF and/or video been produced? **[80%]**
  2. *Additional features of the project*: **[10%]**
     a. Support other types of records apart from 'people', i.e., your classes like BST, Heap or Hashmap are generic
     b. Support additional ways of comparing (thus ordering) other than names? How about using people's birthday?
     c. Able to illustrate the (part of) structure of BST and Heap created for the records in the 'people.txt'? For example, use symbols like '-', '|', '[], etc, or even a graphics library (3rd party plugin).
     d. Able to come with good testing strategies.
  3. *Quality of the programming*, including readability, clarity, sufficient and clear comments, variable definition, length of codes, etc. **[5%]**
  4. *Quality of the documenting* (PDF or video): clear, concise and sufficient details. **[5%]**