CS201 Final Project - People Database Processing

Overview

This project processes a database file (`people.txt`) that contains records of people's information, where each field is separated by semicolons (`;`). The file includes details such as names, addresses, contact info, and more. We developed several data structures in Java to handle operations like searching, sorting, and extracting frequently used words.

Completion

- [x] **PeopleRecord**: A class representing individual records in the file `people.txt` .
  - **Enable birthday sorting**
  - Getter methods for the data.
- [x] **MyBST**: A binary search tree (BST) for managing people's records.
  - `getInfo()` : Returns the total number of nodes and the height of the tree.
  - `insert(PeopleRecord record)` : Inserts a new record into the BST.
  - `search(String familyName, String givenName)` : Searches for all records with matching given and family names.
  - `delete(String familyName, String givenName)` : Deletes a record from the BST.
  - `inOrder()` : Returns a list of all records in in-order traversal.
  - `visualize()` : Visualizes the structure of the BST.
  - `getAllRecords()` : Returns all records as an `ArrayList` .
- [x] **MyHeap**:The MyHeap class is a max-heap data structure which uses an ArrayList to store PeopleRecord objects by name. It includes methods for inserting records (insert), removing the maximum record (remove), and maintaining the heap property through the trickleUp and trickleDown methods. All core functionalities have been implemented, and testing confirmed that records are correctly inserted and removed while maintaining the heap structure.
  - `insert(PeopleRecord newNode)` : Adds a new record to the heap.
  - `remove()` : Removes the top element of the heap.
  - `size()` : Returns the number of elements in the heap.
- [x] **MyHashMap**: The MyHashMap class also uses an ArrayList-based implementation to store PeopleRecord objects by family name. It uses quadratic probing for resolving collisions and contains methods put, get, and delete, with an internal resize function to expand the table when necessary. The implementation is complete, and testing has shown successful insertion, retrieval, and deletion of records.
  - `put(String key, PeopleRecord record)` : Adds a record to the hash map.
  - `get(String key)` : Retrieves a record based on the key.
  - `resize()` : Resizes the hash map when the load factor exceeds 0.75.
- [x] **DatabaseProcessing**: he DatabaseProcessing class contains methods for loading data from people.txt into the binary search tree (loadData), searching for a record by name (search), sorting records by name using MyHeap (sort), and finding the most frequent words from the dataset (getMostFrequentWords). These methods have been

tested in the main method of DatabaseProcessing, which also verified that the MyBST, MyHeap, MyHashMap are functioning properly.

- `loadData(String fileName)` : Loads data from `people.txt` into an instance of `MyBST` , enabling efficient data retrieval.
- `search(String firstName, String lastName)` : Searches for records in the BST based on the provided first and last names.
- `sort()` : Sorts records using heap sort by transferring them from the BST to the heap, ensuring ordered data retrieval.
- `getMostFrequentWords(int count, int len)` : Returns the top `count` most frequent words (of at least length `len`) in specific fields of `people.txt` , facilitating data analysis.

Testing

The `DatabaseProcessing` class manages a collection of `PeopleRecord` objects using a Binary Search Tree (BST), a heap for sorting, and a hash map for tracking word frequencies. The testing strategy consists of various tests to ensure that the functionality works as expected.

1.1 Insertion Functionality

- **Test Method**: `testInsertFromDataset()`
- **Purpose**: To verify that the insertion of records into the BST is functioning correctly.
- **Approach**:
  - Insert a predefined set of `PeopleRecord` objects into the BST.
  - Check if the total number of nodes in the BST matches the number of inserted records.
  - Assert that the BST contains the expected records by checking node details.

1.2 Search Functionality

- **Test Method**: `testSearchFromDataset()`
- **Purpose**: To confirm that the search function correctly retrieves records based on family and given names.
- **Approach**:
  - Test searching for existing records (e.g., "John Smith" and "Jane Doe") and verify that the returned record matches the expected data.
  - Include a test for a non-existent record (e.g., "Michael Brown") and assert that the result is empty.
  - Print results for clarity during testing.

1.3. Info Retrieval

- **Test Method**: `testGetInfo()`
- **Purpose**: To validate the output of the `getInfo()` method from the BST.
- **Approach**:
  - Call the `getInfo()` method and print its output.
  - Assert that the output includes expected phrases indicating the total number of nodes and height of the tree.

2.1. Sorting Functionality

- **Test Method**: `sort()`
- **Purpose**: To confirm that the sorting method correctly sorts records by family name.
- **Approach**:
  - Retrieve sorted records and print the first 10 entries for verification.
  - Ensure that the records are in ascending order based on family names.

2.2. Birthday Sorting

- **Test Method**: `sortByBirthday()`
- **Purpose**: To check that records can be sorted by birthday.
- **Approach**:
  - Retrieve and print the first 10 records sorted by birthday.
  - Verify the sorting by comparing the birthday fields.

3.1. Frequent Words Retrieval

- **Test Method**: `getMostFrequentWords(int count, int len)`
- **Purpose**: To validate the extraction of the most frequently occurring words from the records.
- **Approach**:
  - Set a minimum length for words and retrieve the top `count` frequent words.
  - Print the results and ensure that the returned list contains the expected words and their frequencies.

4.1. Short Length Exception

- **Test Method**: Error handling within `getMostFrequentWords()`
- **Purpose**: To ensure that the method properly throws an exception for lengths below the specified threshold.
- **Approach**:
  - Invoke `getMostFrequentWords()` with a length less than 3.
  - Catch and print the exception message to confirm the error handling works correctly.

Task description

- Member 1: Implemented `PeopleRecord`, `MyBST`, and contributed to the `DatabaseProcessing` class.
- Member 2: Developed `MyHeap`, `MyHashmap`, and handled testing.

Overall contribution ratio: 50-50.

Additional features

- [ ] Support other typesof records apart from people.txt, BST/Heap or Hashmap are generic
- [x] Using email/phone/birthday to compare and order
- [x] Graph illustration of the BST/ Heap/Hashmap
- [x] Good testing stratey