Report For HW2

Author: Abdullah Mesut Güler

Number: 32815522378

About the Project: The provided code is an implementation of a Spotify-like program that uses a hash structure to store and manipulate data about people and songs. The program can be run using command-line inputs, where the commands specify various actions to be taken on the data, such as creating a new person, adding a song to a person's liked songs list, deleting a person and their data, and finding matches or recommendations based on the data.

The program uses a hash structure to store the data, with the personTable and songTable being implemented as ArrayList objects. Each element in the ArrayList is a LinkedList, which is used to store the songs that a person likes or dislikes. The getKey() method is used to convert a person's name or a song's name into an integer key, which is used as the index in the ArrayList tables where the LinkedList for that person or song is stored.

Documentation for the Software:

- To make this project, 'Apache Netbeans IDE 15' and Java version 'jdk19' are used.

- public void **startProcess**(String **filepath**) throws IOException

This method is the only public method. It can be used to start the programm by providing the file path of text file in which there are the commands.

- private void **readCommandTxt**(String **filepath**) throws FileNotFoundException, IOException{
- private boolean **executeCommands**(String **line**)

These two methods work together. **readCommandTxt** reads the file line by line. After reading every line it sends that line to **executeCommands** method. In there, with the help of switch-cases the command is executed.

- private boolean **addPerson**(String **personName**)

The method takes a single parameter, personName, which represents the name of the person to be added to the table. The method starts by initializing a boolean variable named cond to false. This variable will be used to indicate whether the person was successfully added to the table or not.

The next step is to generate a unique key for the person using the getKey() method, which converts the person's name into an integer key. This key is then used as the index in the personTable ArrayList where the LinkedList for this person's liked songs will be stored.

Then, the method checks whether there is already a LinkedList for this person's liked songs at this index in the personTable by calling the get() method of the ArrayList with the keyForPerson parameter as the index. If there is already a LinkedList at this index, it means that the person with this name has been added to the table before and it returns the initial value of the cond variable, which is false. Otherwise, if the person is not in the table yet, it creates a new LinkedList object named pSongsList for this person's liked songs, and sets the element of personTable ArrayList at index keyForPerson to this LinkedList using the set() method.

- private boolean **personLikesSong**(String **personName** ,String **songName**)

The method takes two parameters, personName and songName, which represent the name of the person and the name of the song, respectively.

The method checks whether the person already exists in the personTable by calling the get() method of the ArrayList with the keyForPerson parameter as the index. If the result is not null then the person exists in the table and the code will proceed, otherwise it will return the initial value of the cond variable, which is false.Otherwise, if the person is in the table and exists, it will add the songName to the person's liked songs list by calling the push() method on the LinkedList object stored at the keyForPerson index in the personTable.The method then prints out a message indicating that the song has been added to the person's liked songs list and sets the cond variable to true to indicate that the song was successfully added to the list.

The method also performs some operations for the songTable, which is another hash table that stores information about the songs and the people who like them. The songTable also uses an ArrayList to store the data and a similar process to add songs and update the list. If the result is null then the song is not in the table yet and it will proceed to create a new LinkedList object named sPersonsList for this song. The persons that like this song will be stored here.And it will put the reference of this linkedlist into the songTable at keyForSong index. Then it pushes the personName that likes this song into the linkedlist. Otherwise, if the song already exists in the songTable it will proceed to add the personName to the song's persons list by calling the push() method on the LinkedList object stored at the keyForSong index in the songTable.

Finally, it returns the cond variable, indicating the outcome of the method call whether the song was successfully added or not.

- private boolean **personDisliked**(String **personName**, String **songName**)

It is used to remove a song from a person's liked songs list in the personTable hash table. The method takes two parameters, personName and songName, which represent the name of the person and the name of the song, respectively.The method starts by initializing a boolean variable named cond to false. This variable will be used to indicate whether the song was successfully removed from the person's liked songs list or not.

Then, the method checks whether the person exists in the personTable by calling the get() method of the ArrayList with the keyForPerson parameter as the index. If the result is not null then the person exists in the table and the code will proceed, otherwise it will return the initial value of the cond variable, which is false. Otherwise, if the person is in the table and exists, it will remove the songName from the person's liked songs list by calling the popRequired() method on the LinkedList object stored at the keyForPerson index in the personTable. This method will return true if the song is removed from the liked songs list, otherwise it will return false, if the song is not in the liked songs list .

The method also performs some operations for the songTable, which is another hash table that stores information about the songs and the people who like them. The songTable also uses an ArrayList to store the data and a similar process to remove songs and update the list. It checks whether the song already exists in the songTable by calling the get() method of the ArrayList with the keyForSong parameter as the index. If it exists, it will proceed to remove the personName from the song's persons list by calling the popRequired() method on the LinkedList object.

- private boolean **deletePerson**(String **personName**)

It is used to delete a person's record from the personTable hash table. The method takes one parameter personName, which represents the name of the person that is to be deleted.The method starts by initializing a boolean variable named cond to false. This variable will be used to indicate whether the person's record was successfully removed from the personTable or not. The method checks whether the person exists in the personTable by calling the get() method of the ArrayList with the keyForPerson parameter as the index. If the result is not null then the person exists in the table and the code will proceed to delete the person, otherwise it will print "personName is not in the list" and will return the initial value of the cond variable, which is false. If the person exists in the table, the method will remove the person's record by calling the set() method of the ArrayList with the keyForPerson parameter as the index and passing null as the value. Then, the cond variable is set to true as the deletion is successful.

The method also performs some operations for the songTable, which is another hash table that stores information about the songs and the people who like them. It will delete the person's name from the liked song list in the songTable by checking each song that person liked. It will pop each song from the list and check whether it exists in the songTable. If it exists, it will remove the person's name from the song's persons list by calling the popRequired() method on the LinkedList object stored at the keyForSong index in the songTable.

Finally, the method returns the cond variable, indicating whether the person was successfully deleted or not.

- private boolean **personLikes**(String **personName**)

Simply prints the liked songs by the personName.

- private boolean **offerSong**(String **personName**)

It is used to recommend 5 different songs that a person may like but have not yet listened to. The method takes one parameter personName, which represents the name of the person for whom the recommendations are to be generated. The method starts by initializing a boolean variable named cond to false and an integer variable named cnt to 0. The variable cond will be used to indicate whether the recommendations have been generated successfully or not, while the variable cnt is used to keep track of the number of songs recommended.

Then, the method checks whether the person exists in the personTable by calling the get() method of the ArrayList with the keyForPerson parameter as the index. If the result is null then the person does not exist in the table and the code will print "There is not such a person." and will return the initial value of the cond variable, which is false. If the person exists in the table, the method will proceed to recommend songs. It will iterate through the liked songs linked list stored in the personTable at the keyForPerson index. For each song in the list it will check the persons list that stored in the songTable at the keyForSong index and iterate through the list of persons. It will then check the liked songs of each person in the persons list and iterate through the liked songs list. If the recommended song already in the liked songs list of the person, the method will continue to next song. If the recommended song not in the liked songs list of the person it will print "Recommending "+ recommended song and increment the cnt by 1.

Finally, the method will check the counter variable, if the counter variable is equal to 5, which means 5 songs has been recommended successfully, then the function will return true, indicating that the recommendations were generated successfully. If the counter is less than 5, it means that not enough songs were found to recommend, and the method will return false.

- private boolean **matchMaking**(String **personName**)

The method matchMaking(String personName) is responsible for finding potential friends for a person with the given name. The method first uses the getKey() method to convert the person's name into an integer key, which is used to locate the person in the personTable. If the person does not exist in the table, the method prints "There is not such a person" and returns false.If the person is found in the table, the method accesses the person's liked songs list. If the list is empty, the method prints "does not have any liked songs. Cannot find possible friends" and returns false. Otherwise, the method iterates through the person's liked songs and for each song, it looks at the list of people who liked that song in the songTable, and check if the person is in that list. If yes, the method prints "Possible friend " + dummySongs.data and increase the cnt value. If the cnt reach to the limit of 3 match it returns true else return false. The method performs this operation for each of the person's liked songs and stops if it finds 3 potential friends. However, if the person does not have any liked songs, the method cannot find potential friends and return false.