***Ghaleb Ali***

***E02049735***

***COSC 311 Fall 2020***

***Program #1***

***10/8/19***

***Assignment classes***

***1.DataBase.java***

***2.Index(ordered array)***

***3.IndexRecord***

***4.StudentRecord***

***5.COSC311Drivr***

***COSC311Driver.java***

**package ProgrammingAssign1;**

**/\* IMPORTANT. All of your programs should have the proper heading. The following should be how all your programs begin:**

**Name: <your name here>**

**Assignment Number: <put the assignment number here>**

**COSC 311 - Winter 2015**

**\*/**

**/\*\***

**\* This will be the main driver program for many of your programs. Specifically,**

**\* you will need to define a data structure and related algorithms to use with this program.**

**\* We will be using the data file I have provied for you: a file of 68 records. Each record is composed**

**\* of three fields:**

**\* String lastName**

**\* String firstName**

**\* String ID**

**\* ID may be implemented as an integer, but it is easier to implement as a string. Both lastName and firstName**

**\* may not be unique, but the ID \*\*is\*\* unique.**

**\***

**\* @author Bill Sverdlik**

**\* @version Version 1.0**

**\*/**

**/\* Anything special you want the teacher to know? Are you asking for bonus points? Explain here.\*/**

**import java.util.\*;**

**public class COSC311Driver**

**{**

**public static void main(String[] args)**

**{**

**/\*The following declaration declares a data structure that will change from one assignment to the next. For example, you will need to implement**

**\* the following as a doubly linked list, as well as a tree.**

**\*/**

**DataBase d=new DataBase();**

**int response;**

**Scanner keyboard=new Scanner(System.in);**

**/\* Read the data into the database from the external disk file here**

**\* IMPORTANT: duplicate ID numbers should not be added. Disregard**

**\* the entire record for duplicate IDs**

**\*/**

**do**

**{**

**System.out.println(" 1 Add a new student");**

**System.out.println(" 2 Delete a student");**

**System.out.println(" 3 Find a student by ID");**

**System.out.println(" 4 List students by ID increasing");**

**System.out.println(" 5 List students by first name increasing");**

**System.out.println(" 6 List students by last name increasing");**

**System.out.println(" 7 List students by ID decreasing");**

**System.out.println(" 8 List students by first name decreasing");**

**System.out.println(" 9 List students by last name decreasing");**

**System.out.println(" ");**

**System.out.println(" 0 End");**

**response=keyboard.nextInt();**

**switch (response)**

**{**

**case 1: d.addIt(); //Note: if the user enters an ID already in use, issue a warning and return to the menu**

**break;**

**case 2: d.deleteIt(); //Note: output either "Deleted" or "ID not Found" and return to menu**

**break;**

**case 3: d.findIt(); //Note: output the entire record or the message "ID not Found" and return to menu**

**break;**

**case 4: d.ListByIDAscending();**

**break;**

**case 5: d.ListByFirstAscending();**

**break;**

**case 6: d.ListByLastAscending();**

**break;**

**case 7: d.ListByIDDescending();**

**break;**

**case 8: d.ListByFirstDescending();**

**break;**

**case 9: d.ListByLastDescending();**

**break;**

**default:**

**}**

**} while (response!=0);**

**}**

**}**

***DataBase.java***

package ProgrammingAssign1;

import java.io.\*;

import java.util.\*;

//Main database class to insert info into IndexArray and StudentRecord

public class DataBase

{

//Needed Object array/objects/counter.

Scanner input = new Scanner(System.***in***);

private StudentRecord[] data;

private Index fname,lname, id;

private int next;

/\*

\* Default constructor to read data from file.

\* declare default size for students in all data.

\*/

public DataBase()

{

next = 0;

data = new StudentRecord[100];

fname = new Index(100);

lname = new Index(100);

id = new Index(100);

File f = new File("C:\\Users\\13133\\Desktop\\DataSet.txt");

String fn;

String ln;

String iDn;

//Makes sure same ID number isn't used twice.

try

{

Scanner scan = new Scanner(f);

while(scan.hasNextLine())

{

String name = scan.nextLine();

String lowercase = name.toLowerCase();

String[] n = lowercase.split(" ");

fn = n[0];

ln = n[1];

iDn = n[2];

if(id.getIndexlocation(new IndexRecord(iDn,next) )!=-1 )

{

data[next]= new StudentRecord(fn, ln, iDn);

fname.insert(new IndexRecord(fn,next));

lname.insert(new IndexRecord(ln,next));

id.insert(new IndexRecord(iDn,next));

next++;

}

}

}

catch (FileNotFoundException e)

{

e.printStackTrace();

}

}

/\*Constructor in case someone wants use Database in a different class

\*and declare the size of Database and indexArray's.

\*/

public DataBase(int nStudent)

{

data = new StudentRecord[nStudent];

next = 0;

fname = new Index(nStudent);

lname = new Index(nStudent);

id = new Index(nStudent);

}

/\*

\* AddIt method makes sure the Id number is never entered twice

\* in either IndexArray or Database. AddIt method inserts first/last

\*name and Id number into the main Database base and also into

\*the indexArray with the location of where the first/last name and

\*id number are located in the main database using integer where

\* from IndexRecord.

\*/

public void addIt()

{

System.***out***.println("\nEnter The following Information"

+ " In Oder Asked For The New Student!");

System.***out***.println("First Name: "

+ "Last Name : ID Number:");

String fnamE = input.next();

String lnamE = input.next();

String iD = input.next();

int location = isiDinuse(iD);

if(data[location].getiD().compareTo(iD) != 0 )

{

data[next] = new StudentRecord(fnamE, lnamE, iD);

fname.insert(new IndexRecord(fnamE,next));

lname.insert(new IndexRecord(lnamE,next));

id.insert(new IndexRecord(iD,next));

next++;

System.***out***.println();

}

else if(data[location].getiD().compareTo(iD) == 0)

{

System.***out***.println("\nID Is Already In Use!");

System.***out***.println("Press 3 To Look Up The Record"

+ " By Using The ID Number.\n");

}

}

//Find method to find location in IndexRecord than print from DataBase.

public void findIt()

{

System.***out***.print("Please Enter an ID To Look Up: ");

String iDl = input.next();

int location;

location = id.find(iDl);

if(data[location].getiD().compareTo(iDl) == 0)

{

System.***out***.println("\n"+data[location]+"\n");

}

else if(data[location].getiD().compareTo(iDl) != 0)

{

System.***out***.println("\nID not Found\n");

}

}

//Delete method to delete from IndexRecord.

public void deleteIt()

{

System.***out***.print("\nPleasae Enter an ID to Delete: ");

String iDD = input.next();

int index = id.find(iDD);

if(data[index].getiD().compareTo(iDD) == 0)

{

id.delete(new IndexRecord(iDD, index));

String firstname = data[index].getFnamE();

fname.delete(new IndexRecord(firstname, index));

String lastname = data[index].getLnamE();

lname.delete(new IndexRecord(lastname, index));

System.***out***.println("\nRecord of the student with"

+ " ID number "+iDD+" has been Deleted.\n");

}

else if(data[index].getiD().compareTo(iDD) != 0)

{

System.***out***.println("\nID not Found: Record cannot be Deleted.\n");

}

}

//Use find id to make sure no duplicate ID number gets inserted again

public int isiDinuse(String iD)

{

return (id.find(iD));

}

//Print out DataBase (ID number for students in increasing order).

public void ListByIDAscending()

{

ascending(id);

}

//Print out DataBase (first name for students a-z).

public void ListByFirstAscending()

{

ascending(fname);

}

//Print out DataBase (last name for students a-z).

public void ListByLastAscending()

{

ascending(lname);

}

//Print out DataBase (ID number for students in decreasing order).

public void ListByIDDescending()

{

descending(id);

}

//Print out DataBase (first name for students z-a).

public void ListByFirstDescending()

{

descending(fname);

}

//Print out DataBase (last name for students z-a).

public void ListByLastDescending()

{

descending(lname);

}

/\*

\* Print Database with whatever option user wanted

\*(id,first name/last name) using the iterator..

\*/

public void ascending(Index option)

{

option.Beginiterator();

while(option.hasNextIndex())

{

System.***out***.println(data[option.getNextIndex()]);

}

}

/\*

\* Print Database with whatever option user wanted

\*(id,first name/last name) using the iterator.

\*/

public void descending(Index option)

{

option.Enditerator();

while(option.hasIndexBefore())

{

System.***out***.println(data[option.getIndexBefore()]);

}

}

}

***Index.java***

package ProgrammingAssign1;

//Index class is an ordered array of IndexRecord class.

public class Index

{

//Ordered Array needed global variables.

private IndexRecord[] infoRecord;

private int next;

private int maxSize;

public int iTerator;

//Default constructor.

public Index()

{

this.infoRecord = new IndexRecord[100];

this.next = 0;

this.maxSize = 100;

this.iTerator = 0;

}

//Constructor for the size of the ordered array.

public Index( int sz)

{

this.infoRecord = new IndexRecord[sz];

this.next = 0;

this.maxSize = sz;

this.iTerator = 0;

}

/\*

\* Insertion loop to enter id number and first/ last name in correct

\* order using the compare method with the index of the where

\* they are located in the main database.

\*/

public void insert(IndexRecord unkeptinfo)

{

int i;

for(i = next-1 ; i>=0 ; i-- )

{

if( infoRecord[i].compareTo(unkeptinfo)< 0) break;

infoRecord[i+1] = infoRecord[i];

}

infoRecord[i+1] = unkeptinfo;

next++;

}

/\*

\* A method with a passed ID number from the main database class

\* to be search for using binary search and return the index of where

\* the ID number is in the main data base class. We use this index to

\* print the first/last name of the student with the ID number.

\*/

public int find(String iD)

{

int low=0, high=next-1, mid=0;

while(low <= high)

{

mid = (low+high)/2;

if(infoRecord[mid].getKey().compareTo(iD) == 0) break;

if(infoRecord[mid].getKey().compareTo(iD) < 0)

low = mid + 1;

else

high = mid -1;

}

return (infoRecord[mid].getWhere() );

}

/\*

\* A method with a passed string and integer to be deleted from the

\*index array, so when you print out the main database class using the

\*index the deleted index wont print that student information while also

\*still being in the main data base if needed to be accessed later.

\*/

public int delete(IndexRecord studentDelete)

{

int k;

int where= getIndexlocation(studentDelete);

if(where != -1)

{

for(k=where+1; k<next; k++)

infoRecord[k-1] = infoRecord[k];

next--;

}

return(where==-1?-1:1);

}

/\*

\* A linear search method that helps find the index of where the

\* passed Index record(key,where) is so the binary search in the delete

\* method could find the correct location to delete.

\*/

public int getIndexlocation(IndexRecord index)

{

int i;

for( i=0; i<next-1 ; i++)

if(infoRecord[i].compareTo(index) == 0 ) break;

return (i);

}

//A print method for Index array(ID,first/last name and index).

public void printit()

{

for(int i=0; i<next-1 ; i++)

System.***out***.println(infoRecord[i]+" ");

}

//check if IndexRecord is empty.

public boolean isEmpty()

{

return (next==0);

}

//check if IndexREcord is full.

public boolean isFull()

{

return (next==maxSize);

}

/\*

\* Iterator method that helps print out the main database front wards and

\* backwards which helps print the ID number increasing or decreasing or

\* first/last name from a-z order or z-a order.

\*/

//Reset iterator to the start of the array.(print from front-end)

public void Beginiterator()

{

iTerator = 0;

}

//Iterator at the end of array.(print from end-front)

public void Enditerator()

{

iTerator = next-1;

}

//Get the indexes so main database can print(front-end).

public int getNextIndex()

{

return (infoRecord[iTerator++].getWhere());

}

//Check if there are any values left before ending Iterator

public boolean hasNextIndex()

{

return (iTerator < next);

}

//check if indexes before exist

public boolean hasIndexBefore()

{

return (iTerator >= 0);

}

//Get the indexes so main database can print(end-front).

public int getIndexBefore()

{

return (infoRecord[iTerator--].getWhere());

}

}

***IndexRecord.java***

package ProgrammingAssign1;

/\*

\* IndexRecord class used to indicate where first name

\*last name and Id number are in the main data base

\*with indexes

\*/

public class IndexRecord

{

//needed for keep the index array to use methods(delete,find,insert)

private String key;

private int where;

//Default constructor

public IndexRecord()

{

this.key = "ali";

this.where = 8;

}

//Constructor for more student information with indexes.

public IndexRecord(String k, int w)

{

this.key = k;

this.where = w;

}

//Getter methods : if I need to access or compare information.

public String getKey()

{

return (key);

}

public int getWhere()

{

return (where);

}

//If needed to see output for student information and index.

public String toString()

{

return (key+" "+where);

}

/\*

\* Compare method to compare names and id in main index array

\* class such as (Binary search, insertion loop, linear search) .

\*/

public int compareTo(IndexRecord recorD)

{

return (key.compareTo(recorD.key));

}

}

***StudentRecord.java***

package ProgrammingAssign1;

//Student Record class that contains

//first name , last name, and id number strings.

public class StudentRecord

{

//Needed info for students

private String fnamE;

private String lnamE;

private String iD;

//Default constructor for student record

public StudentRecord()

{

this.fnamE = "Ali";

this.lnamE = "Ghaleb";

this.iD = "1";

}

//Constructor for more students to be entered

public StudentRecord(String f, String l, String i)

{

this.fnamE = f;

this.lnamE = l;

this.iD = i;

}

//Getter methods for first and last name and id.

public String getFnamE()

{

return (fnamE);

}

public String getLnamE()

{

return (lnamE);

}

public String getiD()

{

return (iD);

}

//ToString method to print data from the main database.

public String toString()

{

return (iD+" "+fnamE+" "+lnamE);

}

}

}