```
import java.util.Scanner;
import java.io.FileInputStream;
import java.io.FileNotFoundException;

/**
 * COSC311 - Program 1
 *
 * This file defines the <code>DataStructure</code> type with an array of
 *  <code>DatabaseRecords</code> and three <code>Index</code> objects. An object
 *  initially reads in a list of records from an external file to populate the
 *  database. The structures can then be searched, added to, deleted from,
 *  displayed, and can print an individual record.
 *
 * Student records are referenced to by their position in the database, which
 *  is stored in each <code>Index</code> as the <code>recordNumber</code>.
 *
 * @author Mordechai Sadowsky
 * @version 02-feb-2014
 *
 */
public class DataStructure {

    private DatabaseRecord[] database;
    private Index firstNames, lastNames, ids;
    private int databasePointer;
    private final int SIZE = 100;
    private DBStack deletedRecords = new DBStack(SIZE);
    private final String PATH =
            "/Users/Mordechai/git/COSC311/Program1/src/data.txt";

    public DataStructure() {
        Scanner inputStream = null;
        try {
            inputStream = new Scanner(new FileInputStream(PATH));
        }
        catch (FileNotFoundException e) {
            System.out.println(e.getMessage());
            System.out.println("Don't forget to update file path name!");
            System.exit(1);
        }

        //initialize data members
        database = new DatabaseRecord[SIZE];
        firstNames = new Index(SIZE);
        lastNames =  new Index(SIZE);
        ids =  new Index(SIZE);
        databasePointer = 0;

        //Read in database from external file and
        //  add records to main database and the indexes
        while (inputStream.hasNextLine()) {
            String first = inputStream.next();
            String last = inputStream.next();
            String id = inputStream.next();
            database[databasePointer] = new DatabaseRecord(first, last, id);
            firstNames.insert(first, databasePointer);
```

```
            lastNames.insert(last, databasePointer);
            ids.insert(id, databasePointer);
            databasePointer++;
        }
        if (inputStream.hasNextLine()) {
            System.out.println("File is too big! Increase SIZE.");
            System.exit(1);
        }
    }

    /**
     * Searches through the <code>Index</code> of IDs because the database may
     *  contain deleted records.
     *
     * @param id number of a student to search for
     * @return The reference <code>recordNumber</code> of the student, i.e. the
     *  index of the student record's position in the <code>database</code>
     */
    public int search(String id) {
        return ids.find(id);
    }

    /**
     * Adds a record to the database and each <code>Index</code>
     *  Records are inserted in lexicographical order into the indices,
     *  but are entered into the <code>database</code> at the site of a
     *  previously deleted record or the end of the <code>database</code>.
     * @param first First name of the new student.
     * @param last Last name of the new student.
     * @param id ID number of the new student.
     * @return true for successful insertion, false for failure
     */
    public boolean insert(String first, String last, String id) {
        int bookmark;

        //check the stack to see if any lines in the middle of the database
        //  are free for insertion
        if (!deletedRecords.isEmpty()) {
            bookmark = databasePointer; //keep track of database end
            databasePointer = deletedRecords.pop(); //point to "open" space
        }
        else
            bookmark = databasePointer+1; //if no open spaces, (database end)++

        database[databasePointer] = new DatabaseRecord(first, last, id);

        //insert record pieces into their respective indices
        //  if any do not insert successfully, DataStructure.insert() fails
        if (!(firstNames.insert(first, databasePointer)) ||
            !(lastNames.insert(last, databasePointer)) ||
            !(ids.insert(id, databasePointer)))
            return false;
        databasePointer = bookmark; //update pointer back to end or incremented
        return true;
    }
```

```
/**
 * Removes a record from each index, and adds its location in the main
 *  <code>database</code> to the stack of <code>deletedRecords</code>
 * @param id ID number of student to delete
 */
public void delete(String id) {
    int recordToDelete = ids.find(id); //finds reference recordNumber

    firstNames.delete(recordToDelete);
    lastNames.delete(recordToDelete);
    ids.delete(recordToDelete);

    deletedRecords.push(recordToDelete);
}

/**
 * Displays entire database in one of 6 different orders by reading through
 *  an Index in order to pull the reference numbers and print the associated
 *  records one by one.
 *
 * @param a determines which <code>Index</code> to sort by:
 *  1-ID number; 2-first name; 3-last name
 * @param b determines in which lexicographical order to display:
 *  1-ascending order; 2-descending order
 */
public void listIt(int a, int b) {
    if (b == 1) {
        if (a == 1)
            for (int i = 0; i < ids.length(); i++)
                print(ids.getRecordNumber(i));
        else if (a == 2)
            for (int i = 0; i < firstNames.length(); i++)
                print(firstNames.getRecordNumber(i));
        else if (a == 3)
            for (int i = 0; i < lastNames.length(); i++)
                print(lastNames.getRecordNumber(i));
        else
            return;
    }
    else if (b == 2) {
        if (a == 1)
            for (int i = ids.length()-1; i >= 0; i--)
                print(ids.getRecordNumber(i));
        else if (a == 2)
            for (int i = firstNames.length()-1; i >= 0; i--)
                print(firstNames.getRecordNumber(i));
        else if (a == 3)
            for (int i = lastNames.length()-1; i >= 0; i--)
                print(lastNames.getRecordNumber(i));
        else
            return;
    }
}

/**
 * Displays a single <code>DatabaseRecord</code>
```

```
         * @param recordNumber
         */
        public void print(int recordNumber) {
            System.out.println(database[recordNumber]);
        }
    }
```