```
/**
 * COSC 311 - Program 1
 *
 * This file defines the Index data type as an array of IndexRecords.
 *  Index instances can be added to, searched, and deleted from.
 *
 * @author Mordechai Sadowsky
 * @version 02-feb-2014
 *
 */
public class Index {

    private IndexRecord[] records;
    private int size;
    private int numberOfElements;

    public Index(int sz) {
        size = sz;
        records = new IndexRecord[size];
        numberOfElements = 0;
    }

    public boolean isEmpty() {
        return (numberOfElements == 0);
    }

    public boolean isFull() {
        return (numberOfElements == size);
    }

    public int length() {
        return numberOfElements;
    }

    public int getRecordNumber(int x) {
        return records[x].getRecordNumber();
    }

    /**
     * Adds a new student to the <code>Index</code>
     *
     * @param key is the new value (e.g. first name)
     * @param num is the new <code>recordNumber</code> reference to the full
     *  student record in the <code>database</code>
     * @return (record successfully inserted == true)
     */
    public boolean insert(String key, int num) {
        IndexRecord newRecord = new IndexRecord(key, num);
        if (isFull())
            return false;
        if (isEmpty()) {
            records[0] = newRecord;
            numberOfElements++;
            return true;
        }
        int i;
        for(i = numberOfElements-1; i >= 0; i--) {
            if (newRecord.compareTo(records[i]) < 0)
                records[i+1] = records[i];
            else
                break;
        }
        records[i+1] = newRecord;
        numberOfElements++;
        return true;
    }

    /**
     * Removes a student record from the <code>Index</code> by shifting up any
     *  lower records in the array
     * @param recNum is the reference <code>recordNumber</code> of the student
     *  to be deleted
     */
    public void delete(int recNum) {
        int whereInIndex;
        whereInIndex = find(recNum);
        if (whereInIndex != -1) {
            for (int i = whereInIndex+1; i < numberOfElements; i++)
                records[i-1] = records[i];
            numberOfElements--;
        }
    }

    /**
     * Binary search for a record in array of <code>IndexRecord</code>s
     *  <code>records</code>. Indices are already in order, sorted by keys.
     *
     * @param key is the value (e.g. first name) that is being looked for in an
     *  <code>IndexRecord</code> within <code>Index</code>
     * @return <code>recordNumber</code> of goal in <code>records</code>,
     *  -1 if not found
     */
    public int find(String key) {
        IndexRecord goal = new IndexRecord(key, 0);
        int low = 0, middle = 0, high = numberOfElements-1;
        while (low <= high) {
            middle = (high+low)/2;
            if (goal.compareTo(records[middle]) > 0)
                low = middle+1;
            else if (goal.compareTo(records[middle]) < 0)
                high = middle-1;
            else
                return records[middle].getRecordNumber();
        }
        return -1;
    }

    /**
     * Linear search for a record in array of <code>IndexRecord</code>s because
     *  indices are sorted by keys and not by <code>recordNumber</code>.
     *
     * @param num is the <code>recordNumber</code> that is being looked for in
```

```
         *  an <code>IndexRecord</code> within <code>Index</code>
         * @return array index of the found <code>IndexRecord</code> within the
         *  <code>records</code> array, -1 if not found
         */
        public int find(int num) {
            for (int i = 0; i < numberOfElements; i++) {
                if (records[i].getRecordNumber() == num)
                    return i;
            }
            return -1;
        }
    }
```