```
import java.io.File;
import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.IOException;
import java.nio.channels.FileChannel;
import java.util.ArrayList;
import java.util.Arrays;
import java.util.Collections;
import java.util.Random;
import java.util.Scanner;
import java.util.TreeSet;
import java.nio.ByteBuffer;


public class FastReadingSearchDemo {
        private String[] myWords;
    private ArrayList<String> myWordsAsList;
    private static Random ourRandom = new Random();

    /**
     * Read words and store them for searching via
     * other model methods.
     */
    public void initialize(String fileName) {

        try {
                File f = new File(fileName);
            FileChannel channel = new FileInputStream(f).getChannel();
            ByteBuffer bb = channel.map(FileChannel.MapMode.READ_ONLY,
                                        0,
                                        (int) channel.size());
            bb.position(0);
            int count = 0;
            char[] carray = new char[1000];
            boolean inWord = false;

            double start = System.currentTimeMillis();
            TreeSet<String> set = new TreeSet<String>();
            int total = 0;
            while (bb.hasRemaining()){

               char ch = Character.toLowerCase((char) bb.get());
               if (Character.isWhitespace(ch)){
                        if (inWord){
                                inWord = false;
                                set.add(new String(carray,0,count));
                                total++;
                                count = 0;
                        }
               }
               else {
                        inWord = true;
                        carray[count++] = ch;
               }
            }
            if (inWord){
                total++;
                set.add(new String(carray,0,count));
            }
            myWordsAsList = new ArrayList<String>(set);

            double endReading = System.currentTimeMillis();
            double readingTime = (endReading - start)/1000.0;
            Collections.sort(myWordsAsList);
            double sortingTime = (System.currentTimeMillis() -  endReading)/10
00;
            myWords = (String[]) myWordsAsList.toArray(new String[0]);
            String message = "read "+total+" words "+set.size()+" unique";
            message += "\ntime to read "+readingTime+" time to sort "+sortingTime;
            System.out.println(message);
        } catch (IOException e) {
            e.printStackTrace();
        }
```



```
    }

    private void searchBinary(int amount){

        double start = System.currentTimeMillis();
        for(int k=0; k < amount; k++){
            int index = ourRandom.nextInt(myWords.length);
            int search = Arrays.binarySearch(myWords, myWords[index]);
            if (search != index){
                System.err.println("searching failed");
                return;
            }
        }
        double end = System.currentTimeMillis();
        double time = (end-start)/1000;
        System.out.println("time for "+amount+" binary searches "+time);
    }
    private void searchSequential(int amount){

        double start = System.currentTimeMillis();
        for(int k=0; k < amount; k++){
            int index = ourRandom.nextInt(myWords.length);
            int search = myWordsAsList.indexOf(myWords[index]);
            if (search != index){
                System.out.println("searching failed");
                return;
            }
        }
        double end = System.currentTimeMillis();
        double time = (end-start)/1000;
        System.out.println("time for "+amount+" sequential searches "+time);
    }

    public static void main(String[] args) {
        String fileName = "poe.txt";
        int count = 100;
        if (args.length != 2){
            System.err.println("usage: java SearchModel filename count");
            return;
        }
        fileName = args[0];
        count = Integer.parseInt(args[1]);
        FastReadingSearchDemo sm = new FastReadingSearchDemo();
        sm.initialize(fileName);
        sm.searchBinary(count);
        sm.searchSequential(count);

    }
}
```