```
import java.io.File;
import java.util.*;

import javax.swing.JFileChooser;

public class BenchMarker implements IStringProcessor {

        private static JFileChooser ourChooser = new JFileChooser(".");
    static {
                ourChooser.setFileSelectionMode(JFileChooser.DIRECTORIES_ONLY);
    }

    private TreeSet mySet = new TreeSet();

    public File[] getFiles(){
                int retval = ourChooser.showOpenDialog(null);
            if (retval != JFileChooser.APPROVE_OPTION){
                return null;
            }
            File dir = ourChooser.getSelectedFile();
            return dir.listFiles();
    }

    public void processFiles(File[] list, IWordFactory wordFactory){

                double start = System.currentTimeMillis();

            for(int k=0; k < list.length; k++){
                        if (list[k].getName().endsWith(".txt")){
                                System.out.println("reading "+list[k].getName());
                                wordFactory.open(list[k]);
                                wordFactory.readAndProcess(this);
                        }
            }
            double end = System.currentTimeMillis();
            double time = (end-start)/1000;
            System.out.println("time "+time);
    }

        public static void main(String[] args){
                BenchMarker bench = new BenchMarker();
                File[] list = bench.getFiles();
                IWordFactory factory = new ScannerWordFactory();
                bench.processFiles(list,factory);

        System.exit(0);
        }

        public void process(String s) {
                mySet.add(s);
        }

        public void process(INanoWord s) {
                mySet.add(s);
        }
}
```



```
import java.io.PrintStream;

public class BufferBackedNanoWord implements INanoWord {

    char[] myBuffer;
    int myFirst;
    int myLength;
    int myHash;

    public BufferBackedNanoWord(char[] buf, int first, int len){
        myBuffer = buf;
        myFirst = first;
        myLength = len;
        myHash = 0;
    }
    public void print(PrintStream output) {
        int loc = myFirst;
        for(int k=0; k < myLength; k++){
            output.print(myBuffer[loc++]);
        }
    }

    public int compareTo(Object arg) {
       BufferBackedNanoWord other = (BufferBackedNanoWord) arg;
       int oFirst = other.myFirst;
       int mFirst = myFirst;
       int oLength = other.myLength;
       char[] oBuffer = other.myBuffer;
       int length = Math.min(myLength,oLength);
       for(int k=0; k < length; k++){
           char c1 = myBuffer[mFirst++];
           char c2 = oBuffer[oFirst++];
           if (c1 != c2){
               return c1 - c2;
           }
       }
       return myLength - oLength;
    }

    public boolean equals(Object arg) {
        if (arg == null) return false;
        if (! (arg instanceof BufferBackedNanoWord)) return false;

        BufferBackedNanoWord other = (BufferBackedNanoWord) arg;
        if (myLength != other.myLength) return false;
        for(int k=0; k < myLength; k++){
            if (myBuffer[myFirst+k] != other.myBuffer[other.myFirst+k]){
                return false;
            }
        }
        return true;
     }

    public int hashCode(){
        if (myHash != 0) return myHash;

        int off = myFirst;
        for (int k = 0; k < myLength; k++) {
            myHash = 31*myHash + myBuffer[off++];
        }
        return myHash;
    }
}
```

Sep 14, 05 10:11 **INanoWord.java** Page 1/1

```
import java.io.PrintStream;

public interface INanoWord extends Comparable {

        public abstract void print(PrintStream output);

        // methods below are in Object, remind users to implement!

        public String toString();
        public int hashCode();
        public boolean equals(Object o);
}
```

Sep 14, 05 10:11 **IStringProcessor.java** Page 1/1

```
public interface IStringProcessor {

        public void process(String s);
    public void process(INanoWord s);
}
```

Sep 14, 05 10:11 **IWordFactory.java** Page 1/1

```
import java.io.File;

public interface IWordFactory {

        public void open(File f);
        public abstract void readAndProcess(IStringProcessor isp);

}
```

Sep 14, 05 10:11 **NanoWordFactory.java** Page 1/1

```
import java.io.File;
import java.io.FileInputStream;
import java.io.IOException;
import java.nio.ByteBuffer;
import java.nio.channels.FileChannel;

public class NanoWordFactory implements IWordFactory{
    private File myFile;
    private ByteBuffer myBuffer;
    private char[] myArray;
    public NanoWordFactory(){
       // nothing to do
    }

    public void open(File f){
            myFile = f;
        try {
            FileChannel channel = new FileInputStream(myFile).getChannel();
            myBuffer = channel.map(FileChannel.MapMode.READ_ONLY,
                                   0,
                                   (int) channel.size());
            myArray = new char[myBuffer.limit()];
            myBuffer.position(0);
            int count = 0;
            while (myBuffer.hasRemaining()){
                myArray[count++] = Character.toLowerCase((char) myBuffer.get());
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    public boolean wordPart(char ch){
        return Character.isLetter(ch) || ch == '\'';
    }
    public void readAndProcess(IStringProcessor isp){
        boolean inWord = false;
        int length = 0;
        int start = 0;

        for(int k=0; k < myArray.length; k++){
            char ch = myArray[k];
            if (inWord){
                if (! wordPart(ch)){
                    // word ended, process it
                    BufferBackedNanoWord nw =
                        new BufferBackedNanoWord(myArray,start,length);
                    isp.process(nw);
                    length = 0;
                    inWord = false;
                }
                else {
                    length++;
                }
            }
            else { // not in word, does a word start?
                if (Character.isLetter(ch)){
                    inWord = true;
                    start = k;
                    length = 1;
                }
            }
        }
        if (length > 0){
            BufferBackedNanoWord nw =
                new BufferBackedNanoWord(myArray,start,length);
            isp.process(nw);
        }
    }
}
```

Sep 14, 05 10:11 **ScannerWordFactory.java** Page 1/1

```
import java.io.File;
import java.io.FileNotFoundException;
import java.util.*;

public class ScannerWordFactory implements IWordFactory {

        private File myFile;
        private Scanner myScanner;

        public ScannerWordFactory(){
            // nothing to do
        }

        public void open(File f){
                myFile = f;
                try {
                        myScanner = new Scanner(myFile);
                } catch (FileNotFoundException e) {
                        e.printStackTrace();
                }
        }

        public void readAndProcess(IStringProcessor isp) {
                while (myScanner.hasNext()){
                        String s = myScanner.next();
                        s = SimpleWordFilter.filter(s);
                        if (s != null){
                                isp.process(s);
                        }
                }
        }

}
```

Sep 14, 05 10:11 **SimpleWordFilter.java** Page 1/1

```
public class SimpleWordFilter {

        public static String filter(String s){
        int start = 0;
        while (start < s.length() && !Character.isLetter(s.charAt(start))){
            start++;
        }
        int end = s.length()-1;
        while (start < end && ! Character.isLetter(s.charAt(end))){
            end--;
        }
        if (start <= end){
            char[] buf = s.toCharArray();
            for(int k=start; k <= end; k++){
                buf[k] = Character.toLowerCase(buf[k]);
                if (buf[k] == ':'){
                    return null;
                }
            }
            String sw = new String(buf,start,end-start+1);
            return sw;
        }
        return null;
    }
}
```

```
import java.io.PrintStream;

public class StringNanoWord implements INanoWord {
    private String myString;
    public StringNanoWord(String s){
        myString = s;
    }

    public void print(PrintStream output) {
        output.print(myString);
    }

    public int compareTo(Object arg) {
       StringNanoWord other = (StringNanoWord) arg;
       return myString.compareTo(other.myString);
    }

    public boolean equals(Object arg){
        if (arg == null) return false;
        if (! (arg instanceof StringNanoWord)) return false;
        StringNanoWord other = (StringNanoWord) arg;
        return myString.equals(other.myString);
    }

    public int hashCode(){
        return myString.hashCode();
    }

}
```



```
import java.io.PrintStream;

public class StringWord implements INanoWord {
        private String myString;
        public StringWord(String s){
                myString = s;
        }

        public StringWord(char[] buf, int start, int length){
                myString = new String(buf,start,length);
        }

        public void print(PrintStream output) {
                output.print(myString);
        }

        public int compareTo(Object o) {
                StringWord other = (StringWord) o;
                return myString.compareTo(other.myString);
        }

        public int hashCode(){
                return myString.hashCode();
        }

        public String toString(){
                return myString;
        }

        public boolean equals(Object o){
                if (o instanceof String){
                        return myString.equals(o);
                }
                StringWord other = (StringWord) o;
                return myString.equals(other.myString);
        }
}
```

```
import java.io.*;
import java.nio.*;
import java.nio.channels.*;
public class WordFactory implements IWordFactory {
    private File myFile;
    private ByteBuffer myBuffer;

    public WordFactory(){
        // nothing to do
    }

    public void open(File f){
        myFile  = f;
        try {
            FileChannel channel = new FileInputStream(myFile).getChannel();
            myBuffer = channel.map(FileChannel.MapMode.READ_ONLY, 0, (int) channel.size());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    protected boolean wordPart(char ch){
        return Character.isLetter(ch) || ch == '\'';
    }


    public void readAndProcess(IStringProcessor isp){
        boolean inWord = false;
        int length = 0;
        int start = 0;
        StringBuilder builder = new StringBuilder();

        // read from beginning of buffer, one char at a time
    myBuffer.position(0);
    for(int k=0; k < myBuffer.limit(); k++){
        char ch = (char) myBuffer.get();
            if (inWord){
                if (! wordPart(ch)){
                    // word ended, process it
                    String s = builder.toString();
                    builder = builder.delete(0,builder.length());
                    isp.process(s);
                    length = 0;
                    inWord = false;
                }
                else {
                    builder.append(Character.toLowerCase(ch));
                    length++;
                }
            }
            else { // not in word, does a word start?
                if (Character.isLetter(ch)){
                    inWord = true;
                    start = k;
                    length = 1;
                    builder.append(Character.toLowerCase(ch));
                }
            }
        }
        if (length > 0){
            isp.process(builder.substring(start,start+length));
        }
    }
}
```