```
import java.net.*;
import java.util.*;

import javax.swing.text.MutableAttributeSet;
import javax.swing.text.html.*;
import java.io.*;

/**
 * Illustrates (very simply) how to create and read URLs, how to parse
 * HTML read from a URL, and the Iterator design pattern to
 * process all the relative URLs read from a specified URL.
 * <P>
 * This example is meant to show simply how things work.
 * <P>
 * To create a UrlFileTracker, construct one using
 * an "http://..." base URL, and all the *.txt files linked from
 * the base URL will then be accessible via a BufferedReader returned
 * by iterating over the UrlFileTracker (which implements the
 * Iterator interface).
 * <P>
 * @author Owen Astrachan
 *
 */

public class UrlFileTracker implements Iterator {

    private String[] myTextURLs;   // relative URLs from main page
    private URL myBaseURL;         // base URL for all .txt urls
    private int myCurrentIndex;    // current URL when iterating
    private ArrayList myURList;    // filled when parsing, refactor!

    /**
     * Construct an iterable-over UrlFileTracker from
     * a baseURL --- the assumption is that the base URL is
     * a directory from which .txt files will be obtained.
     * @param baseURL for all the .txt files
     */
    public UrlFileTracker(String baseURL){
        myURList = new ArrayList();
        myCurrentIndex = 0;
        getURLs(baseURL);
    }

    /**
     * Find all relativie URLs and store in array myTextURLs
     * @param baseURL is base of all URLs
     */
    private void getURLs(String baseURL){

        try {
            myBaseURL = new URL(baseURL);
            myURList.clear();
            BufferedReader reader =
                new BufferedReader(new InputStreamReader(myBaseURL.openStream())
);
            LocalParser local = new LocalParser();
            HTMLEditorKit.Parser parser = local.getParser();
            parser.parse(reader, new LocalCallback(),true);
            myTextURLs = (String[]) myURList.toArray(new String[0]);

        } catch (MalformedURLException e) {
            System.err.println("trouble with format of URL "+myBaseURL.getPath());
            e.printStackTrace();
        } catch (IOException e) {
            System.err.println("trouble reading URL "+myBaseURL.getPath());
            e.printStackTrace();
        }
    }
```



```
    public static void main(String[] args) throws IOException{
        String url = "http://www.cs.duke.edu/courses/cps108/fall05/assign/nanogoogle/guten/small/";
        UrlFileTracker uft = new UrlFileTracker(url);
        while (uft.hasNext()){
            BufferedReader reader = (BufferedReader) uft.next();
            String line;
            int count=0;
            while ((line = reader.readLine()) != null){
                count++;
            }
            System.out.println("read "+count+" lines");
        }
        uft.getURLs(url);
    }

    /**
     * Inner class for processing all "<a href.."> tags when
     * reading a base URL. This callback stores the relative
     * URLs in an ArrayList of the class it's nested in.
     * @author ola
     */
    private class LocalCallback extends HTMLEditorKit.ParserCallback{
        public void handleStartTag(HTML.Tag t, MutableAttributeSet a, int pos){
            if (t == HTML.Tag.A){
                String href = (String) a.getAttribute(HTML.Attribute.HREF);
                if (href.endsWith(".txt")){
                    myURList.add(href);
                }
            }
        }
    }

    /**
     * Returns status indicating if more relative .txt URLs are
     * accessible.
     * @return true if more relative URLs are available
     */
    public boolean hasNext() {
        return myCurrentIndex < myTextURLs.length;
    }


    /**
     * Return BufferedReader for next unprocessed .txt URL,
     * returns null if all are processed.
     * @return BufferedReader for the .txt URL
     */
    public Object next() {
        if (hasNext()){
            URL url = null;
            try {
                url = new URL(myBaseURL, myTextURLs[myCurrentIndex]);
                BufferedReader reader = new BufferedReader(new InputStreamReader
(url.openStream()));
                myCurrentIndex++;
                return reader;
            } catch (MalformedURLException e) {
                System.err.println("bad url "+url.getPath());
                e.printStackTrace();
            } catch (IOException e) {
                System.err.println("io error on url "+url.getPath());
                e.printStackTrace();
            }

        }
        return null;
    }
```

```
    /**
     * Required by Iterator interface, not implemented.
     */
    public void remove() {
        throw new UnsupportedOperationException("no remove");
    }
}

/**
 * Hack required because getParser() is protected
 * in HTMLEditorKit, so extend and make method less
 * restricted, e.g., public.
 * @author Owen Astrachan
 */
class LocalParser extends HTMLEditorKit{

    public HTMLEditorKit.Parser getParser(){
        return super.getParser();
    }
}
```