Reading a webpage

Reading a webpage can be useful. Here's an example. Website `genealogy.math.ndsu.nodak.edu` has over 218,300 records of PhDs in mathematics and computer science, showing their advisors and also the students they advised. You can envision the tree of a PhD, with the PhD at the root, their (one or two) advisors being the children, the advisors of the advisors being the children of the children, and so forth. But it's not easy to manually build a tree by clicking around, seeing who a PhDs advisors are, and copying the info. So, we wrote a program that reads webpages from this website and extracts data, and builds a representation of the tree for us.

Reading a webpage using the newer package `java.nio.file` is not easy. So, we show you how to read a webpage using the older package, `java.io` as well as class `java.net.URL`.

Class URL

An object of class `java.net.URL` represents a "Uniform Resource Locator" —a pointer to a resource on the World Wide Web. An example is

https://www.pgatour.com/players.htm

It consists of three parts:

1. The protocol: *https*, standing for "Hyper Text Transfer Protocol Secure", the secure version of protocol *http*. The S tells you that all communications between the browser and your website will be encrypted.
2. The host: *www.pgatour.com*. This gives the server on which this file resides.
3. A path: *players.htm*. This tells you where the file resides on the server.

There are other optional parts to a URL, but this gives you the basics. Visit the API spec of class URL to learn more.

Another example is the `URL` `http://tinyurl.com`[1]. Here, the path is empty. A specific default will be used.

A method to get a BufferedReader for a URL

Use the following method to create a `BufferedReader` for a `URL`. Call this method to get a `BufferedReader`; then just read as you would with any `BufferedReader`.

```
/**  Return a reader for URL url, but
  *  Return null if url is null or its protocol is not  http or https. */
private static BufferedReader getReader(URL url) throws IOException {
    if (url == null) return null;
    String p= url.getProtocol();
    if (!(p.equals("http")  ||  p.equals("https"))) return null;
    InputStream is= url.openStream();
    InputStreamReader isr= new InputStreamReader(is);
    return new BufferedReader(isr);
}
```

The try statement (1) constructs an `InputStream` for `url`, (2) wraps it in an `InputStreamReader isr`, and finally wraps that in `BufferedReader`. You can now use the returned `BufferedReader` to read the lines of the webpage. You don't have to know about `InputStreams` and `InputStreamReaders`.

This method is in the demos `iodemo.zip` found in item (1) of JavaHyperText entry "I/O".

[1] If you don't know about `tinyurl.com`, look at it! Suppose you have a *very long* URL that you want to send to friends, so they can look at that page. Website `tinyurl.com` will create a *very sh*ort URL that you can use in place of the long one, making it easier for you and your friends.