CS 11 java track: lecture 6

- This week:
  - networking basics
  - **`Socket`s**
  - **`Vector`s**
  - parsing strings

what is networking?

- the network:
  - world-wide web ☺ of interconnected computers
  - "the internet"
- networking:
  - programming computers to interact with the network
- examples:
  - download web pages
  - send email
  - instant messaging
  - etc.

this week's assignment

- write a simple web crawler
  - download web pages
  - scan through text, looking for hyperlinks
  - store hyperlinks
  - download web pages hyperlinks point to
  - etc.
  - at end, print out hyperlinks

networking terminology (1)

- URL
  - Uniform Resource Locator
  - used to identify location of individual web pages
  - consists of:
    - string "http://"
    - followed by location of web server *e.g.* "www.cs.caltech.edu"
    - followed by path of web page on the server *e.g.* "courses/cs11/index.html"
  - other types of URLs as well (we'll ignore them)

networking terminology (2)

- HTML
  - Hypertext Markup Language
  - language for writing web pages
    - regular text "marked up" with tags
    - hypertext refers to hyperlinks between pages
  - hyperlinks look like this:
    - **<a href="http://path/to/host/path/to/file.html">some text</a>**
  - URL is embedded in start tag
  - displayed in browser with "some text" underlined
  - clicking on "some text" sends you to another page

networking terminology (3)

- HTTP
  - Hypertext Transfer Protocol
  - text-based format for transmitting web page data over the internet
  - latest version is 1.1
- typical HTTP query:

```
GET /courses/cs11/index.html HTTP/1.1
HOST: www.cs.caltech.edu
Connection: close
<blank line>
```

- request must end in a blank line

networking terminology (4)

- socket
  - software-defined entity (data structure)
  - allows for two-way (send/receive) communication to/from a URL
  - sockets don't have to use HTTP
    - but typically do for downloading web pages
  - technically, sockets use TCP/IP protocols
    - Transmission Control Protocol / Internet Protocol
    - lower level; HTTP rides on top of this

networking terminology (5)

- port
  - location on web server that a socket can bind to
  - identified with a number
  - usually use port 80 for HTTP connections

networking in java (1)

- `java.net` package contains networking classes
- also need `java.io` for streams (`InputStream`, `PrintWriter` etc.)
- `Socket` class creates new sockets
- typical usage:
  - open new socket for each web page to be downloaded
  - send HTTP request
  - receive (download) data
  - close socket

networking in java (2)

- Constructor:

  `Socket(String host, int port)`

  - host is not the entire URL (just the host name)
  - *e.g.* in **`http://www.caltech.edu/foo/bar/index.html`** it's just "**`www.caltech.edu`**"
  - rest of URL is in HTTP request
  - port is 80

networking in java (3)

- methods:
  - **`setSoTimeout(int milliseconds)`**
    - sets a timeout on socket reads
  - **`getOutputStream()`**
    - allows writing to a socket

```
Socket mySocket = new Socket(...);
PrintWriter out = new
  PrintWriter(mySocket.getOutputStream(), true);
out.println(...);  // HTTP request
```

networking in java (4)

- methods, continued:
- getting input data from a stream
  - **`getInputStream()`** method of **`Socket`**
  - allows reading from the other end of the socket
  - typical usage:

```
BufferedReader in = new BufferedReader(new
  InputStreamReader(mySocket.getInputStream()));
```

  - **`BufferedReader`** makes reading more efficient

networking in java (5)

- more useful methods:
  - **`BufferedReader:`**
    - **`boolean ready()`**

      ```
      // buffer is ready to read
      // good to call before reading data
      ```
    - **`String readLine()`**
  - **`System.currentTimeMillis()`**
    - returns current time in milliseconds
    - can use to monitor time spent waiting for response

networking in java (6)

- odds and ends:
- socket provides input/output streams
  - so reading/writing to/from just like with any other streams
- after input received
  - scan line for URLs
  - need `String` parsing methods (coming up)

Vectors (1)

- java arrays store a fixed number of elements of a given type
- sometimes want an array that can grow
- **`Vector`** class fills the bill
- in **`java.util`** package
- holds arbitrary number of elements of arbitrary object types

Vectors (2)

- using Vectors:

```
MyClass m = new MyClass();
MyClass n;
Vector v = new Vector();
v.add(m);  // any object can be added
n = (MyClass)v.elementAt(0); // need to cast
// Not casting is a type error!
// Casting to wrong type gives a
// ClassCastException.
```

- other methods: `size()`, `remove(int)`

String parsing

- need to search for URLs in Strings; useful methods:
  - **`substring(int beginIndex)`**
    - starts at **`beginIndex`**, goes to end of String
  - **`substring(int beginIndex, int endIndex)`**
    - starts at **`beginIndex`**, goes to **`(endIndex - 1)`**
  - **`length()`**
    - not **`size()`**!
  - **`indexOf(char c)`**
    - first index of c in String
  - **`indexOf(char c, int pos)`**
    - first index of c starting from pos

next week

- multithreading!
- the synchronized keyword
  - the horror, the horror!
- hardest topic of course
- don't miss that lecture!