CSE 331
Software Design & Implementation

Kevin Zatloukal

Fall 2020

Servers

Event-driven programming

An *event-driven* program is designed to wait for events:

- program initializes then enters the *event loop*
- abstractly:

```
do {
    e = getNextEvent();
    process event e;
} while (e != quit);
```

Server Programming

- Servers sit around waiting for events like:
  - new client connections
  - new data from the client (high scale servers)

- Simple version (normal scale):

```
while (true) {
  wait for a client to connect
  process the request; send a response back
}
```

  - probably want to use a new thread for processing
  - high scale web servers might look quite different

Example: Chat Server

`ChatServer.java`

Server Sockets & Ports

- Server creates a "server socket" and waits for a connection
  - each connection comes with an individual socket
  - allows reading from / writing to that client

- Servers on the same machine distinguished by a **port** number
  - numbers below 1024 require admin privileges

```
ServerSocket ssock = new ServerSocket(80);
```

- Clients indicate the port when trying to connect:

```
Socket sock = new Socket("attu", 80);
```

Ports & Protocols

- Sockets API allows reading & writing of byte data
  - like the File API

- Each server can define its own **protocol** for communication
  - the language it uses to speak to clients

- By convention, ports are associated with particular protocols
  - 80 = HTTP
  - 443 = HTTPS
  - 25 = SMTP relay
  - …

- Client that wants to talk HTTP can try connecting to 80

Protocols

- HTTP (Hyper-Text Transfer Protocol) is the most important
  - initially created for retrieving HTML documents
  - simple, text-based protocol

- Trend moving away from new protocols toward re-use of HTTP
  - Google (2010s) used HTTP for almost everything

- Allows for re-use of **libraries** for creating HTTP servers…
  - use of libraries reduces bugs, saves time, etc.
  - do not write your own HTTP server