Notes on Java sockets

Socket Interface

- Need an interface for applications on end--hosts to set up and send/receive data over end--to--end communication channels
  - Berkeley sockets interface --- originally provided by BSD 4.1 in about 1982
  - Java interface is slightly different from the C/C++ interface

- Java methods
  - Socket()
    - Create a new socket
    - Socket is similar to a file descriptor
    - Uses TCP underneath to provide reliable delivery of a stream of bytes; data is automatically broken into packets by the network stack in the OS
    - To send individual packets (unreliably), use DatagramSocket, which uses UDP underneath
  - bind(SocketAddress bindpoint)
    - Associates a network layer (IP) address and port with the socket
    - This is the address and port used on this host
  - connect(SocketAddress endpoint)
    - Sets up the socket to communicate with a specific remote host
    - When using a Socket (i.e., TCP), a handshake occurs to establish the reliable communication channel
    - Specify the address and port used on the remote host
    - Not called when using a DatagramSocket (i.e., UDP)
  - accept()
    - Wait for an incoming connection request
    - Only used with TCP
    - Use with a ServerSocket() rather than just a Socket()
    - When a connection is accepted, a Socket object for that specific connection is returned; can call accept on ServerSocket again to accept another connection from a different host/port
  - read() (from InputStream)
    - With TCP, read bytes sent by the other side
    - With UDP, receive a packet sent by the other side; since UDP is connectionless, packet could have come from anyone --- call getAddress() and getPort()
  - write (to OutputStream)
    - With TCP, write bytes to send to the other send
    - With UDP, send a packet to the other side; since UDP is connectionless, you need to provide the address and port for the remote host when constructing the packet
  - close()
    - Close the socket
- With TCP, terminate the connection

Example

Client

1. `c = new Socket()`
2. `c.connect(SocketAddress remote)`
3. `c.getOutputStream().write()`
4. `c.getInputStream().read()`
5. `c.close()`

Server

1. `s = new ServerSocket()`
2. `s.bind(SocketAddress local)`
3. `Socket c = s.accept()`
4. `c.getOutputStream().write()`
5. `c.getInputStream().read()`
6. `c.close()`
7. `s.close()`