15-440 Distributed Systems Recitation 2

Laila Elbeheiry
Slides adopted from: Tamim Jabban

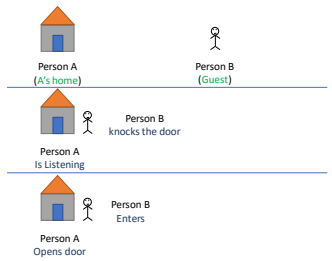
Agenda

- **Communication via Sockets in Java**
  (*this enables you to complete PS1 and start P1*)
- **Multi-threading in Java**
- **Coding a full Client-Server Example**
  On Eclipse, we'll code an "echo" TCP Server-Client Example

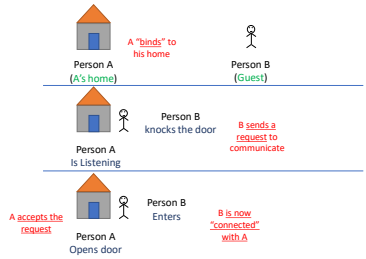
Communication via Sockets

- Sockets provide a communication mechanism between networked computers.
- A Socket is an end-point of communication that is identified by an IP address and port number.
- A client sends requests to a server using a client socket.
- A server receives clients' requests via a listening socket

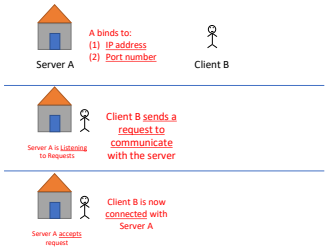
Communication via Sockets



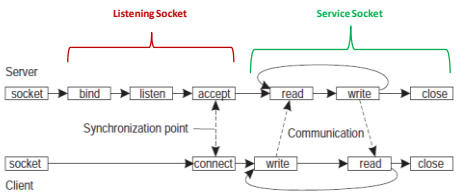
Communication via Sockets



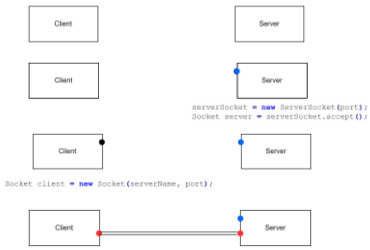
Communication via Sockets



Communication via Sockets



Socket Communication Recipe



ServerSocket Methods

SN	Methods with Description
1	**public ServerSocket(int port)** Attempts to create a server socket bound to the specified port. An exception occurs if the port is already bound by another application.
2	**public ServerSocket()** Creates an unbound server socket. When using this constructor, use the bind() method when you are ready to bind the server socket.
3	**public Socket accept()** Waits for an incoming client. This method blocks until either a client connects to the server on the specified port or the socket times out, assuming that the time-out value has been set using the setSoTimeout() method. Otherwise, this method blocks indefinitely.
4	**public void bind(SocketAddress host)** Binds the socket to the specified server and port in the SocketAddress object. Use this method if you instantiated the ServerSocket using the no-argument constructor.
5	**public SocketAddress getLocalSocketAddress()** Returns the address of the endpoint this socket is bound to, or null if it not bound yet.

Socket Methods

SN	Methods with Description
1	**public Socket(String host, int port)** This method attempts to connect to the specified server at the specified port. If this constructor does not throw an exception, the connection is successful and the client is connected to the server.
2	**public Socket()** Creates an unconnected socket. Use the connect() method to connect this socket to a server.
3	**public void connect(SocketAddress host)** This method connects the socket to the specified host. This method is needed only when you instantiated the Socket using the no-argument constructor.
4	**public InputStream getInputStream()** Returns the input stream of the socket. The input stream is connected to the output stream of the remote socket.
5	**public OutputStream getOutputStream()** Returns the output stream of the socket. The output stream is connected to the input stream of the remote socket
6	**public SocketAddress getLocalSocketAddress()** Returns the address of the endpoint this socket is bound to, or null if it is not bound yet.
7	**public void close()** Closes the socket, which makes this Socket object no longer capable of connecting again to any server

Demo Time ☺

Multi-Threading

- A class intended to execute as a thread must implement the ***Runnable*** interface

```
public class Service implements Runnable
```

- STEP 1: Implement the method ***run()***

```
public void run() {  //thread's logic goes here }
```

- STEP 2: Instantiate a Thread object

```
Thread t = new Thread(new Service())
```

- STEP 5: Invoke ***start()*** on the new thread

```
t.start()          // invokes the run() method
```

Transport Protocols

- Socket: endpoint to read and write data
- Each Socket has a **network protocol**
- Two types of **protocols** used for communicating data/*packets* over the internet:
  - TCP:
    - ***Transmission Control Protocol***
    - Connection Oriented (*handshake*)
  - UDP:
    - ***User Datagram Protocol***
    - "Connectionless"

Transport Protocols