CS 351
Design of Large Programs
Sockets Example

Brooke Chenoweth

University of New Mexico

Spring 2024

Socket

- `Socket(String host, int port)`
- `InputStream getInputStream()`
- `OutputStream getOutputStream()`
- `void close()`

ServerSocket

- `ServerSocket(int port)`
- `Socket accept()` – blocks until connection is made
- `void close()`

Knock Knock Example

- Adapted from sockets tutorial on Oracle site.
- KnockKnockServer is running on particular machine and port.
- KnockKnockClient programs connect to server and are told a joke.

KnockKnockClient: main

```
public static void main(String[] args) throws IOException {

  String hostName = args[0];
  int portNumber = Integer.parseInt(args[1]);

  try (Socket socket = new Socket(hostName, portNumber);
       PrintWriter out = new PrintWriter(socket.getOutputStream(), true);
       BufferedReader in =
         new BufferedReader(new InputStreamReader(socket.getInputStream()));
      ) {
    BufferedReader stdIn =
      new BufferedReader(new InputStreamReader(System.in));
    String fromServer = in.readLine();
    while(fromServer != null) {
      System.out.println("Server: " + fromServer);
      if(fromServer.equals("Bye.")) { break; }
      String fromUser = stdIn.readLine();
      if(fromUser != null) {
        System.out.println("Client: " + fromUser);
        out.println(fromUser);
      }
      fromServer = in.readLine();
    }
  }
}
```

KnockKnockServer: main

```
public static void main(String[] args)
  throws IOException {

  int portNumber = Integer.parseInt(args[0]);

  ServerSocket serverSocket =
    new ServerSocket(portNumber);

  // Listen for new clients forever
  while(true) {
    // Create new thread to handle each client
    Socket clientSocket = serverSocket.accept();
    KnockKnock kk = new KnockKnock(clientSocket);
    Thread t = new Thread(kk);
    t.start();
  }
}
```

KnockKnockServer: constants

```
private static String BYE = "Bye.";

private enum State {
   WAITING,
   SENT_KNOCK_KNOCK,
   SENT_CLUE,
   ANOTHER
}

private static String[] clues =
   { "Turnip",
     "Little Old Lady",
     "Atch", "Who", "Who" };
private static String[] answers =
   { "Turnip the heat, it's cold in here!",
     "I didn't know you could yodel!",
     "Bless you!",
     "Is there an owl in here?",
     "Is there an echo in here?" };
```

KnockKnock: init

```
public static class KnockKnock implements Runnable {
  private final Socket clientSocket;
  private PrintWriter out;
  private BufferedReader in;

  private State state = State.WAITING;
  private int currentJoke = 0;

  public KnockKnock(Socket clientSocket)
    throws IOException {
    this.clientSocket = clientSocket;
    out =
      new PrintWriter(clientSocket.getOutputStream(),
                      true);
    in = new BufferedReader(new InputStreamReader(
           clientSocket.getInputStream()));
  }
```

KnockKnock: run

```
public void run() {
  String inputLine = null;
  String outputLine;

  do {
    outputLine = processInput(inputLine);
    out.println(outputLine);
    if(outputLine.equals(BYE)) {
      break;
    }
    try {
      inputLine = in.readLine();
    } catch (IOException ex) {
      inputLine = null;
    }
  } while(inputLine != null);
}
```

KnockKnock: processInput 1

```
private String processInput(String input) {
  String output = null;
  switch (state) {
    case WAITING:
      output = "Knock! Knock!";
      state = State.SENT_KNOCK_KNOCK;
      break;
    case SENT_KNOCK_KNOCK:
      if(input.equalsIgnoreCase("Who's there?")) {
        output = clues[currentJoke];
        state = State.SENT_CLUE;
      } else {
        output =
        "You're supposed to say \"Who's there?\"! " +
        "Try again. Knock! Knock!";
      }
      break;
```

KnockKnock: processInput 2

```
  case SENT_CLUE:
    if(input.equalsIgnoreCase(clues[currentJoke] + " who?")) {
      output = answers[currentJoke] +
        " Want another? (y/n)";
      state = State.ANOTHER;
    } else {
      output = "You're supposed to say \"" +
        clues[currentJoke] + " who?\"! " +
        "Try again. Knock! Knock!";
      state = State.SENT_KNOCK_KNOCK;
    }
    break;
    case ANOTHER:
      if(input.equalsIgnoreCase("y")) {
        output = "Knock! Knock!";
        currentJoke = (currentJoke + 1) % clues.length;
        state = State.SENT_KNOCK_KNOCK;
      } else {
        output = BYE;
        state = State.WAITING;
      }
      break;
  }
  return output;
}
```

Knock Knock Protocol