SER 321 A Session

SI Session

Sunday, February 2nd 2025

7:00 pm - 8:00 pm MST

Agenda

- OSI Review Challenge
- HTTP Review
- JSON Syntax Review & Practice
- Socket Review
  - Properties & Steps
  - Port Examination

SI Session Expectations

Thanks for coming to the **SER 321** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
  - tutoring.asu.edu
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

Interact with us:

Zoom Features

Zoom Chat

- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data		
Segment		
Packet		
Frame		
Bits		

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data		
Segment		
Packet		
Frame		
Bits	Physical	Signal, Binary transmission

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data		
Segment		
Packet		
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data		
Segment		
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data		
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data		
Data	Session	AuthN, authZ, session mgmt
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

SER 321

OSI Model

Unit	Layer	What we are *really* talking about
Data		
Data	Presentation	Translation, compression, encryption
Data	Session	AuthN, authZ, session mgmt
Segment	Transport	TCP/UDP
Packet	Network	IP address, routing and delivery
Frame	Data Link	LLC, MAC, data transmission in LAN
Bits	Physical	Signal, Binary transmission

MDN HTTP Request Methods

SER 321

HTTP Requests

What are the ***FOUR*** request types we reviewed?

1.

2.

3.

4.

MDN HTTP Request Methods

SER 321

HTTP Requests

What's the difference?

1. GET
2. POST
3. PUT
4. DELETE

SER 321

HTTP Matching

Match the HTTP response code with its meaning:

Code:	Meaning:
1XX	User Error
2XX	Server Error
3XX	Information
4XX	Redirect
5XX	Success

SER 321

HTTP Matching

Match the HTTP response code with its meaning:

Code:	Meaning:
1XX	Information
2XX	Success
3XX	Redirect
4XX	User Error
5XX	Server Error

org.json Docs

JSON Guide

SER 321

JSON Structure

Data is stored in…

Name:Value pairs → AKA → Members

"Katie"

"student" : "Katie"

What uses curly braces?

Objects

{ }

What do Objects contain?

Members

org.json Docs

JSON Guide

SER 321

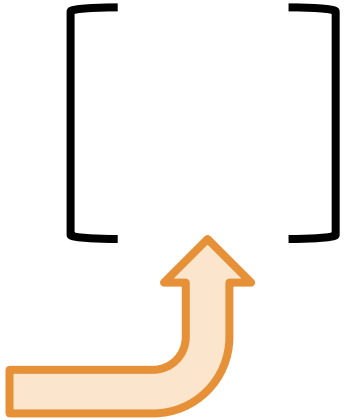
JSON Structure

What uses brackets?

Arrays

What do Arrays contain?

Any ***Valid*** Value



Members → Name : Value

Objects → { Members }

org.json Docs

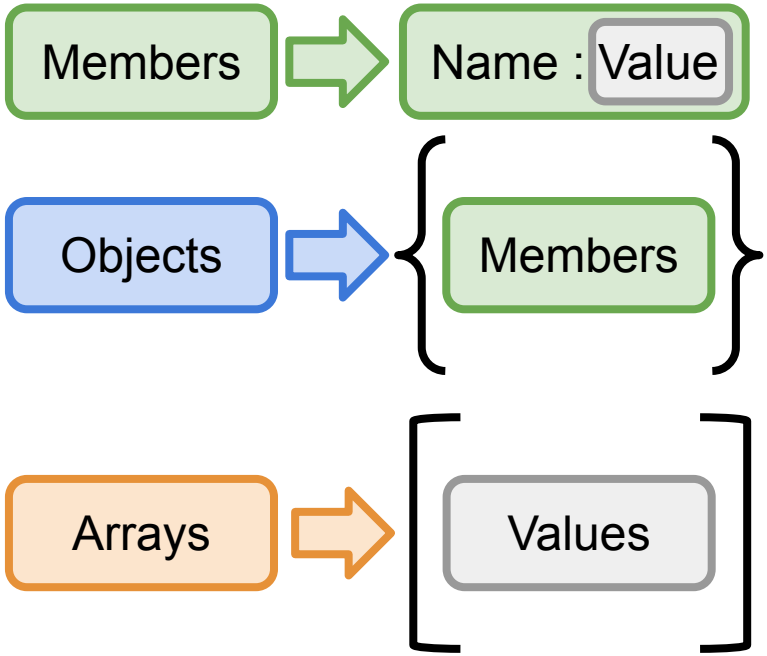
JSON Guide

SER 321

JSON Structure

What is a valid value?

- Strings
- Booleans
- Numbers
- NULL
- Objects
- Arrays



org.json Docs

JSON Guide

SER 321

JSON Recognition

How many Objects?

How many Arrays?

How many Members?


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 14.97,
    "wind_deg": 280,
    "wind_gust": 21.85,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  }
}
```

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 14.97,
    "wind_deg": 280,
    "wind_gust": 21.85,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  }
}
```


How would we…

Check for the timezone member?

```
boolean hasTimezone =
```

Get the timezone?

```
String timezone =
```

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "wind_deg": 280
    "id": 800
```


How would we…

Obtain the temp value?

~~String temp = json.getString("temp");~~

Recall that nested members require multiple steps!

Step 1:

Step 2:

Step 3:

Step 4:

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "wind_deg": 280
    "id": 800
  }
}
```


Recall that nested members require multiple steps!

How would we…

Obtain the temp value?

~~String temp = json.getString("temp");~~

```
if (json.has("current") {

}
```

Step 1: Check for parent object

Step 2:

Step 3:

Step 4:

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "wind_deg": 280
    "id": 800
```


Recall that nested members require multiple steps!

How would we…

Obtain the temp value?

~~String temp = json.getString("temp");~~

```
if (json.has("current") {

   JSONObject current =
            json.getObject("current");

 }
```

Step 1: Check for parent object

Step 2: Obtain parent object

Step 3:

Step 4:

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "wind_deg": 280
    "id": 800
  }
}
```


Recall that nested members require multiple steps!

How would we…

Obtain the temp value?

~~String temp = json.getString("temp");~~

```
if (json.has("current") {

   JSONObject current =
            json.getObject("current");

   if (current.has("temp")) {
      temp = current.getString("temp");
}
```

Step 1: Check for parent object

Step 2: Obtain parent object

Step 3: Check for nested member

Step 4:

org.json Docs

JSON Guide

SER 321

JSON Practice

```
JSONObject json =
```


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 14.97,
    "wind_deg": 280,
    "wind_gust": 21.85,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  }
}
```


How would we create the "weather" object?

org.json Docs

JSON Guide

SER 321

JSON Practice

JSONObject json =


```
{
  "lat": 42.3434,
  "lon": -88.0412,
  "timezone": "America/Chicago",
  "timezone_offset": -21600,
  "current": {
    "dt": 1733070576,
    "sunrise": 1733058144,
    "sunset": 1733091649,
    "temp": 18.57,
    "feels_like": 5.97,
    "pressure": 1025,
    "humidity": 63,
    "dew_point": 9.21,
    "uvi": 0.79,
    "clouds": 0,
    "visibility": 10000,
    "wind_speed": 14.97,
    "wind_deg": 280,
    "wind_gust": 21.85,
    "weather": [
      {
        "id": 800,
        "main": "Clear",
        "description": "clear sky",
        "icon": "01d"
      }
    ]
  }
}
```


How would we create the "weather" object?

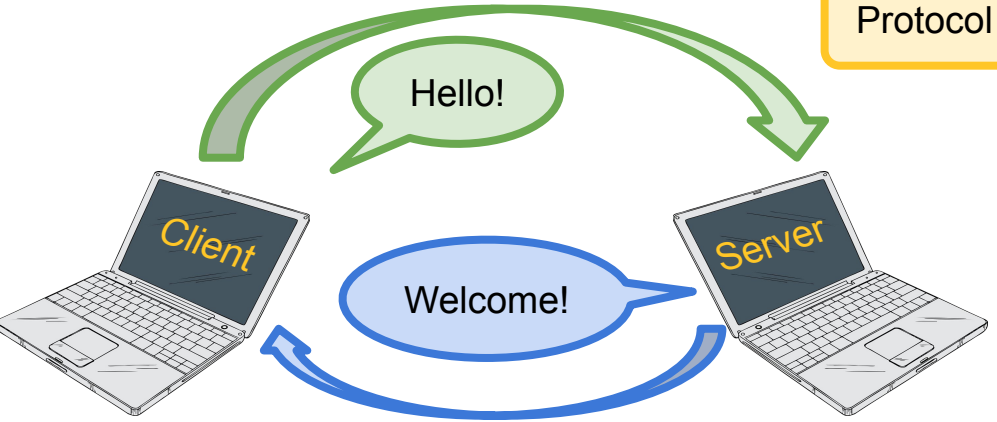
```
JSONObject json = new JSONObject();
JSONObject weather = new JSONArray();
JSONObject content = new JSONObject();
content.put("id", 800);
content.put("main", "Clear");
content.put("description", "clear sky");
content.put("icon", "01d");

weather.put(content.toMap());

json.put(weather.toMap());
```

SER 321
Socket Properties

Sockets allow our client and server to communicate!

Need to define **3 properties** before usage

Location	IP or DNS	142.251.46.206	www.google.com
Connection Semantics	TCP or UDP	Connection Oriented	Connectionless
Message Format	Protocol Specs	Synchronous	Asynchronous
		Stateless	Stateful
		Binary	Text
		Headers	No Headers



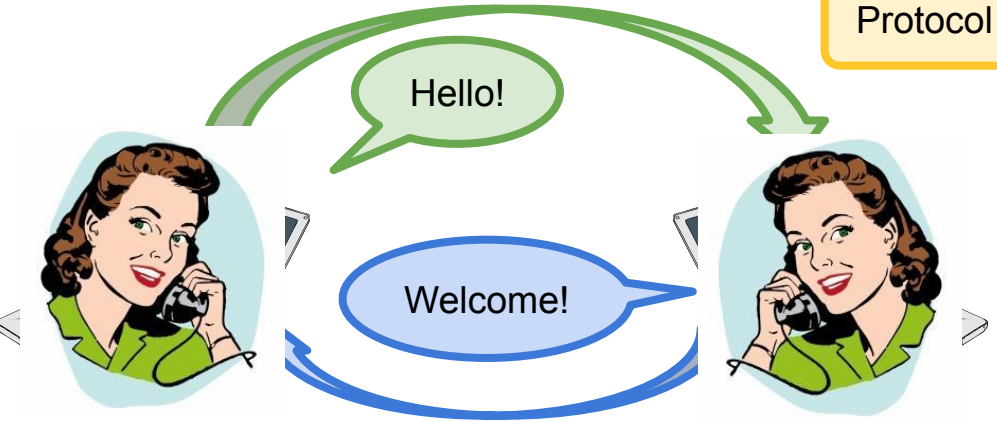
SER 321

Socket Properties

Sockets allow our client and server to communicate!

d to define **3 properties** before usage

Person	IP or DNS	142.251.46.206	www.google.com
Conversation Flow	TCP or UDP	Connection Oriented	Connectionless
Conversation Content	Protocol Specs	Synchronous	Asynchronous
		Stateless	Stateful
		Binary	Text
		Headers	No Headers



SER 321

Socket Protocol Types

Two Main Conversation Models

1. Client Request

2. Server Response

Pull/Polling Model

Client *pulls* info from the server

Client *polls* the server for info

SER 321

Socket Protocol Types

Two Main Conversation Models

1. Server sends update

2. Client acknowledges

Push Model

Server *pushes* info to client

Push notifications

SER 321

Socket Protocol Types

Two Main Conversation Models

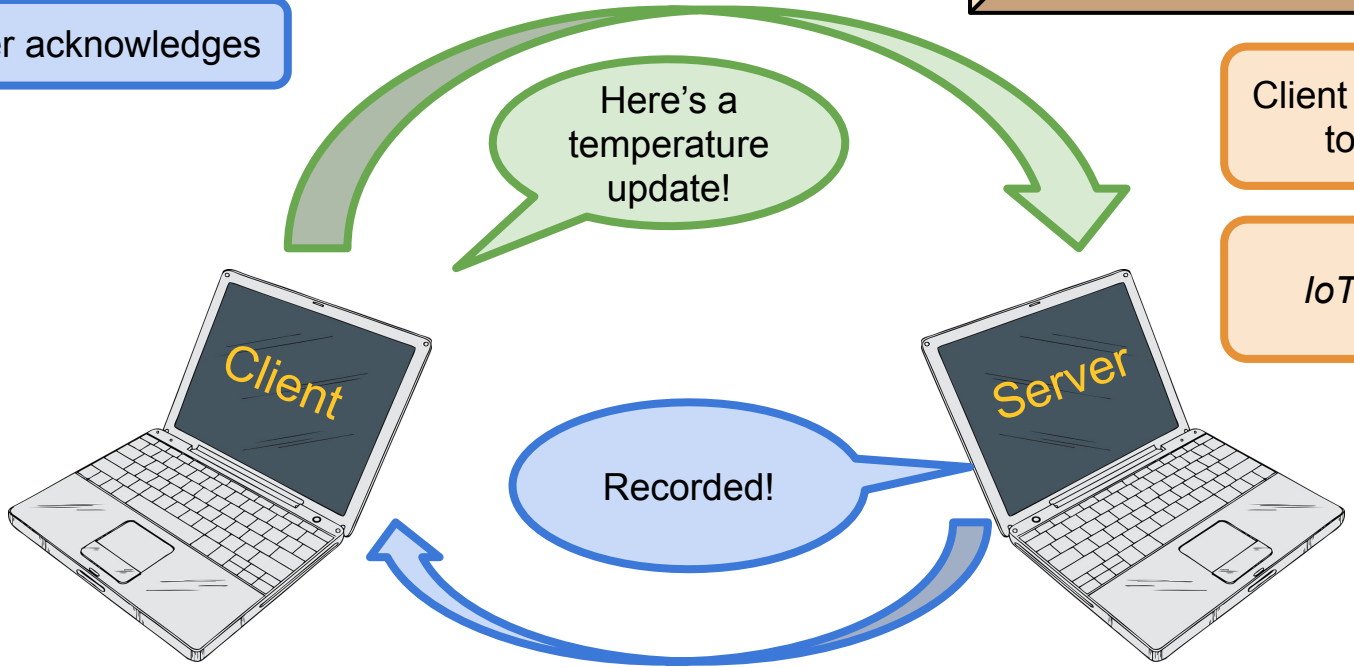
1. Client sends update

2. Server acknowledges

Client Push Model

Client *pushes* info to Server

IoT sensors



SER 321

Client Socket

Put the Steps for the **Client Socket** in the correct order:

1.

2.

3.

4.

5.

6.

7.

8.

A. Send Message

B. Close Socket

C. Define Params

D. Create Param Struct

E. Receive Message

F. Create Socket

G. Repeat

H. Establish Connection

SER 321

Client Socket

Put the Steps for the **Client Socket** in the correct order:

C	1. Define Params
F	2. Create Socket
D	3. **C ONLY** Create a struct for the address
H	4. Establish Connection
A	5. Send Message
E	6. Receive Message
G	7. Repeat #5 and #6 as needed
B	8. Close Socket

A. Send Message

B. Close Socket

C. Define Params

D. Create Param Struct

E. Receive Message

F. Create Socket

G. Repeat

H. Establish Connection

SER 321

Server Socket

Put the Steps for the **Server Socket** in the correct order:

1.

2.

3.

4.

5.

6.

7.

8.

9.

A. Mark Socket to Listen

B. Close Socket

C. Define Params

D. Create Param Struct

E. Continue Listening

F. Handle Client

G. Wait for Connection

H. Bind Socket to Address

I. Create Socket

SER 321

Server Socket

Put the Steps for the **Server Socket** in the correct order:

F	1. Define Params
I	2. Create Socket
E	3. **C ONLY** Create a struct for the address
H	4. Bind Socket to Local Address
A	5. Mark Socket to Listen for Connections
C	6. Wait for Connection
B	7. Handle Client Connection
G	8. Close Client Connection
D	9. Continue Listening for Connections

A. Mark Socket to Listen

B. Handle Client

C. Wait for Connection

D. Continue Listening

E. Create Param Struct

F. Define Params

G. Close Socket

H. Bind Socket to Address

I. Create Socket

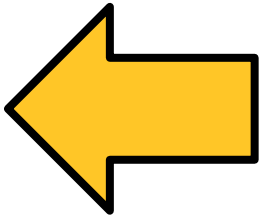
Assign 3-1 Starter Code

SER 321

Server Socket

What needs to be done here?

1. Define Params
2. Create Socket
3-5. Mark Socket to Listen
6. Wait for Connection
7. Handle Client Connection
8. Close Client Connection
9. Continue Listening



1

2

3

4

5

Assign 3-1 Starter Code

SER 321
Server Socket

What needs to be done here?

```
sock = serv.accept(); // blocking wait
System.out.println("Client connected");
```

1. Define Params
2. Create Socket
3-5. Mark Socket to Listen
6. Wait for Connection
7. Handle Client Connection
8. Close Client Connection
9. Continue Listening

1

2

3

4

5

Assign 3-1 Starter Code

Echo_Java

SER 321

Server Socket

What needs to be done here?

Is input
from the client
or
to the client ?

1. Define Params

```
// setup the object reading channel
in = new ObjectInputStream(sock.getInputStream());

// get output channel
OutputStream out = sock.getOutputStream();

// create an object output writer (Java only)
os = new DataOutputStream(out);
```

```
clientSock = sock.accept(); // blocking wait
PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
InputStream input = clientSock.getInputStream();
System.out.println("Server connected to client");
```

1

2

3

4

5

Assign 3-1 Starter Code

SER 321

Server Socket

What needs to be done here?

```
static void overandout() {
  try {
    os.close();
    in.close();
    sock.close();
  } catch(Exception e) {e.printStackTrace();}

}
```

```
  try {
    s = (String) in.readObject();
  } catch (Exception e) {
    System.out.println("Client disconnect");
    connected = false;
    continue;
  }
```

1 Create input/output streams

2

3

4

5

Assign 3-1 Starter Code

SER 321

Server Socket

What needs to be done here?

```java
JSONObject res = isValid(s);

if (res.has( key: "ok")) {
  writeOut(res);
  continue;
}

JSONObject req = new JSONObject(s);

res = testField(req,  key: "type");
if (!res.getBoolean( key: "ok")) {
  res = noType(req);
  writeOut(res);
  continue;
}
```

```java
public static JSONObject isValid(String json) {
  try {
```

```java
static JSONObject testField(JSONObject req, String key){
  JSONObject res = new JSONObject();

  // field does not exist
  if (!req.has(key)){
    res.put("ok", false);
    res.put("message", "Field " + key + " does not exist in request");
    return res;
  }
  return res.put("ok", true);
}
```

```java
      return res;
    }
  }
  return new JSONObject();
}
```

Assign 3-1 Starter Code

Echo_Java

SER 321

Server Socket

What needs to be done here?

```
int numr = input.read(clientInput, off: 0, bufLen);
```

```
String received = new String(clientInput, offset: 0, numr);
System.out.println("read from client: " + received);
out.println(received);
```

```
if (req.getString( key: "type").equals("echo")) {
  res = echo(req);
} else if (req.getString( key: "type").equals("add")) {
  res = add(req);
} else if (req.getString( key: "type").equals("addmany")) {
  res = addmany(req);
} else {
  res = wrongType(req);
}
writeOut(res);
```

1 Create input/output streams

2 Check for disconnect

3 Check Protocol

4

5

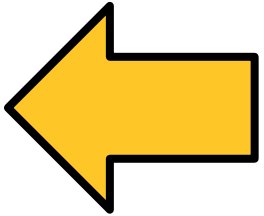
Assign 3-1 Starter Code

SER 321

Server Socket

What needs to be done here?

1. Define Params
2. Create Socket
3-5. Mark Socket to Listen
6. Wait for Connection
7. Handle Client Connection
8. Close Client Connection
9. Continue Listening



1 Create input/output streams

2 Check for disconnect

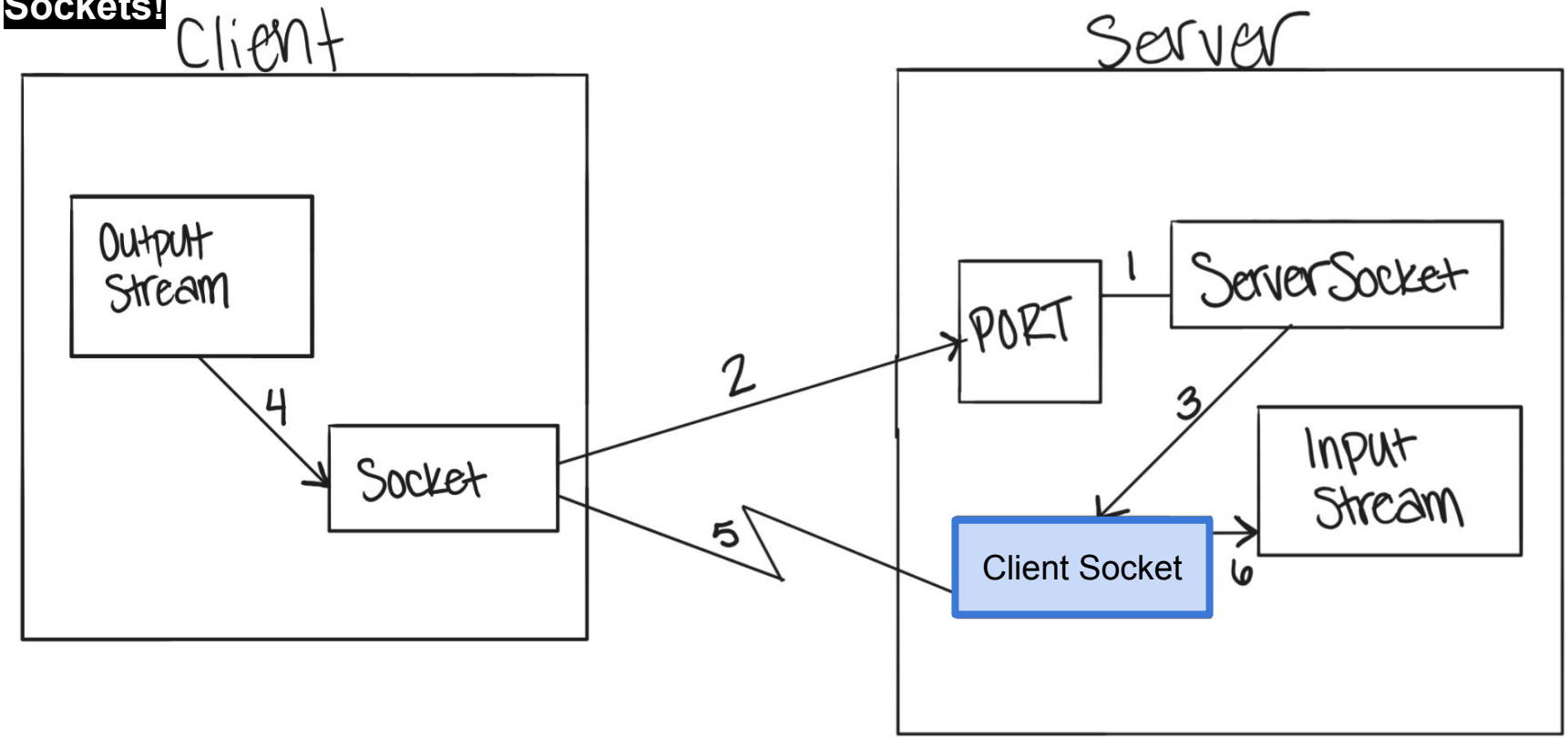
3 Check Protocol

4 Read Headers

5 Handle Accordingly

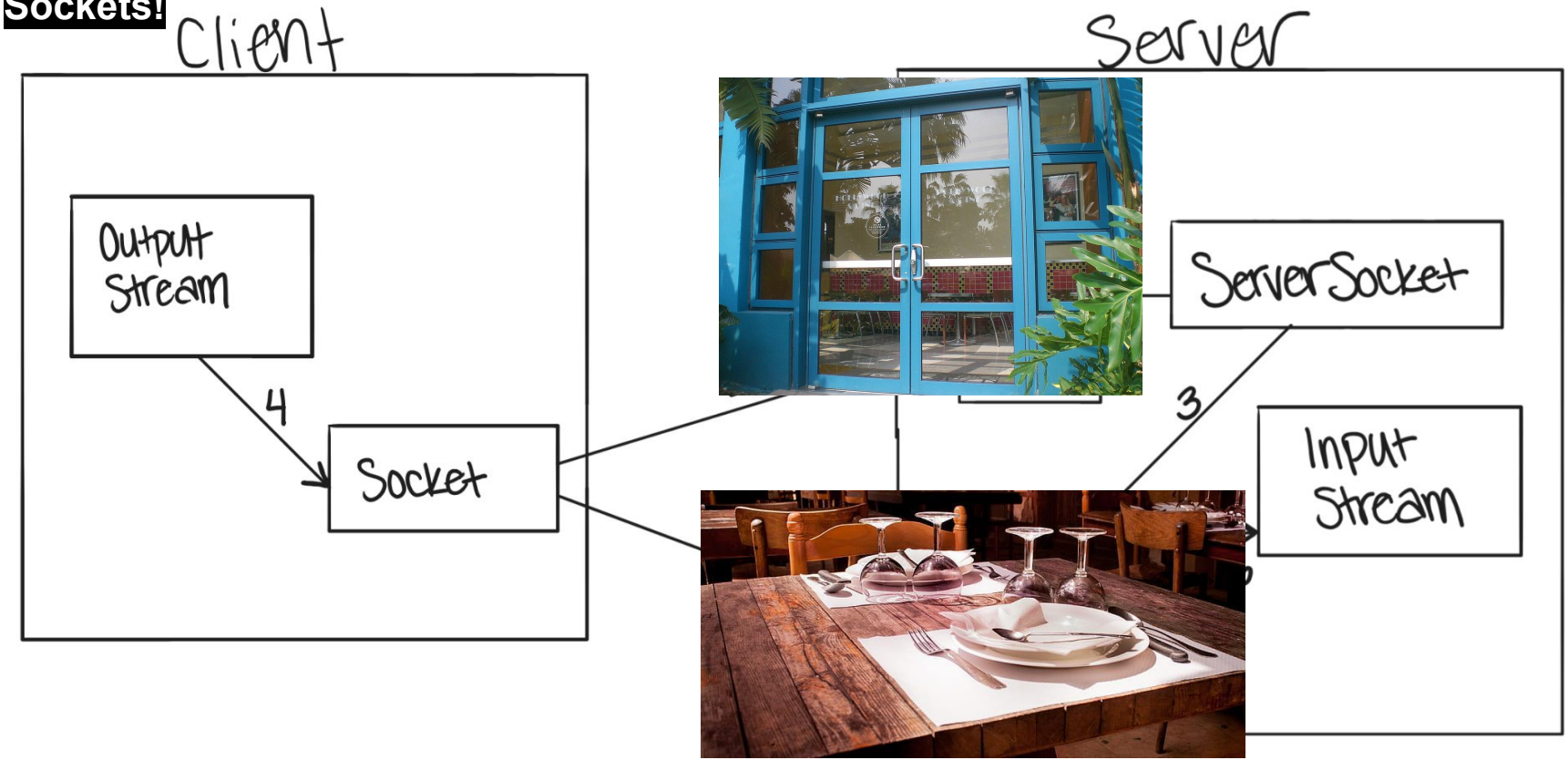
SER 321

Sockets!



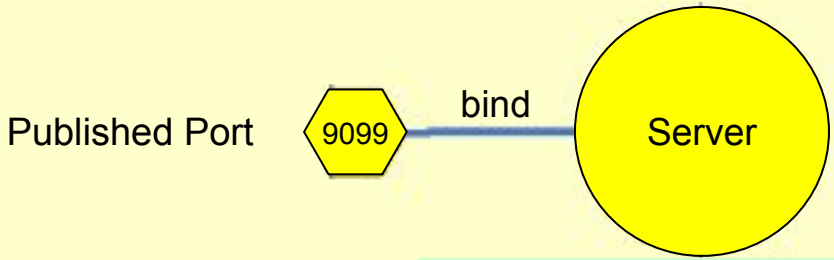
SER 321

Sockets!



SER 321
Sockets!

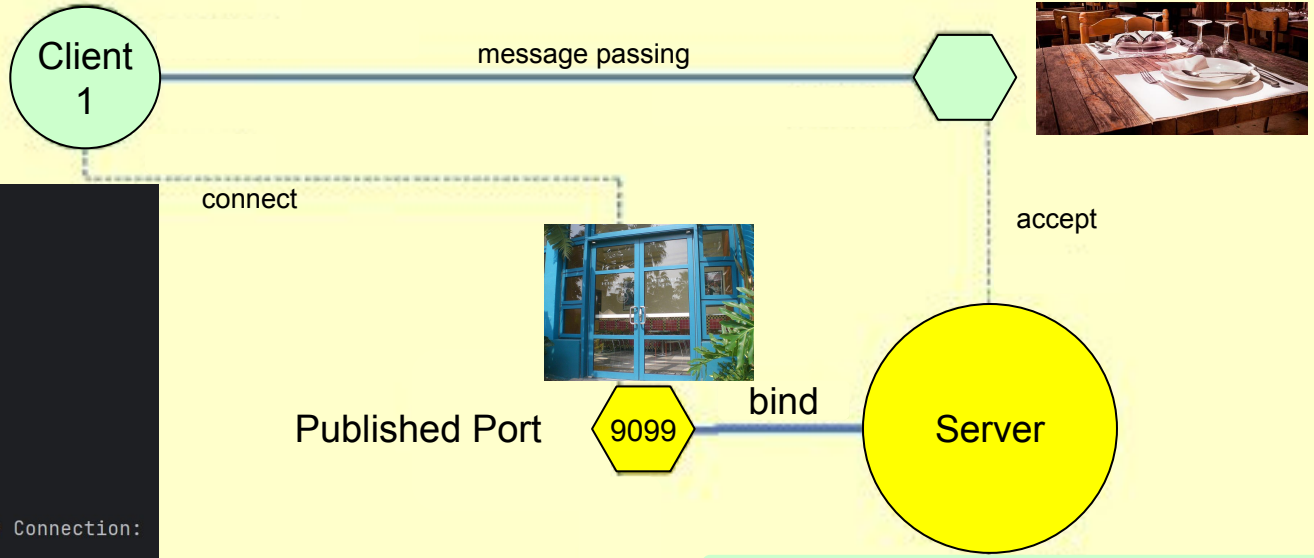
```
> Task :runServer
Server ready for connections
Server is listening on port: 9099
-----
Values of the ServerSocket Object:
Inet Address: 0.0.0.0/0.0.0.0
Local Port: 9099
Server waiting for a connection
Server connected to client
-----
Values of the Client Socket Object after Connection:
        Inet Address: /127.0.0.1
        Local Address: /127.0.0.1
        Local Port: 9099
        Allocated Client Socket (Port): 60296
<=========----> 75% EXECUTING [2m 36s]
> :runServer
```




```
> Task :runClient
Connected to server at localhost:9099
Values of the Socket Object for the Server:
        Host: /127.0.0.1
        Port: 9099
        Local Port: 60296
String to send>
<=========----> 75% EXECUTING [2m 18s]s]
> :runClient
```

Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321
Sockets!




```
> Task :runServer
Server ready for connections
Server is listening on port: 9099
-----
Values of the ServerSocket Object:
Inet Address: 0.0.0.0/0.0.0.0
Local Port: 9099
Server waiting for a connection
Server connected to client
-----
Values of the Client Socket Object after Connection:
        Inet Address: /127.0.0.1
        Local Address: /127.0.0.1
        Local Port: 9099
        Allocated Client Socket (Port): 60296
<=========----> 75% EXECUTING [2m 36s]
> :runServer
```

```
> Task :runClient
Connected to server at localhost:9099
Values of the Socket Object for the Server:
        Host: /127.0.0.1
        Port: 9099
        Local Port: 60296
String to send>
<=========----> 75% EXECUTING [2m 18s]s]
> :runClient
```

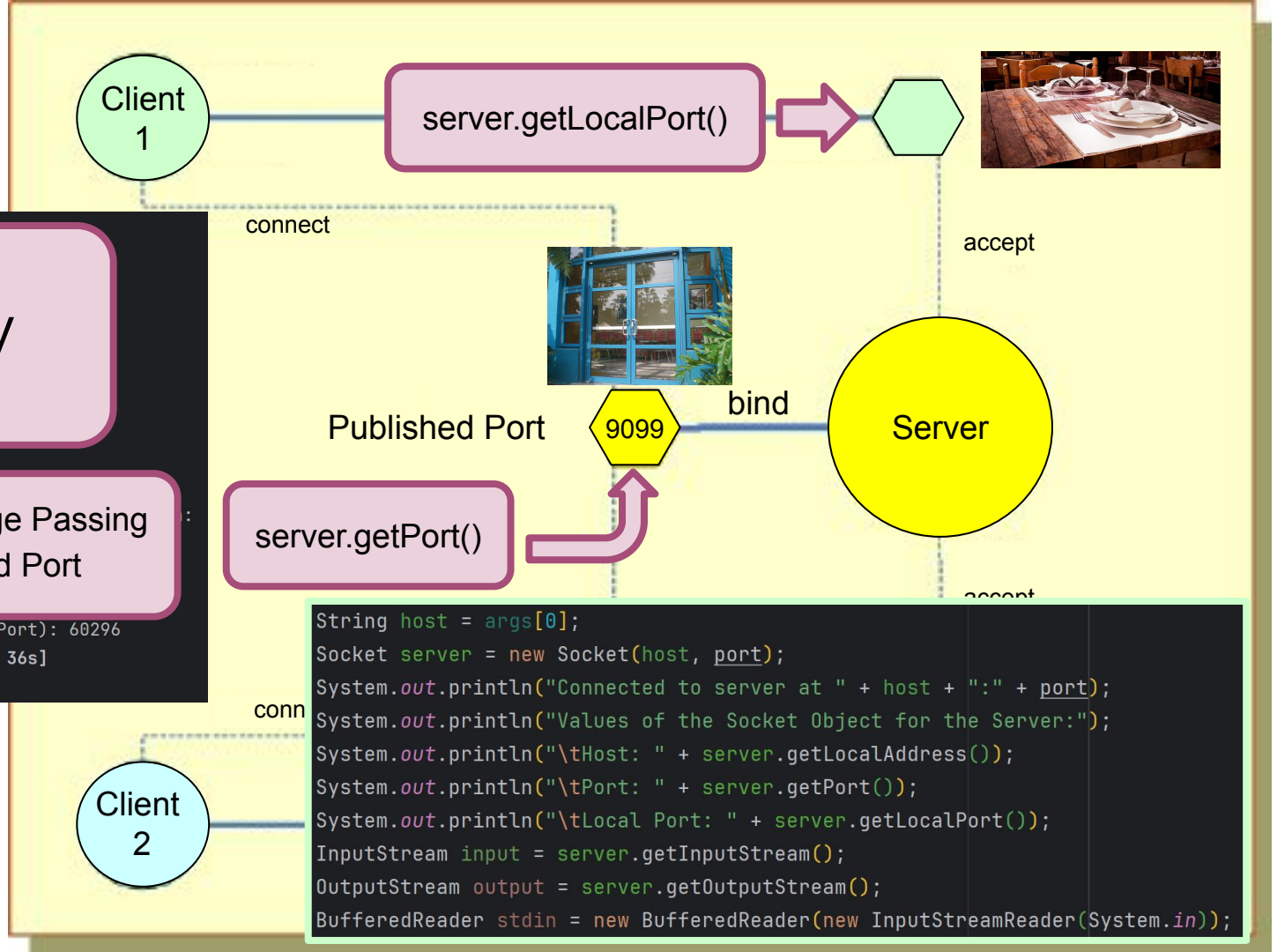
Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321
Sockets!

Client POV

Local Port → Message Passing
Port → Published Port

```
Server connected to client
        Allocated Client Socket (Port): 60296
<=========----> 75% EXECUTING [2m 36s]
> :runServer
```




```
String host = args[0];
Socket server = new Socket(host, port);
System.out.println("Connected to server at " + host + ":" + port);
System.out.println("Values of the Socket Object for the Server:");
System.out.println("\tHost: " + server.getLocalAddress());
System.out.println("\tPort: " + server.getPort());
System.out.println("\tLocal Port: " + server.getLocalPort());
InputStream input = server.getInputStream();
OutputStream output = server.getOutputStream();
BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
```

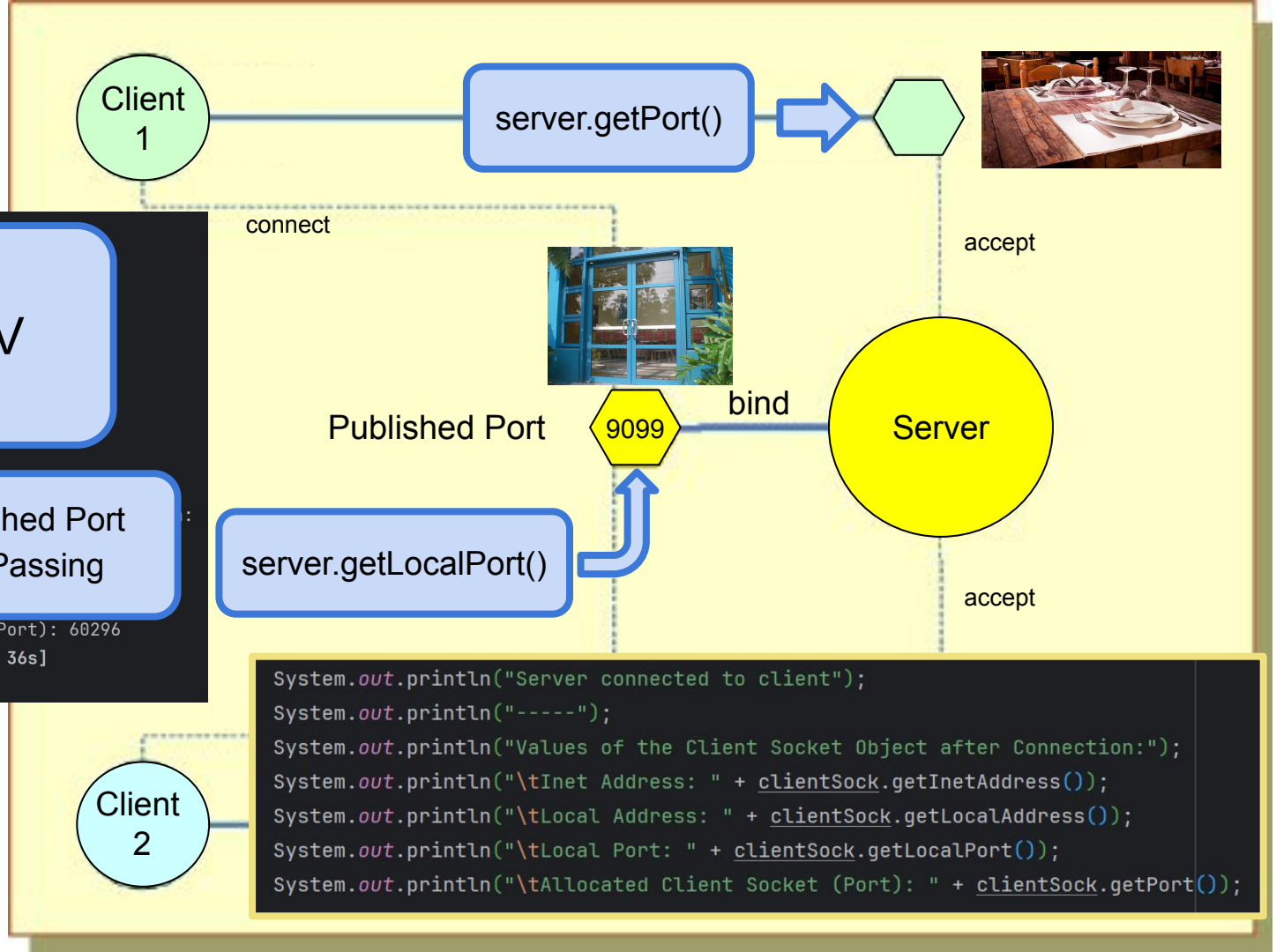
Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321
Sockets!

Server POV

Local Port → Published Port
Port → Message Passing

```
Server connected to client
Allocated Client Socket (Port): 60296
<=========----> 75% EXECUTING [2m 36s]
> :runServer
```




```
System.out.println("Server connected to client");
System.out.println("-----");
System.out.println("Values of the Client Socket Object after Connection:");
System.out.println("\tInet Address: " + clientSock.getInetAddress());
System.out.println("\tLocal Address: " + clientSock.getLocalAddress());
System.out.println("\tLocal Port: " + clientSock.getLocalPort());
System.out.println("\tAllocated Client Socket (Port): " + clientSock.getPort());
```

Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321
Connections

Connections!

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Tuesday, February 4th at 11:00 am MST
- Thursday, February 6th at 7:00 pm MST
- Sunday, February 9th at 7:00 pm MST

Review Sessions:

- Tuesday, February 25th at 11:00 am MST - **Q&A Session**
- Thursday, February 27th at 7:00 pm MST - **Exam Review Session (2hrs)**

Questions?

Survey:

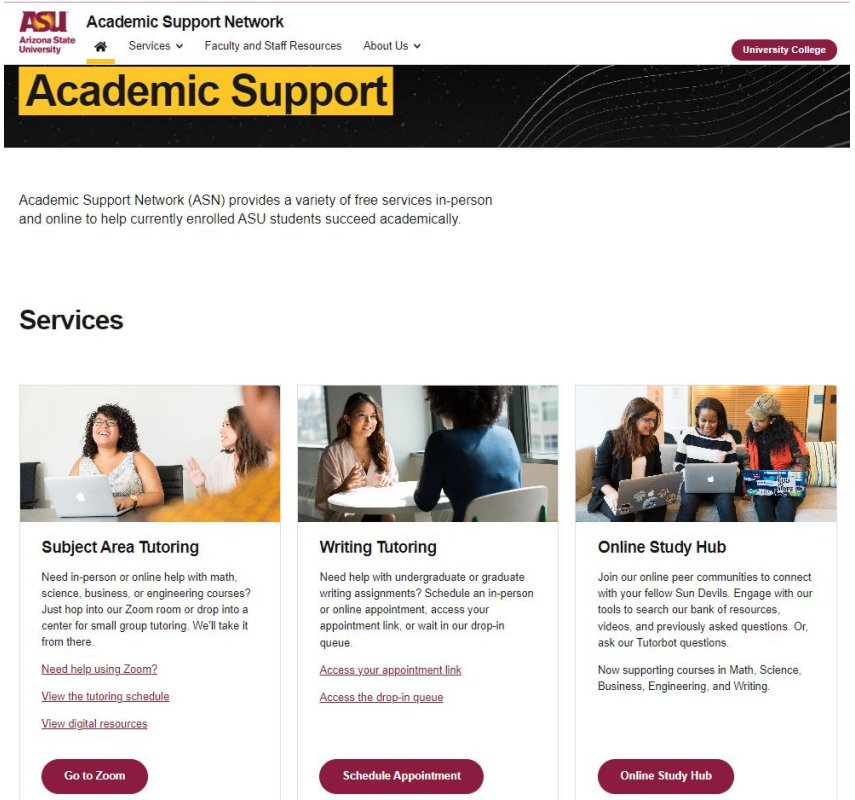
https://asuasn.info/ASNSurvey

52

More Questions?

Check out our other resources!

tutoring.asu.edu



1 – Go to Zoom

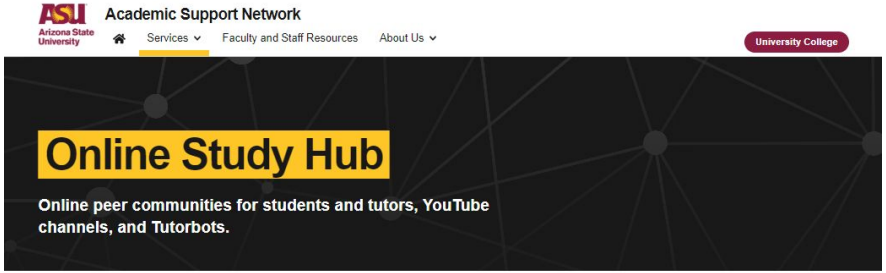
2- Need help using Zoom?
View the tutoring schedule
View digital resources

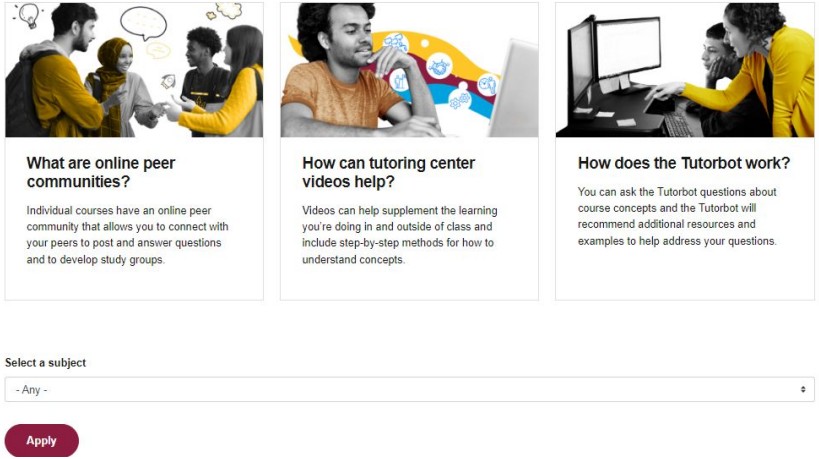
1. **Click on 'Go to Zoom' to log onto our Online Tutoring Center.**
2. **Click on 'View the tutoring schedule' to see when tutors are available for specific courses.**

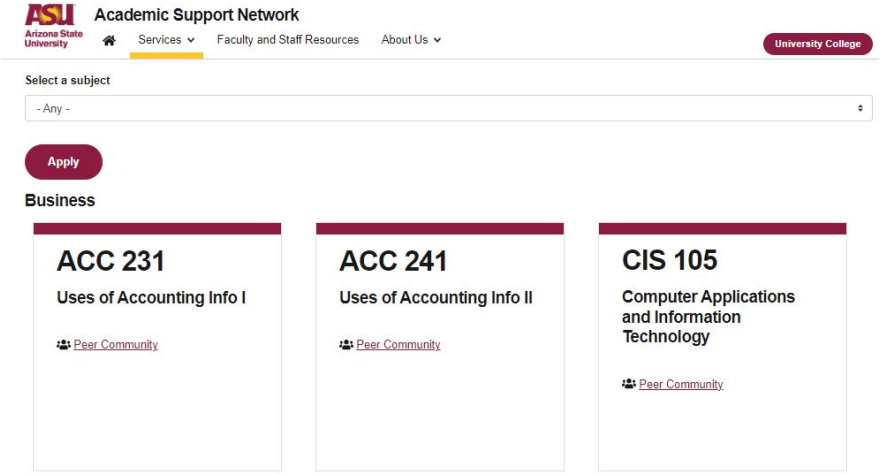
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



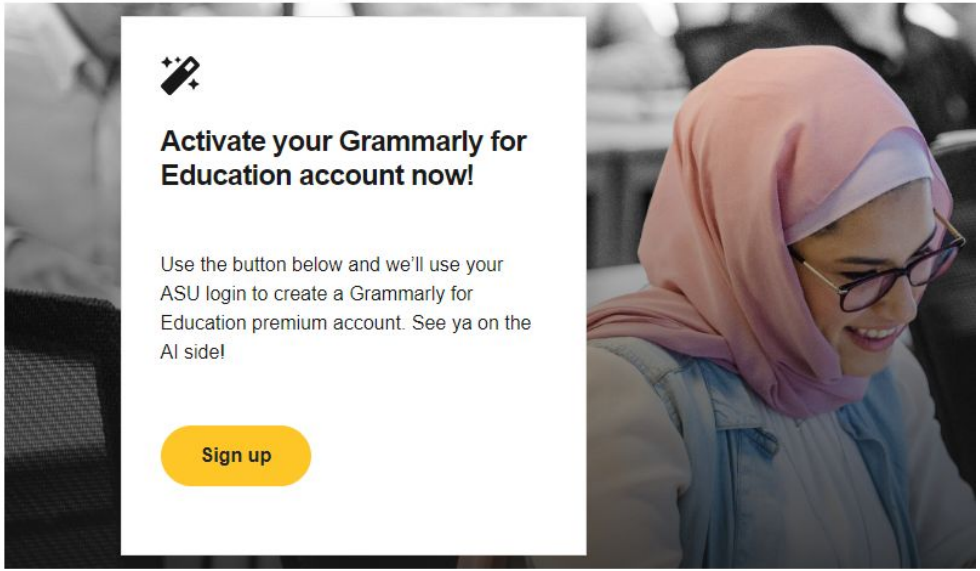




Don't forget to check out the Online Study Hub for additional resources!

Expanded Writing Support Available

Including Grammarly for Education, at no cost!



tutoring.asu.edu/expanded-writing-support

*Available slots for this pilot are limited

Additional Resources

- Course Repo
- Gradle Documentation
- GitHub SSH Help
- Linux Man Pages
- OSI Interactive
- MDN HTTP Docs
  - Requests
  - Responses
- JSON Guide
- org.json Docs
- javax.swing package API
- Swing Tutorials