

# SER 321 B Session

**SI Session**

**Tuesday, April 8th 2025**

*10:00 am - 11:00 am MST*

# Agenda



Serialization

JSON Review

Formats, Types, & Streams

Client Port Examination

Threading our System

# 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](https://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

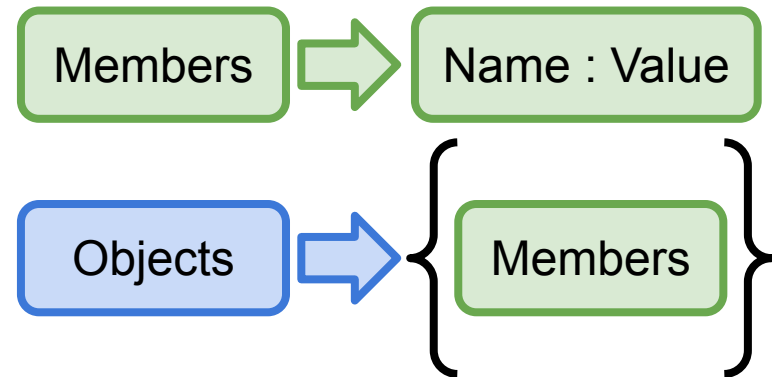
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

# SER 321

## JSON Structure

[org.json Docs](https://www.org.json/docs)

[JSON Guide](#)



# SER 321

## JSON Structure

What is a valid value?

Strings

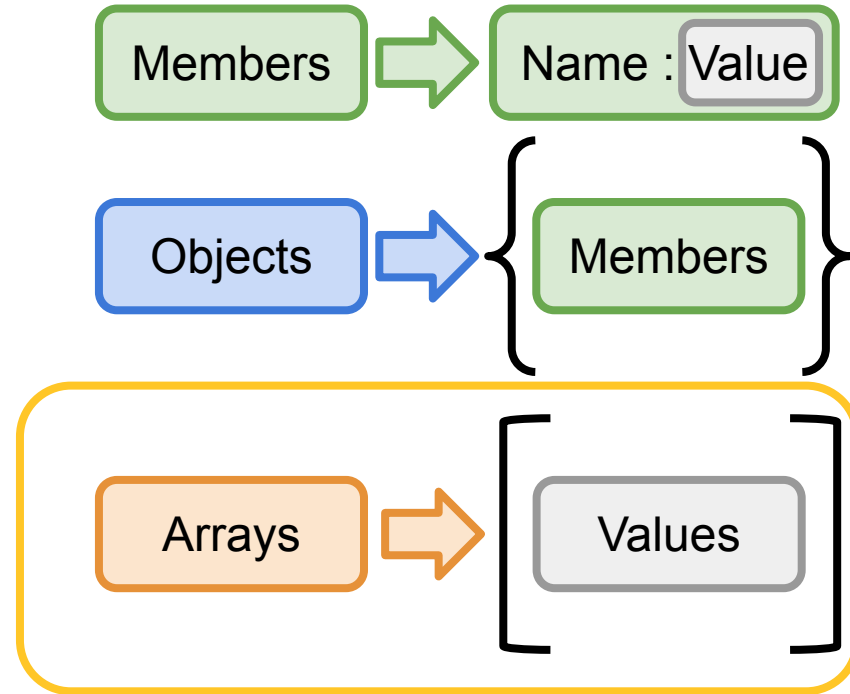
Booleans

Numbers

NULL

Objects

Arrays



# SER 321

## JSON Structure

Which of the following are valid JSON arrays?

1. 

```
{ "hooks": [  
  "192.30.252.0/22",  
  "185.199.108.0/22",  
  ]  
}
```

2. 

```
{ "weather": [  
  { "id": 803,  
    "main": "Clouds",  
    "icon": "04d"  
  }  
]
```

3. 

```
{ "makes" : [  
  true,  
  "Ford",  
  "Honda",  
  "Toyota"  
  ]  
}
```

4. 

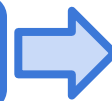
```
{ "readings" : [  
  58.93,  
  76,  
  65.81  
  ]  
}
```

Members



Name : Value

Objects



Members

Arrays



Values

***Check out the recording for  
the discussion and  
solution!***

***Check out the recording for the discussion!***

**SER 321**  
**Serialization**



What is *serialization*?

“Translating data structures or object states for storage or transmission”



***Check out the recording for the discussion!***

What is *serialization*?



Data

“Translating data structures or object states for storage or transmission”

*Check out the recording for the discussion!*

## **SER 321**

### **Serialization**



What is *serialization*?



“Translating data structures or object states for storage or transmission”

**SER 321**

**Serialization**

Can we recall some of the formats?

JSON

Java Object  
Serialization

Protocol Buffers

XML

***Check out the recording for the discussion!***

# SER 321

## Serialization

Binary

Text

Two main  
approaches for  
storing the  
content...

What about the data format?

JSON

Java Object  
Serialization

Protocol Buffers

XML

*Check out the recording for the discussion!*

# SER 321

## Serialization

Binary

Text

Who uses *TEXT*?

Text

JSON

Java Object  
Serialization

Protocol Buffers

Text

XML

*Check out the recording for the discussion!*

**SER 321**  
**Serialization**

Binary

Text

What does  
this imply?

Who uses ***BINARY***?

Text

JSON

Binary

Java Object  
Serialization

Binary

Protocol Buffers

Text

XML

***Check out the recording for the discussion!***

**SER 321**

**Serialization**

Generic  
Superclass

## Streams and their types

```
OutputStream out = sock.getOutputStream();
```

Buffered Stream

Bytes

Data Stream

Primitive DATA Types

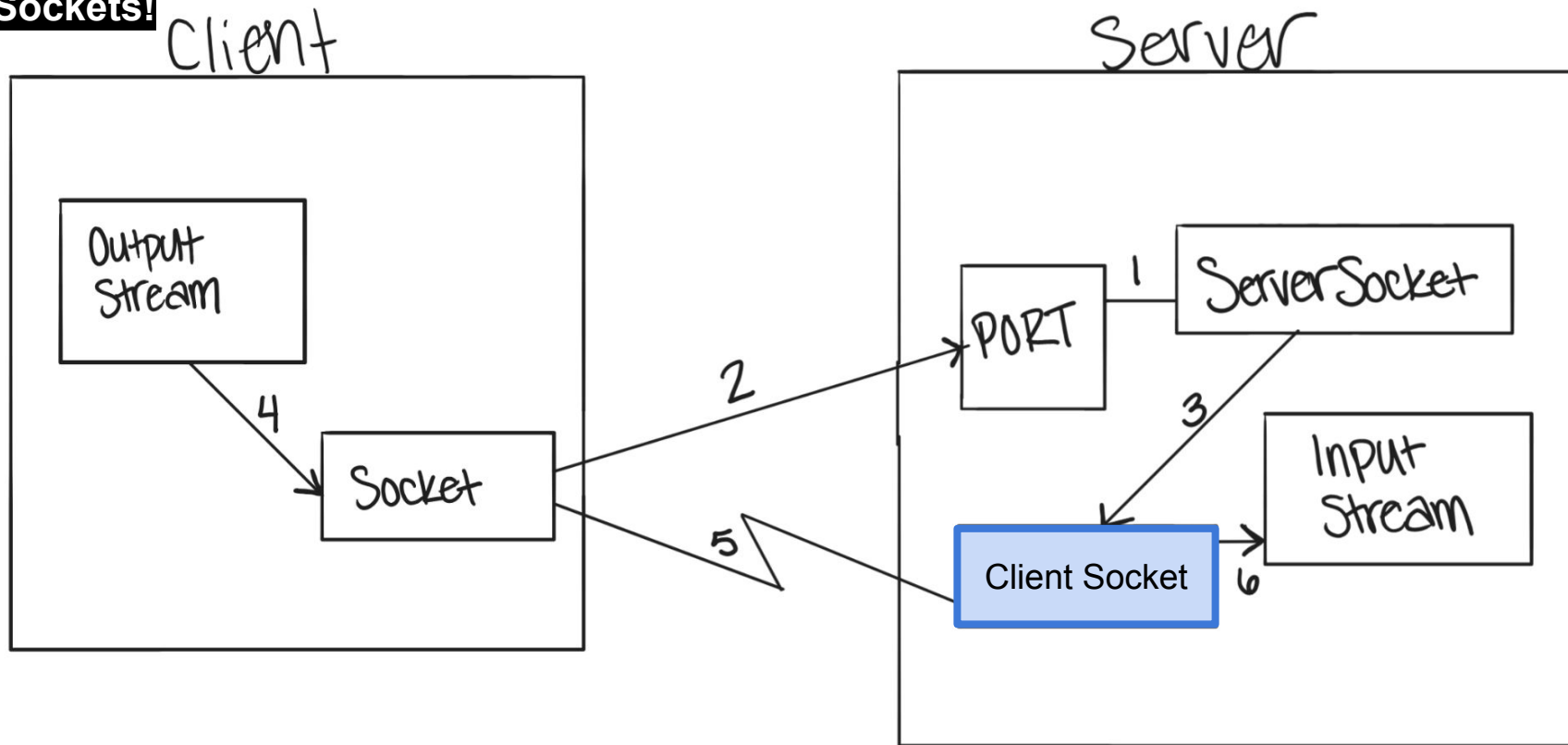
***Check out the recording for the discussion!***

Object Stream

Java Objects

# SER 321

## Sockets!



***Check out the recording for the discussion!***

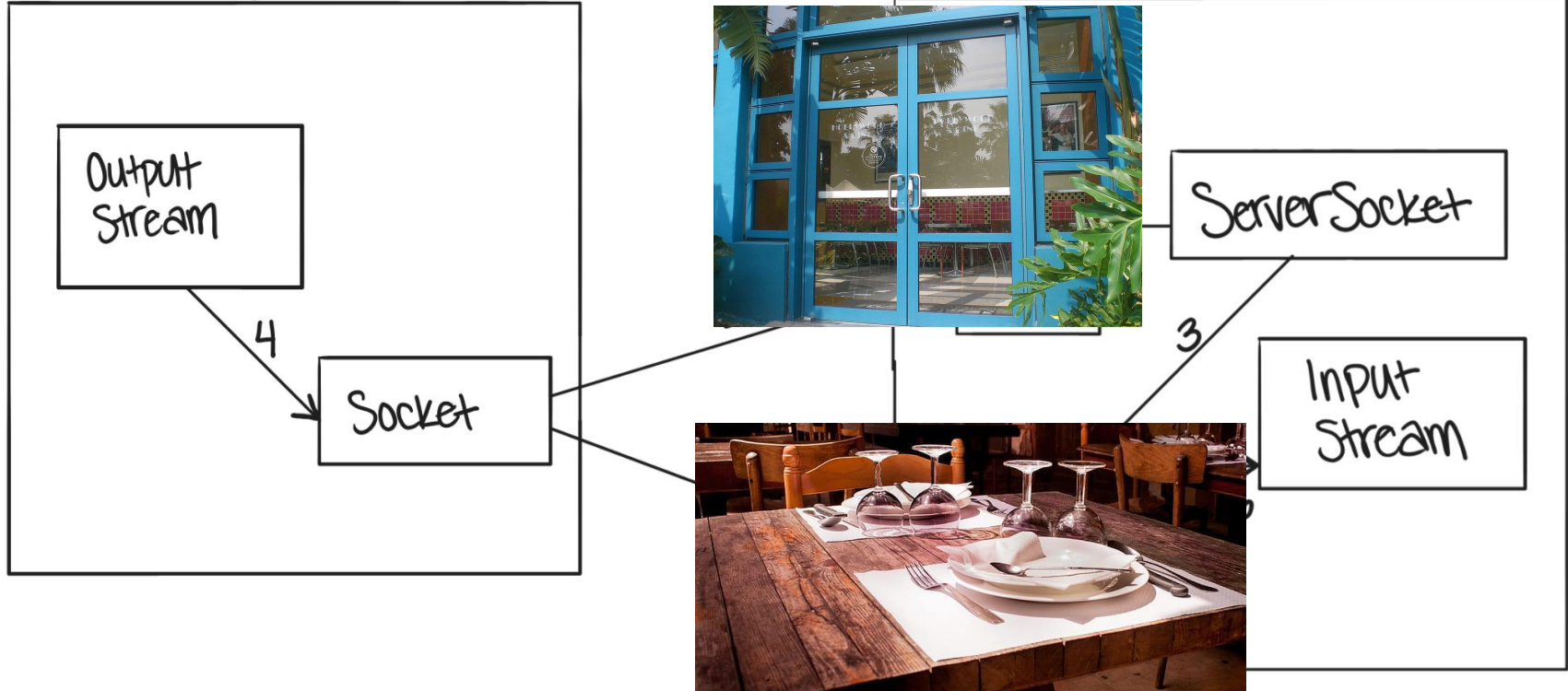


# SER 321

## Sockets!

Client

Server



***Check out the recording for the discussion!***

# SER 321

## Port Examination

Let's see this in action using  
Echo\_Java

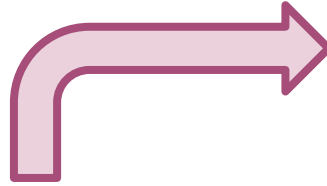
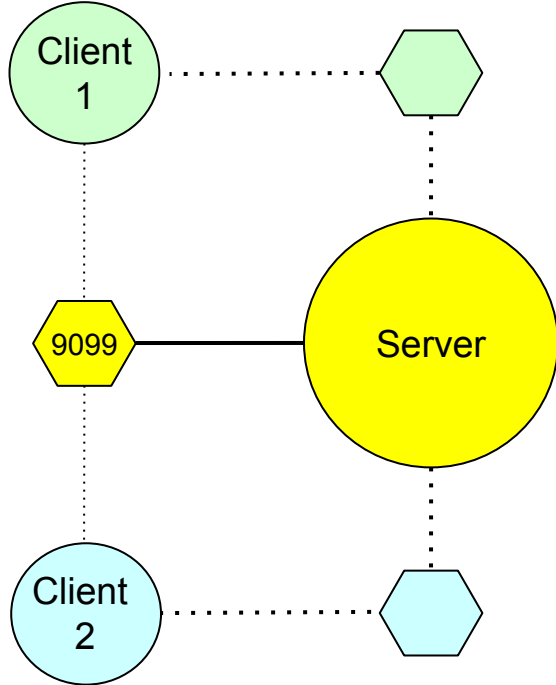
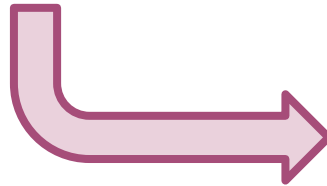


Table  
Please!



***Check out the recording for the discussion!***

# SER 321

## Sockets!

Original

```
String host = args[0];
Socket server = new Socket(host, port);
System.out.println("Connected to server at " + host + ":" + port);
InputStream input = server.getInputStream();
OutputStream output = server.getOutputStream();
BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
```

Client

```
try {
    if (args.length != 1) {
        System.out.println("Usage: gradle runServer -Pport=9099");
        System.exit(status: 0);
    }
    int port = -1;
    try {
        port = Integer.parseInt(args[0]);
    } catch (NumberFormatException nfe) {
        System.out.println("[Port] must be an integer");
        System.exit(status: 2);
    }
    Socket clientSock;
    ServerSocket sock = new ServerSocket(port);
    System.out.println("Server ready for connections");

    int bufLen = 1024;
    byte clientInput[] = new byte[bufLen]; // up to 1024 bytes in a message.
    while(true) {
        System.out.println("Server waiting for a connection");
        clientSock = sock.accept(); // blocking wait
        PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
        InputStream input = clientSock.getInputStream();
        System.out.println("Server connected to client");
        int numr = input.read(clientInput, off: 0, bufLen);
        while (numr != -1) {
            String received = new String(clientInput, offset: 0, numr);
            System.out.println("read from client: " + received);
            out.println(received);
            numr = input.read(clientInput, off: 0, bufLen);
        }
    }
}
```

Sockets/Echo Java

***Check out the  
recording for  
the discussion!***

# SER 321

## Sockets!

### Modification

## Sockets/Echo Java

**Check out the recording for the discussion!**

```
try {
    if (args.length != 1) {...}
    int port = -1;
    try {
        port = Integer.parseInt(args[0]);
    } catch (NumberFormatException nfe) {
        System.out.println("[Port] must be an integer");
        System.exit(status: 2);
    }

    Socket clientSock;
    ServerSocket sock = new ServerSocket(port);
    System.out.println("Server ready for connections");
    System.out.println("Server is listening on port: " + port);
    System.out.println("-----");
    System.out.println("Values of the ServerSocket Object:");
    System.out.println("Inet Address: " + sock.getInetAddress());
    System.out.println("Local Port: " + sock.getLocalPort());

    int bufLen = 1024;
    byte clientInput[] = new byte[bufLen]; // up to 1024 bytes in a message.
    while(true) {
        System.out.println("Server waiting for a connection");
        clientSock = sock.accept(); // blocking wait

        PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
        InputStream input = clientSock.getInputStream();
        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());

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

### Client

```
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));
```



# 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
<=====-----> 75% EXECUTING [10s]
> :runServer
```

Server

```
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));
```

Client

```
try {
    if (args.length != 1) {...}
    int port = -1;
    try {
        port = Integer.parseInt(args[0]);
    } catch (NumberFormatException nfe) {
        System.out.println("[Port] must be an integer");
        System.exit(status: 2);
    }

    Socket clientSock;
    ServerSocket sock = new ServerSocket(port);
    System.out.println("Server ready for connections");
    System.out.println("Server is listening on port: " + port);
    System.out.println("-----");
    System.out.println("Values of the ServerSocket Object:");
    System.out.println("Inet Address: " + sock.getInetAddress());
    System.out.println("Local Port: " + sock.getLocalPort());

    int bufLen = 1024;
    byte clientInput[] = new byte[bufLen]; // up to 1024 bytes in a message.
    while(true) {
        System.out.println("Server waiting for a connection");
        clientSock = sock.accept(); // blocking wait

        PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
        InputStream input = clientSock.getInputStream();
        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());

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

Sockets/Echo Java

*Check out the recording for the discussion!*

# SER 321

## Sockets!

> Task :runServer **Server**

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

**Server**

-----

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 [1m 13s]

> :runServer

□

## Sockets/Echo Java

```
try {  
    if (args.length != 1) {...}  
    int port = -1;  
    try {  
        } catch
```

> Task :runClient

**Client**

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 [31s]

> :runClient

```
System.out.println("Server waiting for a connection");  
clientSock = sock.accept(); // blocking wait
```

```
PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);  
InputStream input = clientSock.getInputStream();  
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());  
int numr = input.read(clientInput, off: 0, bufLen);
```

# SER 321

## Sockets!

***Check out the  
recording for  
the discussion!***

```
> 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
```

Published Port



bind

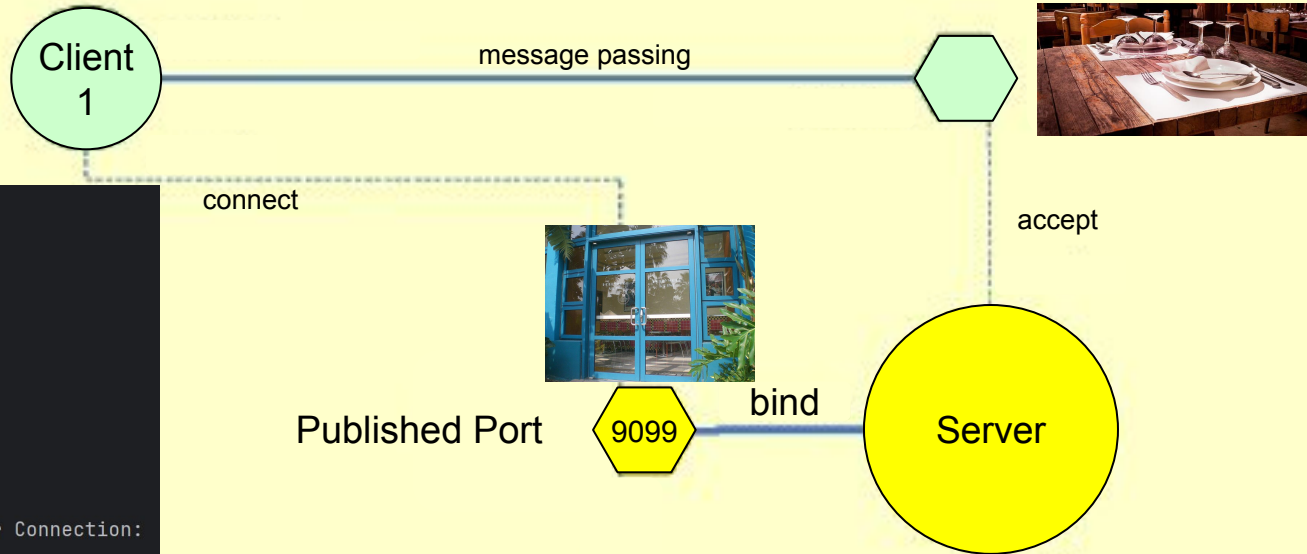
Server

```
> 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
```

# 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
```



**Check out the  
recording for  
the discussion!**

```
> 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
```



# SER 321

## Sockets!

### Client POV

Local Port → Message Passing  
Port → Published Port

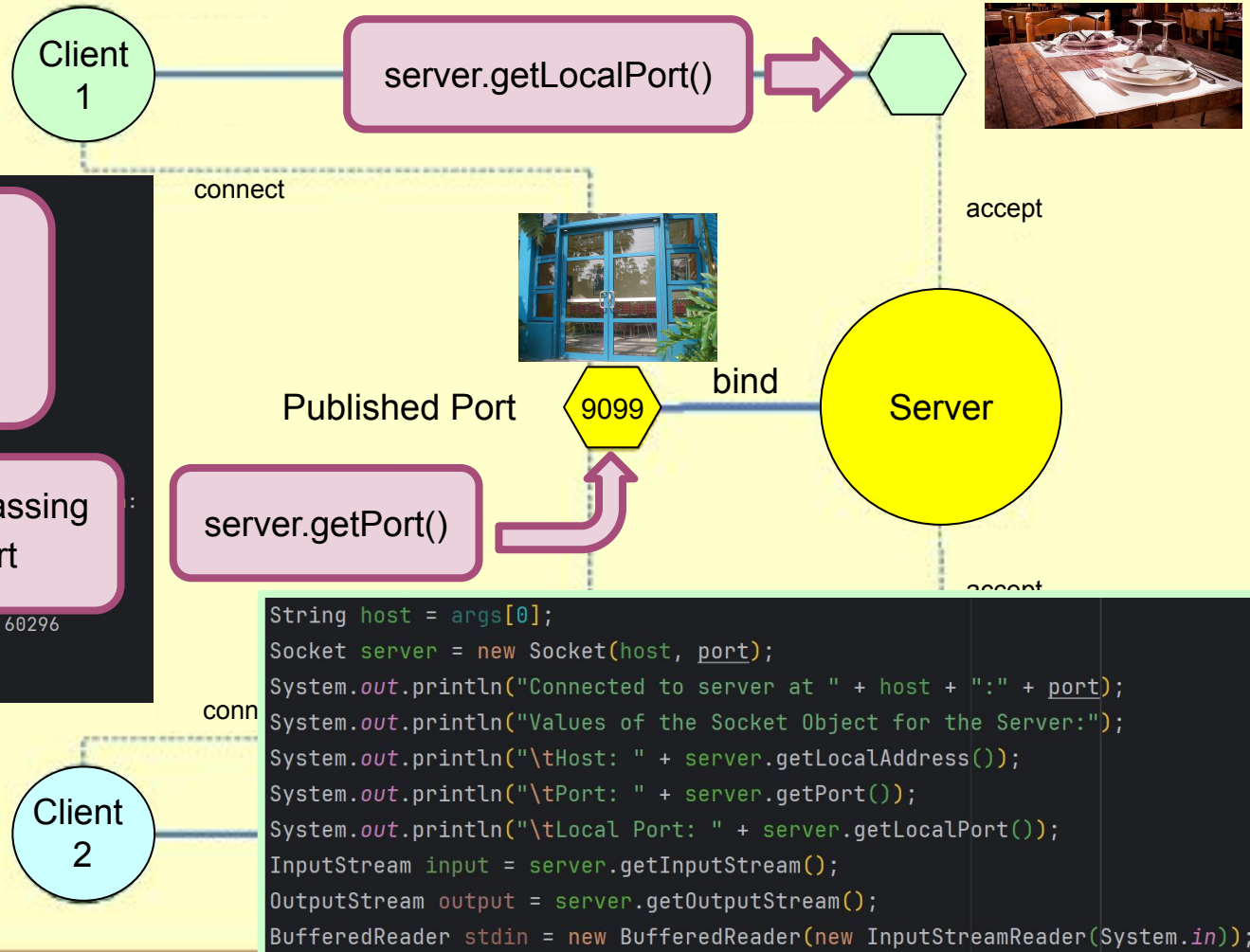
Allocated Client Socket (Port): 60296

<=====--> 75% EXECUTING [2m 36s]

> :runServer

**Check out the recording for the discussion!**

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](https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198)



# SER 321

## Sockets!

### Server POV

Local Port → Published Port  
Port → Message Passing

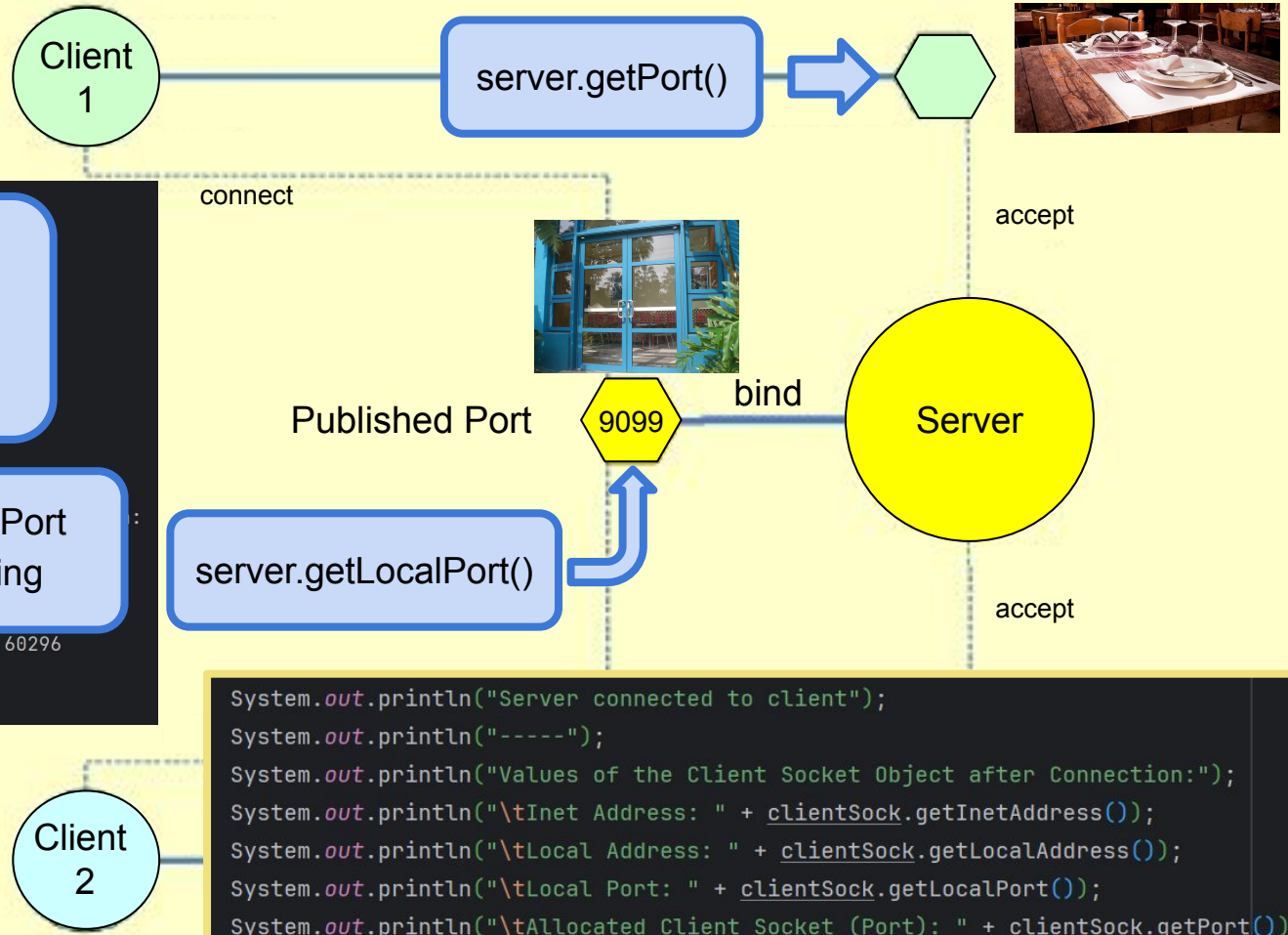
Allocated Client Socket (Port): 60296

<===== > 75% EXECUTING [2m 36s]

> :runServer

**Check out the  
recording for  
the discussion!**

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](https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198)



**SER 321**

**Scratch Space**

## Upcoming Events

### SI Sessions:

- Thursday, April 10th at 7:00 pm MST
- Sunday, April 13th at 7:00 pm MST
- Tuesday, April 15th at 10:00 am MST

### Review Sessions:

- Sunday, April 27th at **6:00 pm MST - 2 hour Exam Review Session**
- Tuesday, April 29th, at 10:00 am MST - **Q&A Session**

# Questions?

## Survey:

<https://asuasn.info/ASNSurvey>



# More Questions?

Check out our other resources!

tutoring.asu.edu



## Academic Support

Academic Support Network (ASN) provides a variety of free services in-person and online to help currently enrolled ASU students succeed academically.

### Services



#### Subject Area Tutoring

Need in-person or online help with math, science, business, or engineering courses? Just hop into our Zoom room or drop into a center for small group tutoring. We'll take it from there.

[Need help using Zoom?](#)

[View the tutoring schedule](#)

[View digital resources](#)

Go to Zoom



#### Writing Tutoring

Need help with undergraduate or graduate writing assignments? Schedule an in-person or online appointment, access your appointment link, or wait in our drop-in queue.

[Access your appointment link](#)

[Access the drop-in queue](#)

Schedule Appointment



#### Online Study Hub

Join our online peer communities to connect with your fellow Sun Devils. Engage with our tools to search our bank of resources, videos, and previously asked questions. Or, ask our Tutorbot questions.

Now supporting courses in Math, Science, Business, Engineering, and Writing.

Online Study Hub

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](https://tutoring.asu.edu/online-study-hub)

 **Academic Support Network**

 [Services](#)  [Faculty and Staff Resources](#) [About Us](#) 

[University College](#)

## Online Study Hub

Online peer communities for students and tutors, YouTube channels, and Tutorbots.



### What are online peer communities?

Individual courses have an online peer community that allows you to connect with your peers to post and answer questions and to develop study groups.



### How can tutoring center videos help?

Videos can help supplement the learning you're doing in and outside of class and include step-by-step methods for how to understand concepts.



### How does the Tutorbot work?

You can ask the Tutorbot questions about course concepts and the Tutorbot will recommend additional resources and examples to help address your questions.

Select a subject

- Any -

[Apply](#)



Academic Support Network



[Services](#) 

[Faculty and Staff Resources](#)

[About Us](#) 

[University College](#)

Select a subject

- Any -

[Apply](#)

Business

### ACC 231

Uses of Accounting Info I

 [Peer Community](#)

### ACC 241

Uses of Accounting Info II

 [Peer Community](#)

### CIS 105

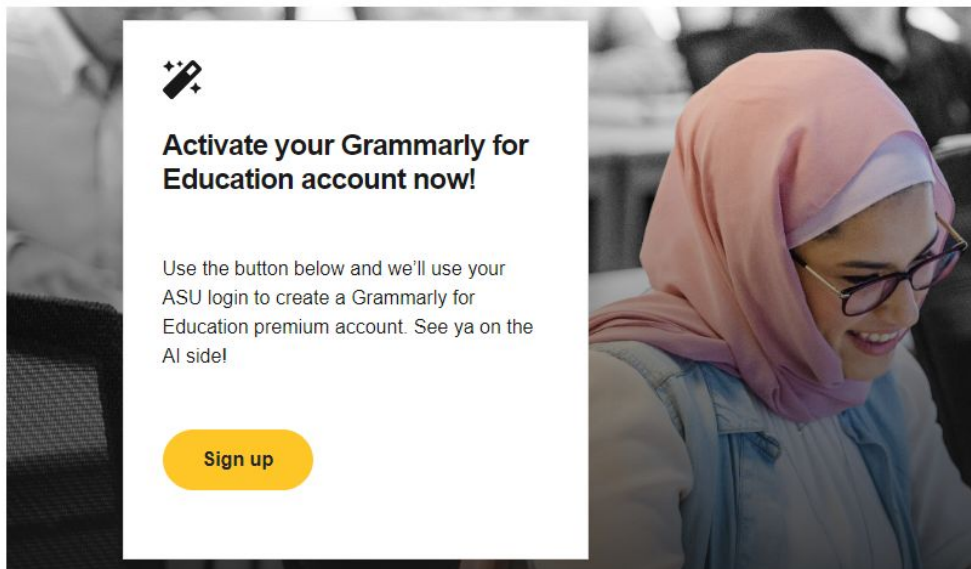
Computer Applications and Information Technology

 [Peer Community](#)

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](https://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](#)
- [Dining Philosophers Interactive](#)
- [Austin G Walters Traffic Comparison](#)