

# SER 321 A Session

**SI Session**

**Tuesday, February 4th 2025**

*11:00 am - 12:00 pm MST*

# Agenda



Sockets Review

Steps & Handling the Client

Port Examination

Serialization

Threading the 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

### Client Socket

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

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

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

### Server Socket

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

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

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

What needs to be done here?

1. Define Params

2. Create Socket

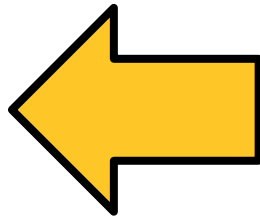
3-5. Mark Socket to Listen

6. Wait for Connection

7. Handle Client Connection

8. Close Client Connection

9. Continue Listening



1

2

3

4

5

## 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

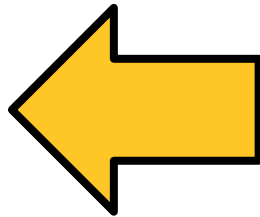
3-5. Mark Socket to Listen

6. Wait for Connection

7. Handle Client Connection

8. Close Client Connection

9. Continue Listening



1

2

3

4

5



## SER 321

### Server Socket

What needs to be done here?

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

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);
```

2. Close Client 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");
```

1

2

3

4

5

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

**SER 321**

**Server Socket**

What needs to be done here?

```
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;
}
```

```
public static JSONObject isValid(String json) {
    try {
        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);
        }

        return res;
    }
}

return new JSONObject();
}
```

# SER 321

## Server Socket

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

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);
```

Just grabbed the input and  
printed to the console

1 Create input/output streams

2 Check for disconnect

3 Check Protocol

4

5

## SER 321

### Server Socket

What needs to be done here?

1. Define Params

2. Create Socket

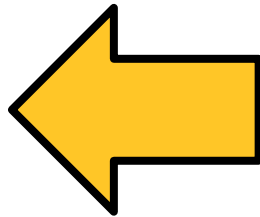
3-5. Mark Socket to Listen

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

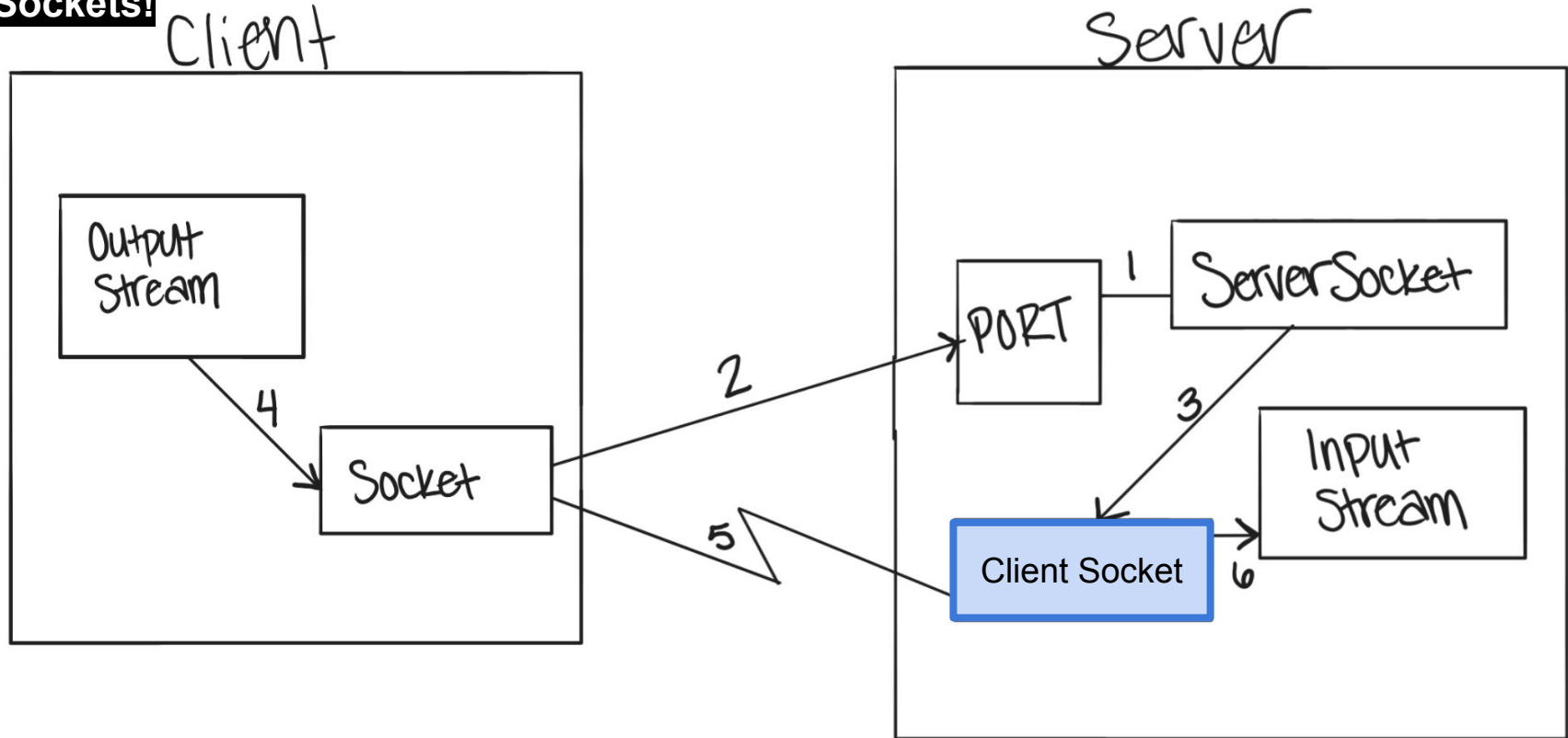
4 Read Headers

5 Handle Accordingly

*Check out the recording for the discussion!*

**SER 321**

**Sockets!**



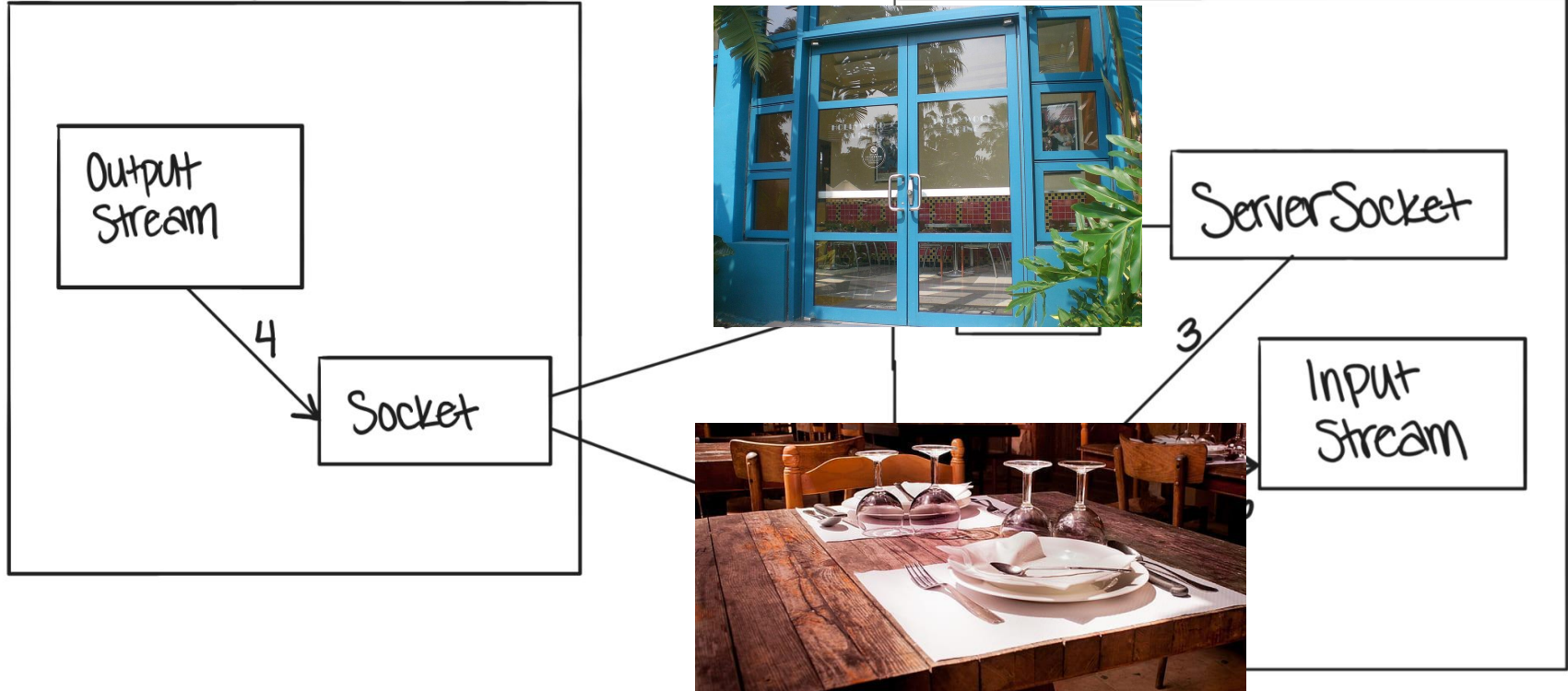
*Check out the recording for the discussion!*

**SER 321**

**Sockets!**

Client

Server

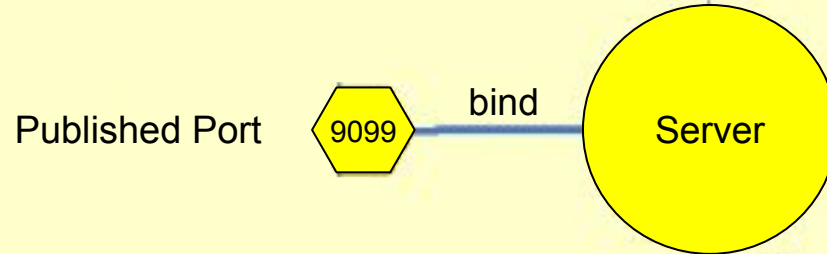


***Check out the recording for the discussion!***

# 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
```



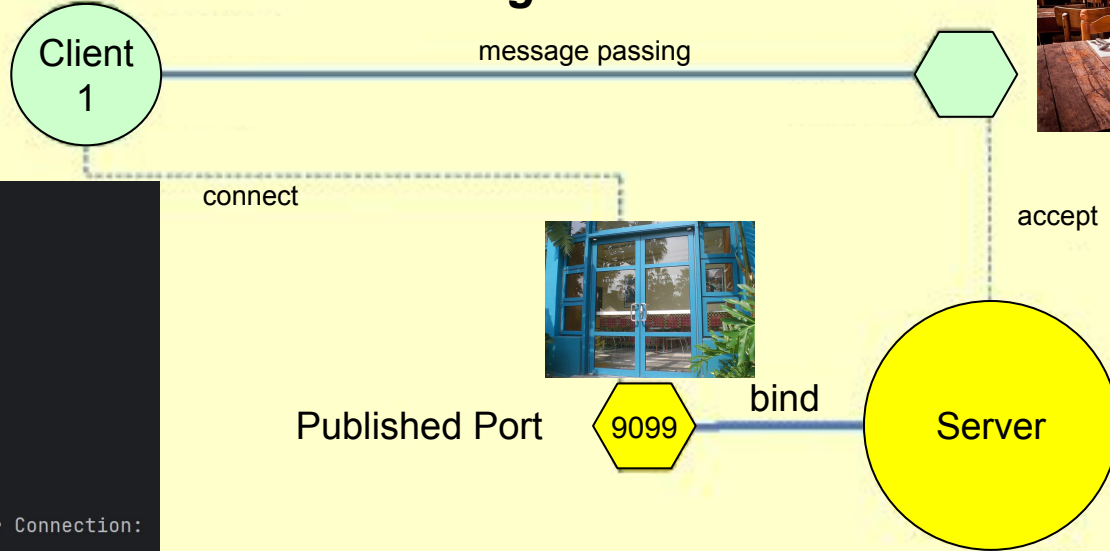
```
> Task :runClient
Connected to server at localhost:9099
Values of the Socket Object for the Server:
    Host: /127.0.0.1
    Port: 9099
    Local Port: 60296
String to send>
<=====--> 75% EXECUTING [2m 18s]s]
> :runClient
```



# 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
```

```
> 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!

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

## Client POV

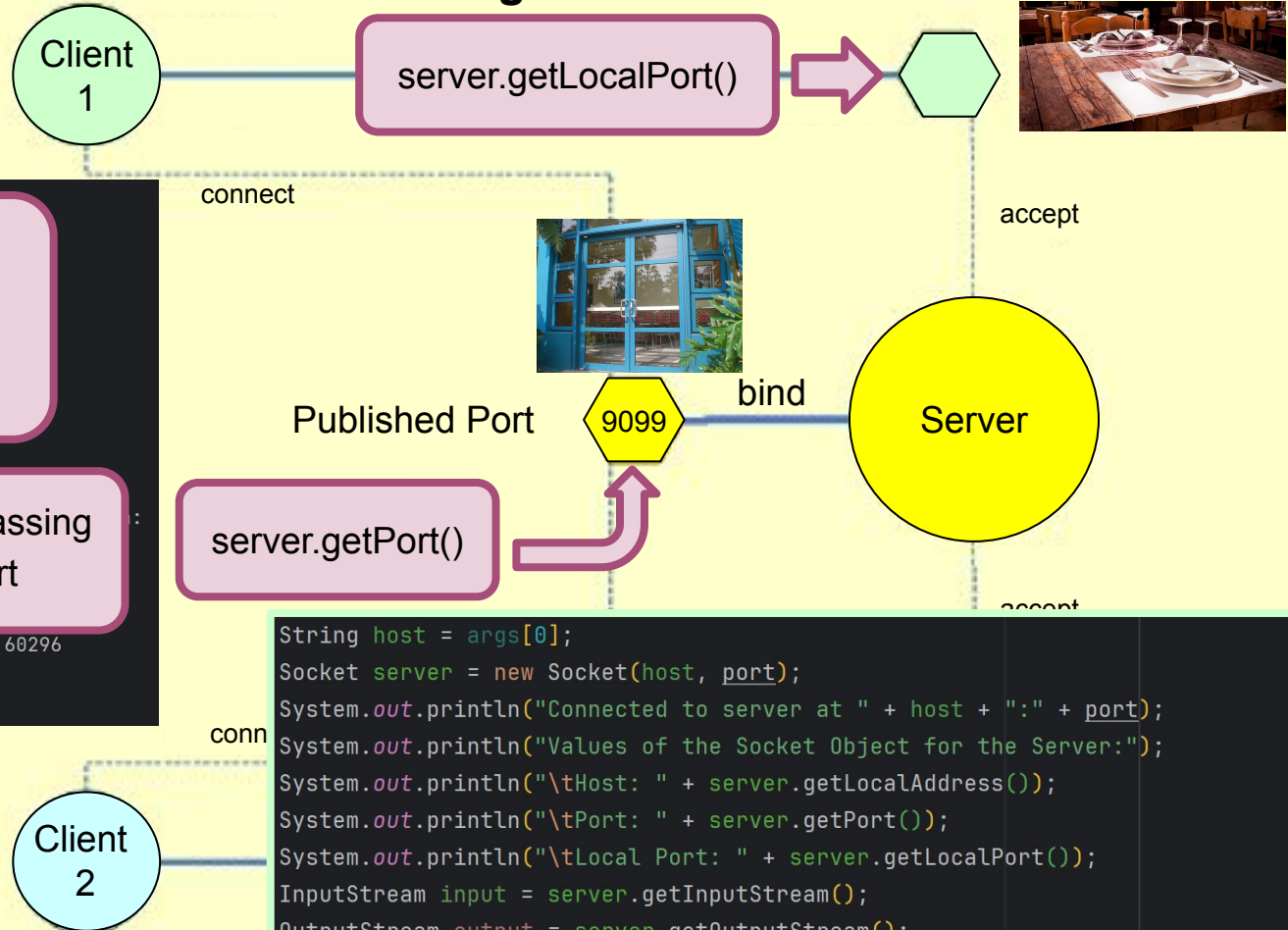
Local Port → Message Passing  
Port → Published Port

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));
```



# SER 321

## Sockets!

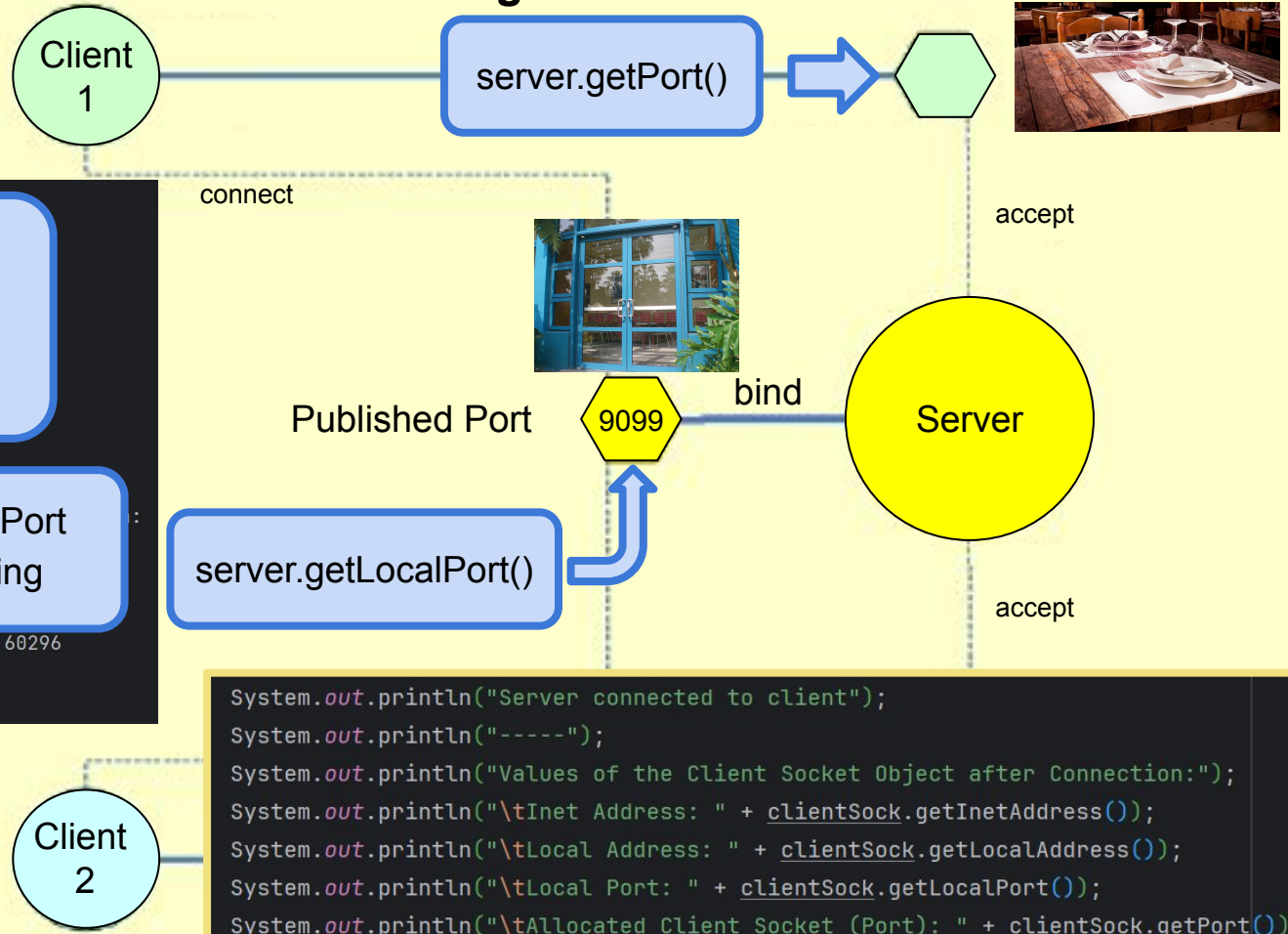
Check out the recording for the discussion!

Server POV

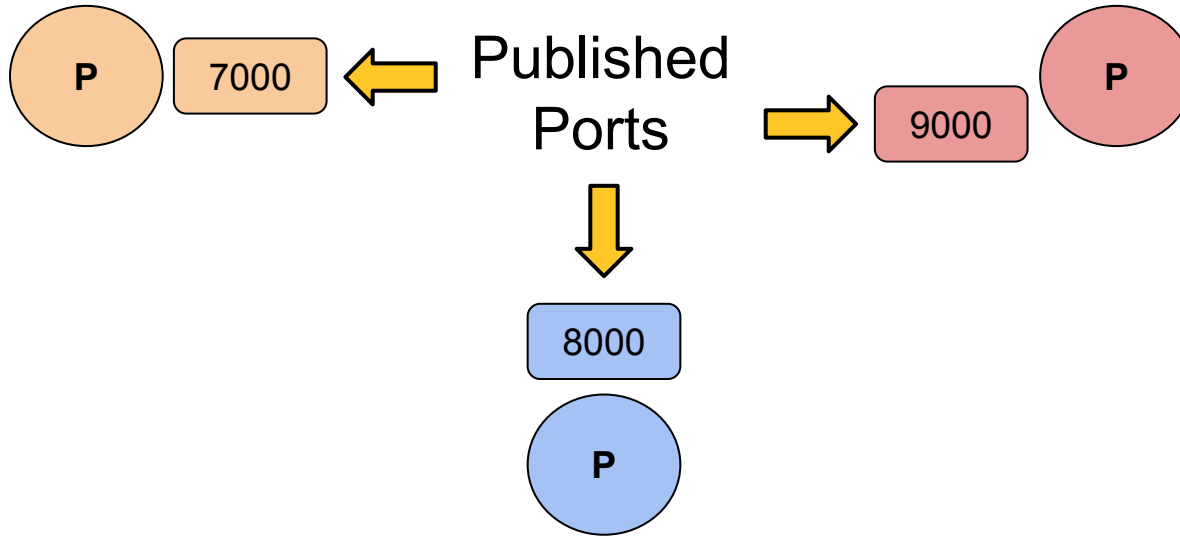
Server connected to client

Local Port → Published Port  
Port → Message Passing

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



Remember that the OS allocates a new port for the client socket!

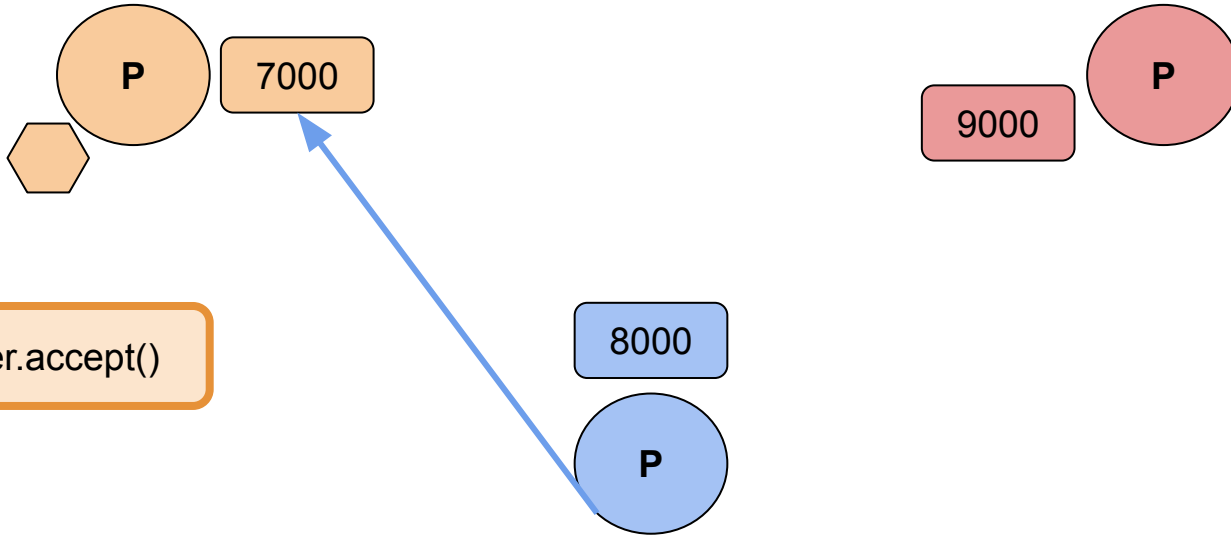


***Check out the recording for the discussion!***

# SER 321

## Communication

Remember that the OS allocates a new port for the client socket!



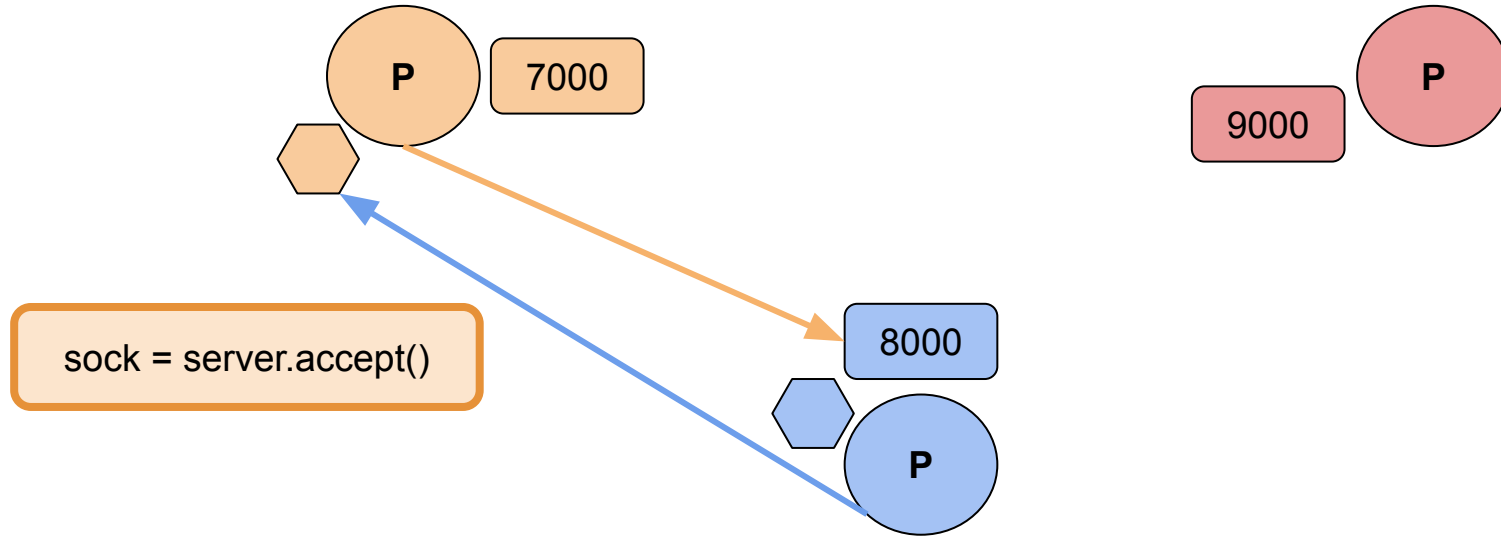
```
sock = server.accept()
```

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

# SER 321

## Communication

Remember that the OS allocates a new port for the client socket!

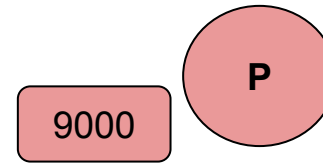
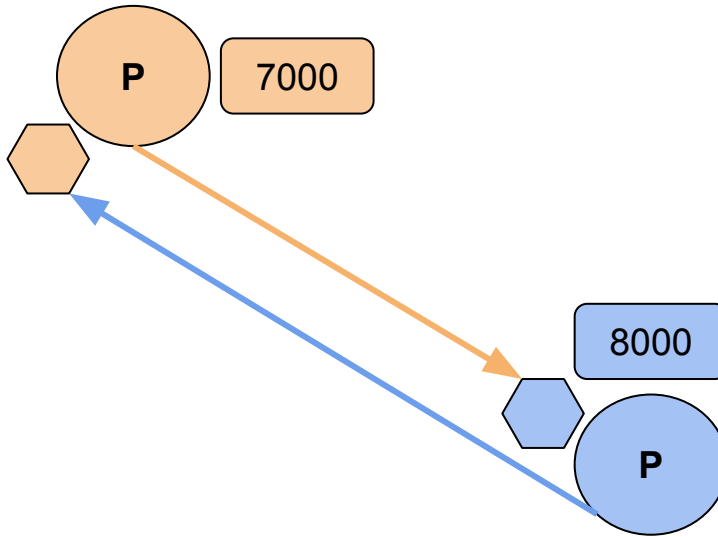


***Check out the recording for the discussion!***

# SER 321

## Communication

Remember that the OS allocates a new port for the client socket!



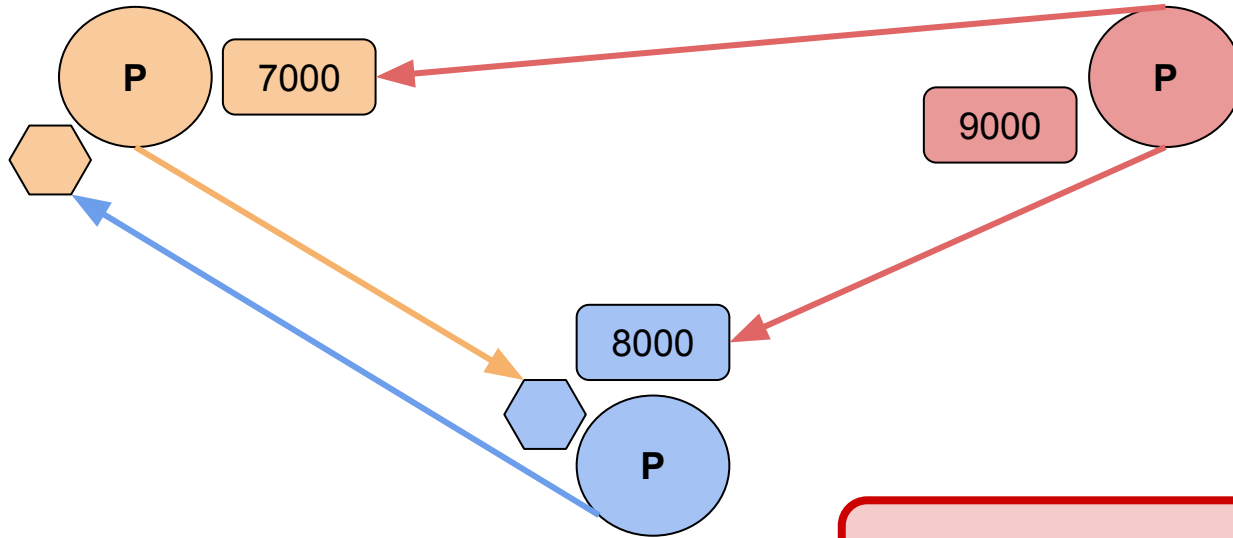
What about Peer 9000?

*Check out the recording for the discussion!*

# SER 321

## Communication

Remember that the OS allocates a new port for the client socket!



*Check out the recording for the discussion!*

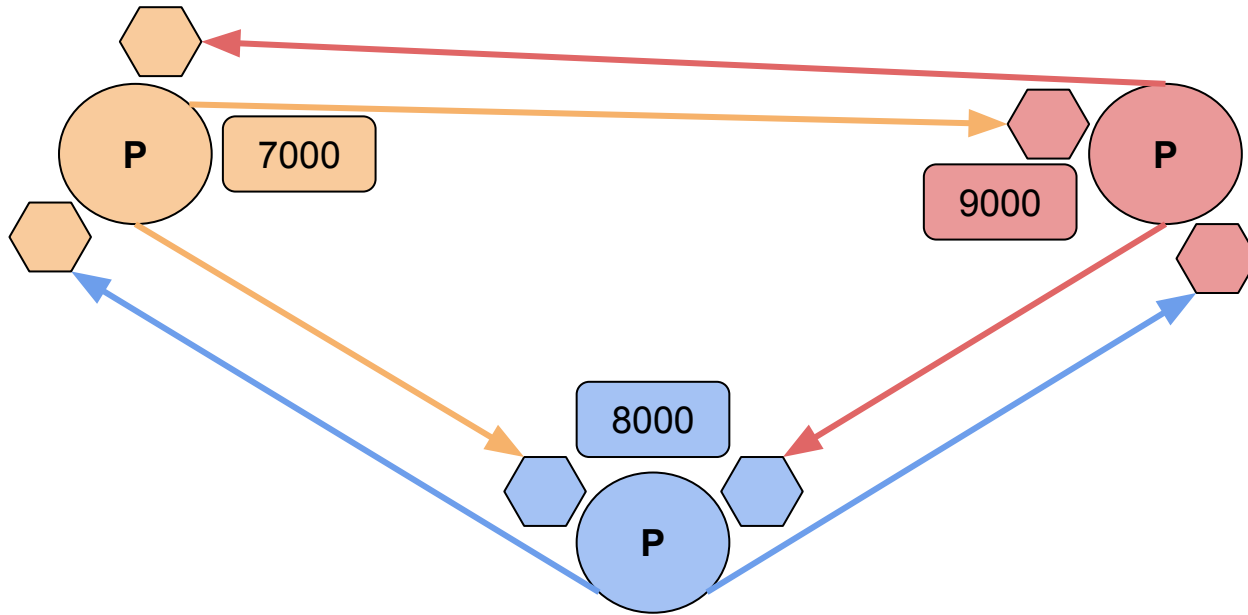
What about Peer 9000?



**SER 321**

**Communication**

Remember that the OS allocates a new port for the client socket!



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

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

**SER 321**

**Serialization**

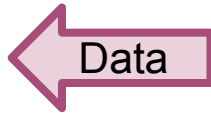


What is *serialization*?

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

# SER 321

## Serialization



??

What is *serialization*?

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

## **SER 321**

### **Serialization**



??

What is *serialization*?

Serialized Data

**SER 321**

**Serialization**

Can we recall some of the formats?

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

Two main  
approaches for  
storing the  
content...

What about the data format?

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

Binary

Text

Who uses ***TEXT***?

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

Binary

Text

What does  
this imply?

Who uses ***BINARY***?

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



**SER 321**

**Serialization**

Generic  
Superclass

# Streams and their types

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

Buffered Stream

Bytes

*Check out the recording for the discussion and memory trick!*

Data Stream

Primitive DATA Types

Object Stream

Java Objects

**SER 321**

**System Layout**

You have two systems...

How can we test our server with multiple clients?

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



?



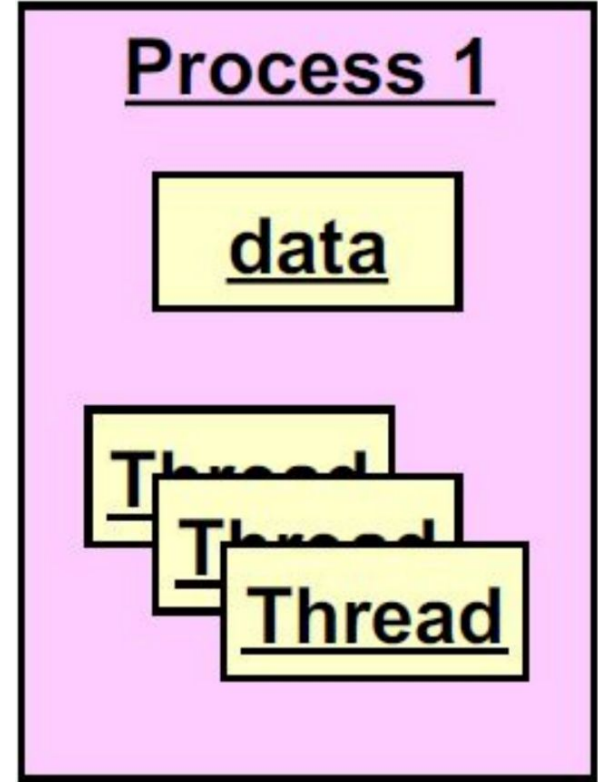
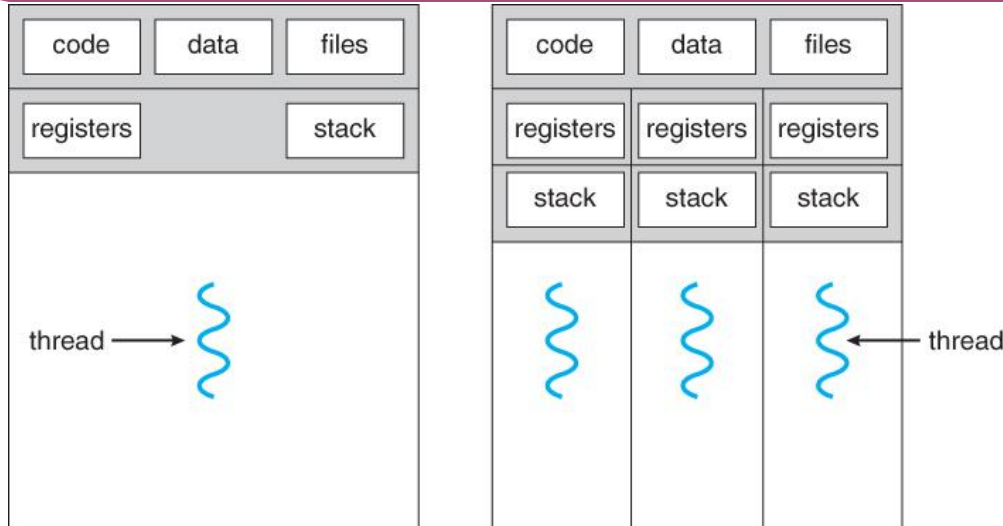
***Check out the recording for the discussion!***

**SER 321**

**Threads**

What does that imply?

Remember that they exist  
*within* the parent process



**SER 321**

**Scratch Space**

## Upcoming Events

### SI Sessions:

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



# 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

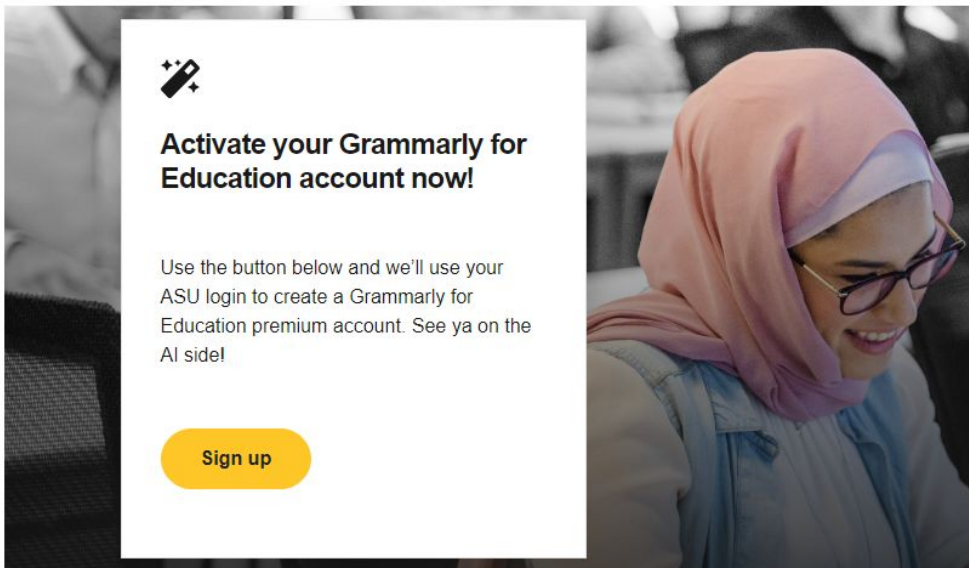
 [Peer Community](#)


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



# Expanded Writing Support Available

Including Grammarly for Education, at no cost!





**Activate your Grammarly for Education account now!**

Use the button below and we'll use your ASU login to create a Grammarly for Education premium account. See ya on the AI side!

[Sign up](#)



[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](#)