SER 321 C Session

SI Session

Thursday, June 6th 2024

6:00 pm - 7:00 pm MST

Agenda

{

Working with JSON

Sockets!

Review Steps

Diagram Connection

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:

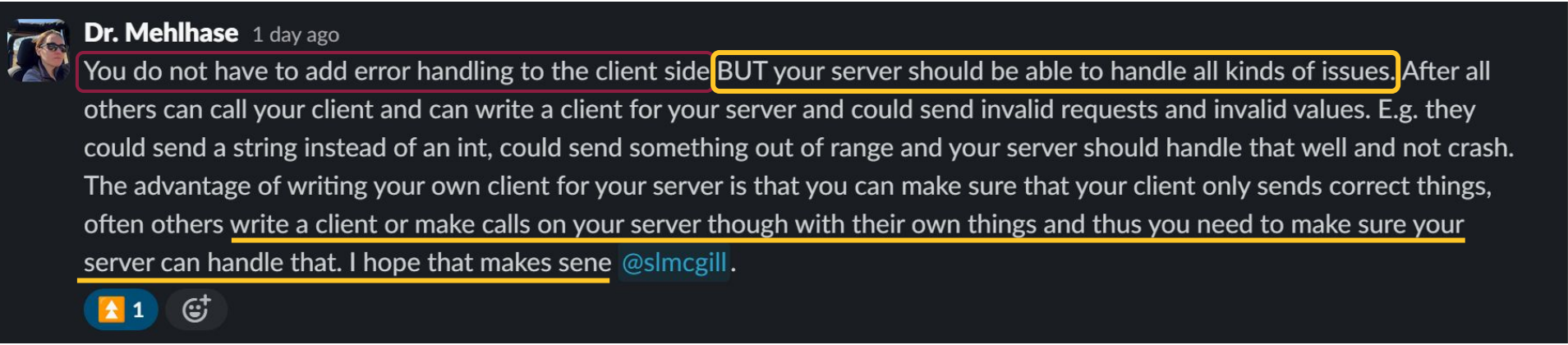
Zoom Features

Zoom Chat

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

SER 321
Error Handling

Excellent question concerning Error Handling for Assign 3



Dr. Mehlhase 1 day ago

You do not have to add error handling to the client side BUT your server should be able to handle all kinds of issues. After all others can call your client and can write a client for your server and could send invalid requests and invalid values. E.g. they could send a string instead of an int, could send something out of range and your server should handle that well and not crash. The advantage of writing your own client for your server is that you can make sure that your client only sends correct things, often others write a client or make calls on your server though with their own things and thus you need to make sure your server can handle that. I hope that makes sene @slmcgill.

1

SER 321

JSON

Which of the following would be a valid response?

Assign 3-1 Starter Code

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

A.
```
{
    “type” : “echo”,
    “echo” : <String>
}
```

B.
```
{
    “type” : “echo”,
    “ok” : false,
    “echo” : <String>
}
```

C.
```
{
    “type” : “echo”,
    “message” : <String>
}
```

D.
```
{
    “type” : “echo”,
    “ok” : true,
    “echo” : <String>
}
```

SER 321

JSON

Which of the following would be a valid response?

Assign 3-1 Starter Code

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

A.
```
{
    “type” : “echo”,
    “echo” : <String>
}
```

B.
```
{
    “type” : “echo”,
    “ok” : false,
    “echo” : <String>
}
```

C.
```
{
    “type” : “echo”,
    “message” : <String>
}
```

D.
```
{
    “type” : “echo”,
    “ok” : true,
    “echo” : <String>
}
```

How would you check if the *type* header exists?

Assign 3-1 Starter Code

SER 321

JSON

```
{
    "type" : "echo", -- type of request
    "data" : <String>  -- String to be echoed
}
```

Echo Request

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

Think of yourself as the server!

You can assume STR is the data sent from the client

```
String reqType = “”; //to hold request type
String content = “”;  // to hold data from client
JSONObject request = new JSONObject(str);
```

```
if (!request.has(“type”)) {
     //error - no type header
     //send error response
}
```

SER 321

JSON

How would you fetch the type content?

Assign 3-1 Starter Code

```
{
    "type" : "echo", -- type of request
    "data" : <String>  -- String to be echoed
}
```

Echo Request

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

Think of yourself as the server!

You can assume STR is the data sent from the client

```
String reqType = “”; //to hold request type
String content = “”;  // to hold data from client
JSONObject request = new JSONObject(str);
if (!request.has(“type”)) {
     //error - no type header
     //send error response
}
```

```
reqType = request.getString(“type”);
```

SER 321
JSON

How would you obtain the message sent?

Assign 3-1 Starter Code

```
{
    "type" : "echo", -- type of request
    "data" : <String>  -- String to be echoed
}
```

Echo Request

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

Think of yourself as the server!

You can assume STR is the data sent from the client

```
String reqType = “”; //to hold request type
String content = “”;  // to hold data from client
JSONObject request = new JSONObject(str);
if (!request.has(“type”)) {
     //error - no type header
     //send error response
}
reqType = request.getString(“type”);
if (reqType.equals(“echo”)) {
     //fetch message


}
```

```
content = request.getString(“data”);
```

SER 321
JSON

How would you construct the response?

Assign 3-1 Starter Code

```
{
    "type" : "echo", -- type of request
    "data" : <String>  -- String to be echoed
}
```

Echo Request

```
{
    "type" : "echo", -- echoes the initial response
    "ok" : <bool>, -- true or false depending on request
    "echo" : <String>,  -- echoed String if ok true
    "message" : <String>,  -- error message if ok false
}
```

Echo General Response

Think of yourself as the server!

```
String content = request.getString("data");
JSONObject res = new JSONObject();
```

```
res.put("type", "echo");
res.put("ok", true);
res.put("echo", content);
```

```
{
    "type" : "echo",
    "ok" : true,
    "echo" : <String>
}
```

Target Response

Then what do you do?

SER 321
Client Socket

Steps for the **Client Socket**

1.
2.
3.
4.
5.
6.
7.
8.

SER 321
Server Socket

Steps for the **Server Socket**

1.
2.
3.
4.
5.
6.
7.
8.
9.

SER 321
Server Socket

Java handles a few steps for us…

1. Define Params
2. Create Socket
3. **C ONLY** Create a struct for the address
3-5. Mark Socket to Listen
5. Mark Socket to Listen for Connections
6. Wait for Connection
7. Handle Client Connection
8. Close Client Connection
9. Continue Listening for Connections

Assign 3-1 Starter Code

SER 321

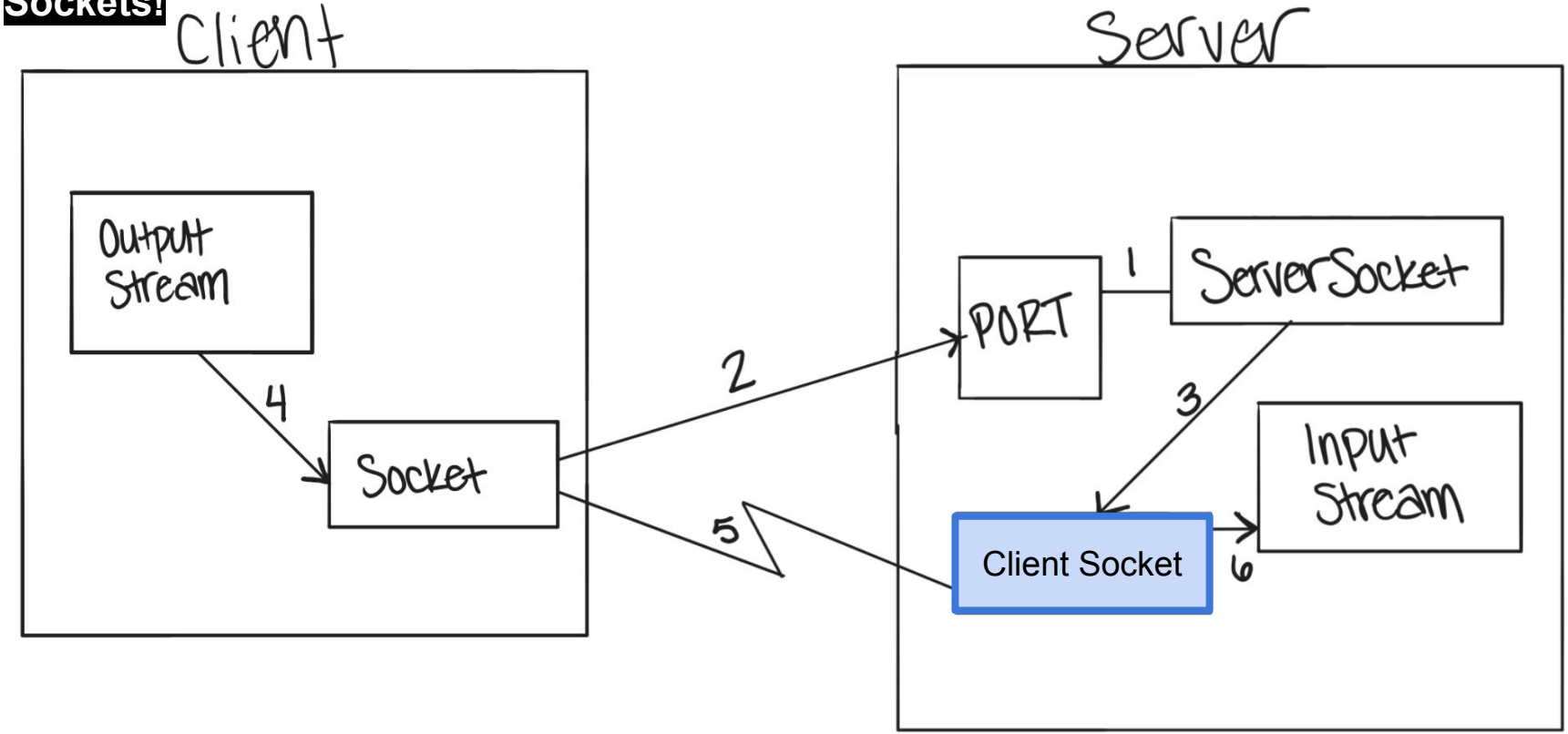
Server Socket

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

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

  if (args.length != 1) {
    System.out.println("Expected arguments: <port(int)>");
    System.exit( status: 1);
  }

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

  try {
    //open socket
2 & 3-5 ServerSocket serv = new ServerSocket(port);
    System.out.println("Server ready for connections");

    /** Simple loop accepting one client and calling handling one request. */

9   while (true){
      System.out.println("Server waiting for a connection");
6     sock = serv.accept(); // blocking wait
      System.out.println("Client connected");
```

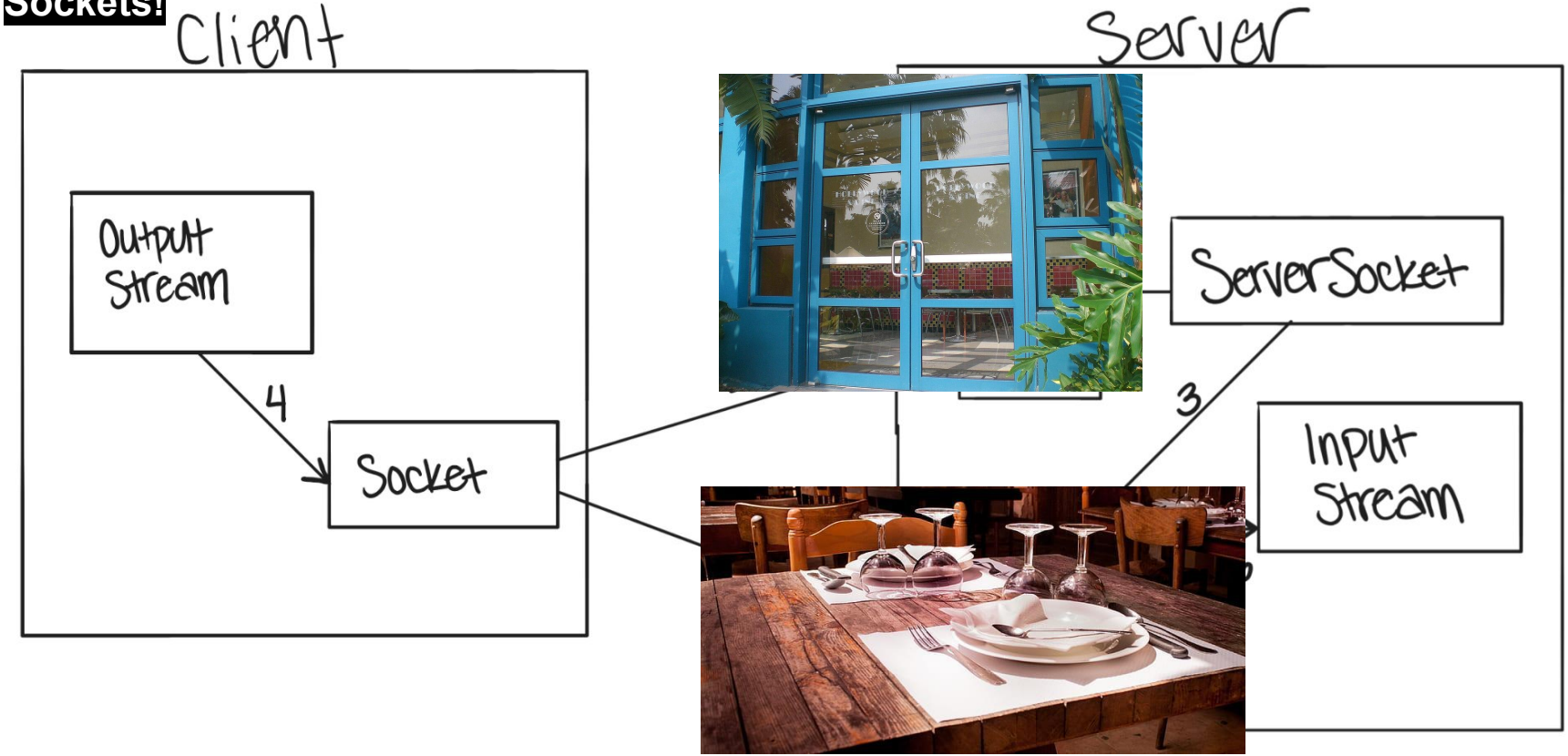
SER 321

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 [1m 13s]
> :runServer
```

Client

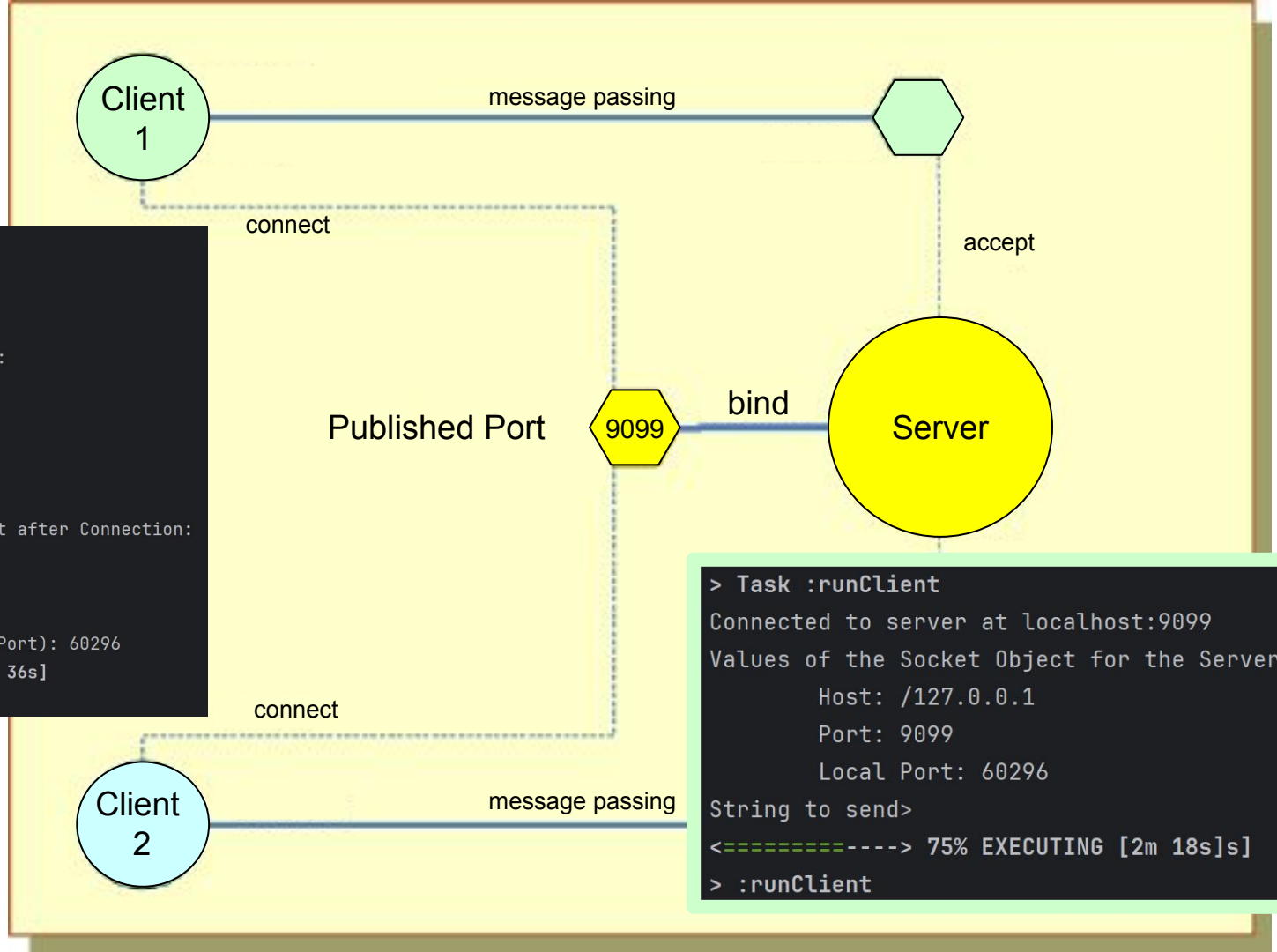
```
String host = args[0];
Socket server = new Socket(host, port);
System.out.println("Connected to server at " + host + ":" + port);
System.out.println("Values of the Socket Object for the Server:");
System.out.println("\tHost: " + server.getLocalAddress());
System.out.println("\tPort: " + server.getPort());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

```
> 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 [31s]
> :runClient
```

```
try {
    if (ar
    int po
    try {

    } catc

    }
    Socket
    ServerSo
    System.
    System.
    System.
    System.
    System.
    System.

    int bufl
    byte cli
     hile(tr
```

```
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!




```
> 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

Scratch Space

Questions?

Survey:

http://bit.ly/ASN2324

21

Upcoming Events

SI Sessions:

- Sunday, June 9th at 6:00 pm MST
- Monday, June 10th at 6:00 pm MST
- Thursday, June 13th at 6:00 pm MST

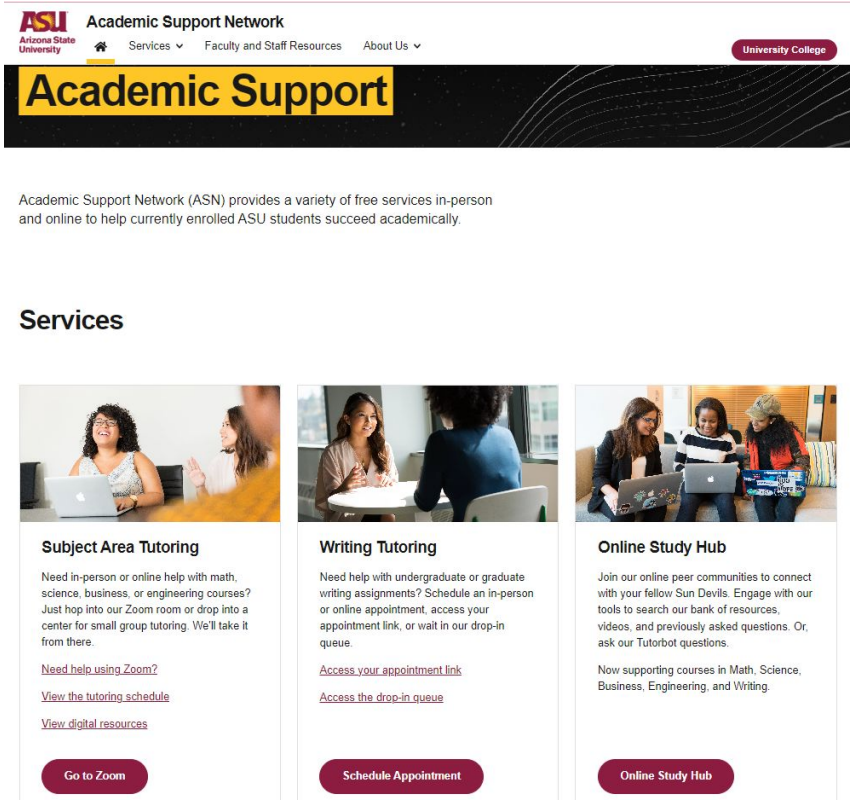
Review Sessions:

- Review Session - **Wednesday**, July 3rd at 6:00 pm MST (2 hr Session)
- Q&A Session - Sunday, July 7th at 6:00 pm MST (Final Session)

More Questions?

Check out our other resources!

tutoring.asu.edu

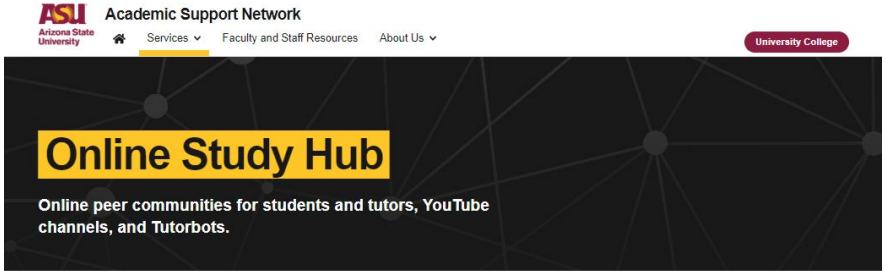


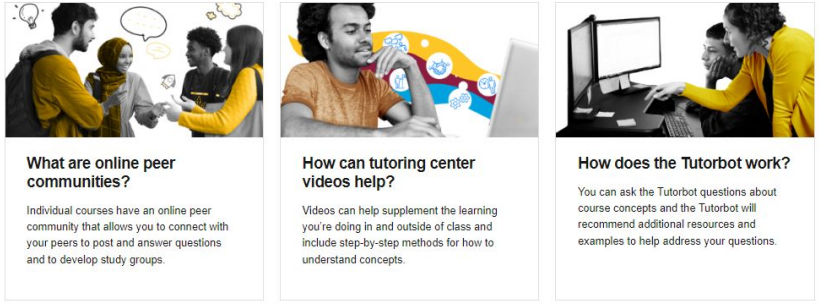
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

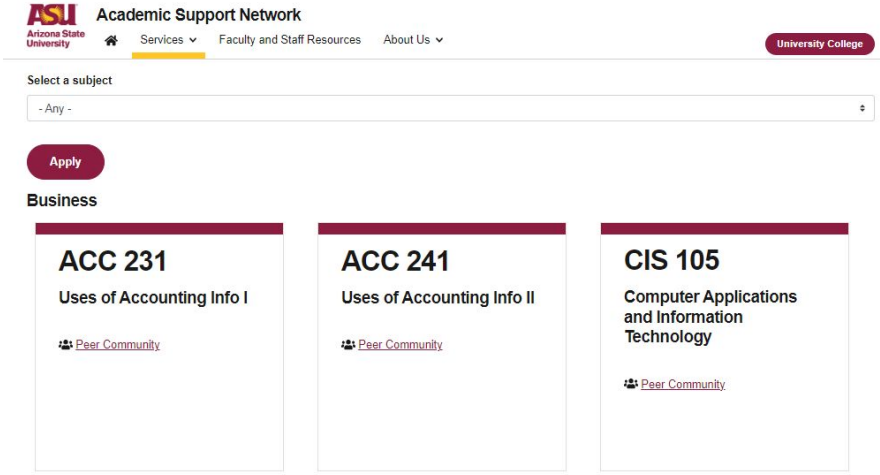
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



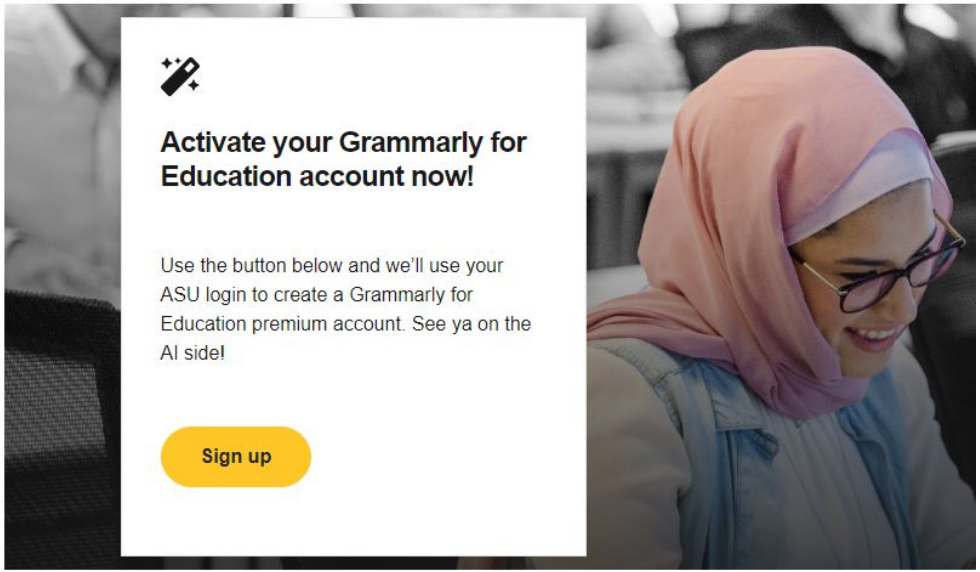




Don't forget to check out the Online Study Hub for additional resources!

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