SER 321 A Session

SI Session

Monday, September 18th 2023

6:00 - 7:00 pm MST

Agenda

- Threading Mini-Quiz
- Thread Review
- Assignment Structure
- Running your Assignment

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
Thread Review

Which of the following correctly enables the Client to start a thread?

A.

```
public class Client implements Thread{
```

B.

```
public class Client makes Thread{
```

C.

```
public class Client extends Thread{
```

D.

```
public class Client permits Thread{
```

Check out the recording for the solution!

SER 321
Thread Review

```
client.start();
```

The code above starts which of the following methods?

A.

```
public void start() {
```

B.

```
public void run() {
```

C.

```
public void thread() {
```

D.

```
public void execute() {
```

Check out the recording for the solution!

SER 321
Thread Review

```
//create threaded server
ThreadedServer threadedServer = new ThreadedServer(                    );
//send off to work
threadedServer.start();
```

Which of the following constructors correctly initializes a ThreadedServer allowing the code above to function?

Check out the recording for the solution!

A.

```
public ThreadedServer(Socket sock) {
    this.conn = sock;
}
```

B.

```
public ThreadedServer(Socket sock, int id) {
    this.conn = sock;
    this.id = id;
}
```

C.

```
public ThreadedServer(int id) {
    this.id = id;
}
```

D.

```
public ThreadedServer(Socket sock, int id, Performer performer) {
    this.conn = sock;
    this.id = id;
    this.performer = performer;
}
```

SER 321

Socket Server - No Threads

Make Socket

Wait for connections

Handle the connection

Perform the task

Clean up - what is that again?

```
in.close();

out.close();

sock.close();
```

```
public static void main (String args[]) {
  Socket sock;
  try {
    //open socket
    ServerSocket serv = new ServerSocket( port: 8888); // create server socket on port 8888
    System.out.println("Server ready for 3 connections");
    // only does three connections then closes
    // NOTE: SINGLE-THREADED, only one connection at a time
    for (int rep = 0; rep < 3; rep++){
      System.out.println("Server waiting for a connection");
      sock = serv.accept(); // blocking wait
      // setup the object reading channel
      ObjectInputStream in = new ObjectInputStream(sock.getInputStream());

      // read in one object, the message. we know a string was written only by knowing what the client sent.
      // must cast the object from Object to desired type to be useful
      String s = (String) in.readObject();
      System.out.println("Received the String "+s);
      // read in the number, we know it's an integer because that's the second thing sent by the client.
      Integer i = (Integer) in.readObject();
      System.out.println("Received the Integer "+ i);

      // generate an output
      // get output channel
      OutputStream out = sock.getOutputStream();
      // create an object output writer (Java only)
      ObjectOutputStream os = new ObjectOutputStream(out);
      // write the whole message
      os.writeObject("Got it!");
      // make sure it wrote and doesn't get cached in a buffer
      os.flush();
    }
  } catch(Exception e) {e.printStackTrace();}
}
```

SockServer from JavaSimpleSock2 in examples Repo

SER 321

Threading your Server

Make Socket

Wait for connections

Start Thread

Handle the connection

Perform the task

Clean up

JavaThreadedSock in Sockets

```
ServerSocket serv = new ServerSocket(portNo);
while (true) {
  System.out.println("Threaded server waiting for connects on port " + portNo);
  sock = serv.accept();
  System.out.println("Threaded server connected to client-" + id);
  // create thread
  ThreadedSockServer myServerThread = new ThreadedSockServer(sock, id++);
  // run thread and don't care about managing it
  myServerThread.start();
}
```

```
public ThreadedSockServer(Socket sock, int id) {
  this.conn = sock;
  this.id = id;
}
```

```
public void run() {
  try {
    // setup read/write channels for connection
    ObjectInputStream in = new ObjectInputStream(conn.getInputStream());
    ObjectOutputStream out = new ObjectOutputStream(conn.getOutputStream());

    // read the digit being send
    String s = (String) in.readObject();
```

SER 321
Threading

Make Socket

Wait for connections

Start Thread

Handle the connection

Perform the task

Clean up

```
in.close();
out.close();
conn.close();
```

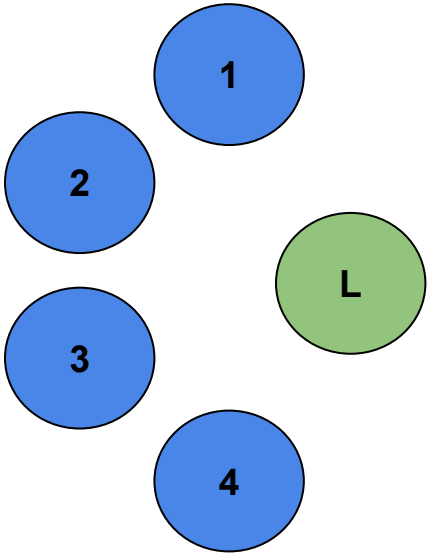
```
int index;
// while client hasn't ended
while (!s.equals("end")) {
  Boolean validInput = true;

  // checks if input only contains digits
  if (!s.matches( expr: "\\d+")) {
    validInput = false;
    out.writeObject("Not a number: https://gph.is/2yDymkn");
  }

  // if it contains only numbers
  if (validInput) {
    // convert to an integer
    index = Integer.valueOf(s);
    System.out.println("From client " + id + " get string " + index);
    if (index > -1 & index < buf.length) {
      // if valid, pull the line from the buffer array above and write it to socket
      out.writeObject(buf[index]);
    } else if (index == 5) {
      // fun surprise for mostly correct
      out.writeObject("Close but out of range: https://youtu.be/dQw4w9WgXcQ");
    } else {
      // really wrong
      out.writeObject("index out of range");
    }
  }
  //  wait for next token from the user
  s = (String) in.readObject();
}
```

SER 321
Assignment Structure

Leader with worker nodes



Theoretically each node is a different computer connected to the system scattered over the globe.

We are going to use threads to simulate

Each node receives a portion of the data to perform a task for/on

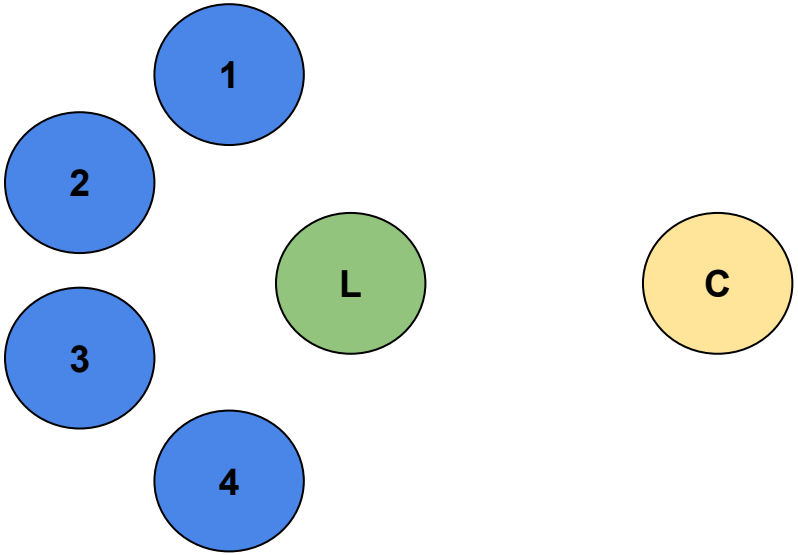
We are encrypting strings to simulate

Once complete, each node reports back to the leader with the result

SER 321

Assignment Structure

Leader with worker nodes



The leader is responsible for everything

- Data
- Partitioning data
- Nodes
- New nodes
- Unresponsive nodes
- Faulty nodes
- Node responses
- Client communication

SER 321

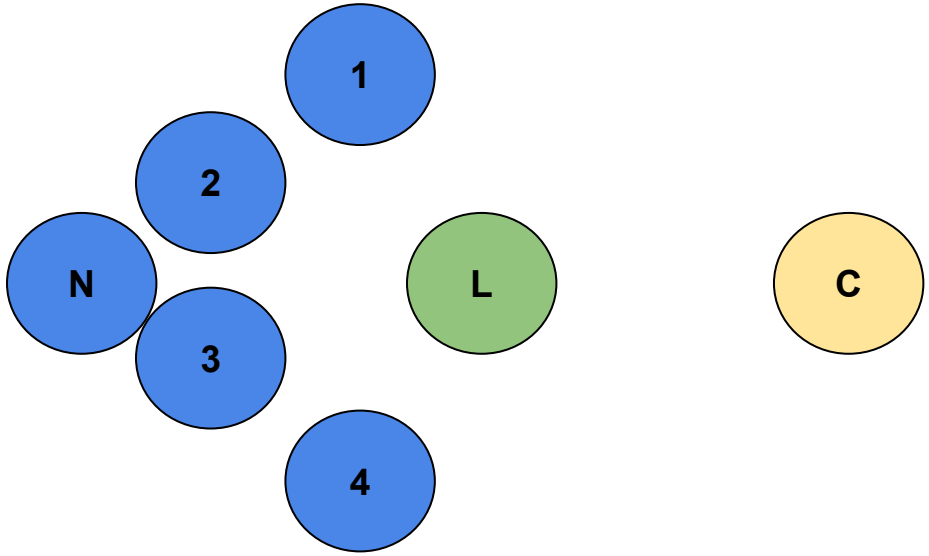
Assignment Structure

Leader with worker nodes

All nodes are identical

Really only need three classes then

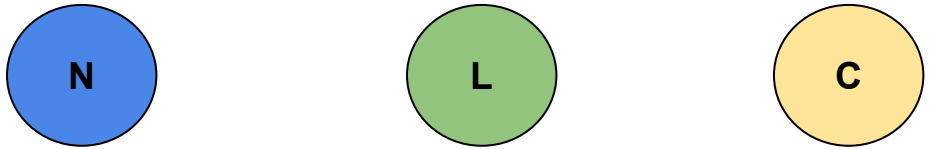
- Client
- Leader
- Node



SER 321

Assignment Structure

Leader with worker nodes



Need at least three nodes at all times

What if you drop below three?

Send error message to client - graceful!

Need a set max limit for nodes (8)

Nodes are threaded!

SER 321
Assignment Structure

Client will communicate with leader
then backs off to wait

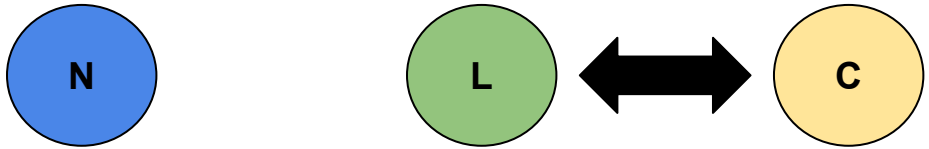
Connection is established

Leader prompts client for a sentence

Client sends sentence

Leader *does work*

Leader sends client the encrypted sentence



Think about the protocol you want to use!

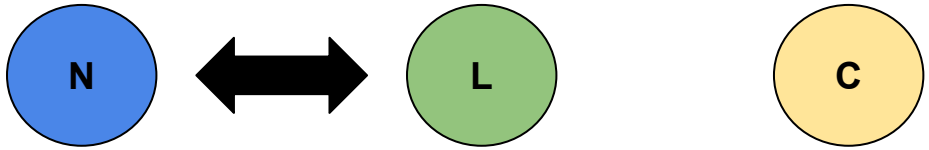
JSON or Protobuf?

Start getting a rough outline together

Two “Areas” to cover
- Leader-Client communication
- Leader-Node communication

SER 321
Assignment Structure

What does the node have to do?



Task 1:
- Receive data
- Encrypt data
- Return data

Task 2:
- Receive data
- Encrypt data
- **Check data**
- Return result

Both Encrypt steps should **use the same method**

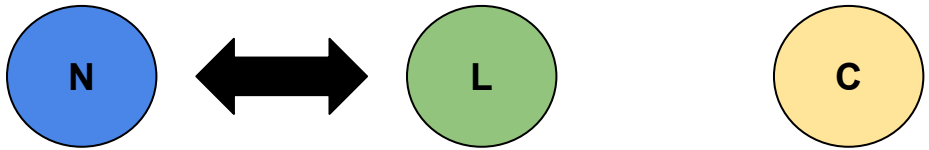
Could have an error in one method and not the other

How could we handle this elegantly?

Think in terms of protocol!

SER 321
Assignment Structure

Leader *does work*



Overview of Leader's job

- Split the data up for each node
- Send NodeData to each node
- Wait for response
- Receive encrypted data
- Start Consensus
    - Send NodeData and encrypted NodeData to a *new* node
    - Check response
        - If no - previous node was faulty
        - Else continue
    - Once all encryptions have been checked by a second node, continue
- Reassemble data
- Return data to client

SER 321

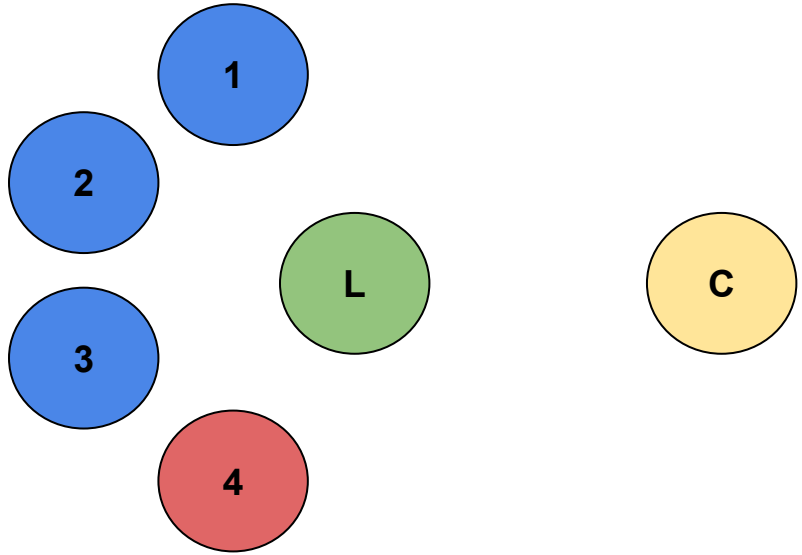
Running your Assignment

First start your leader

Then start at least 3 nodes

You can make a node faulty with the Fault flag

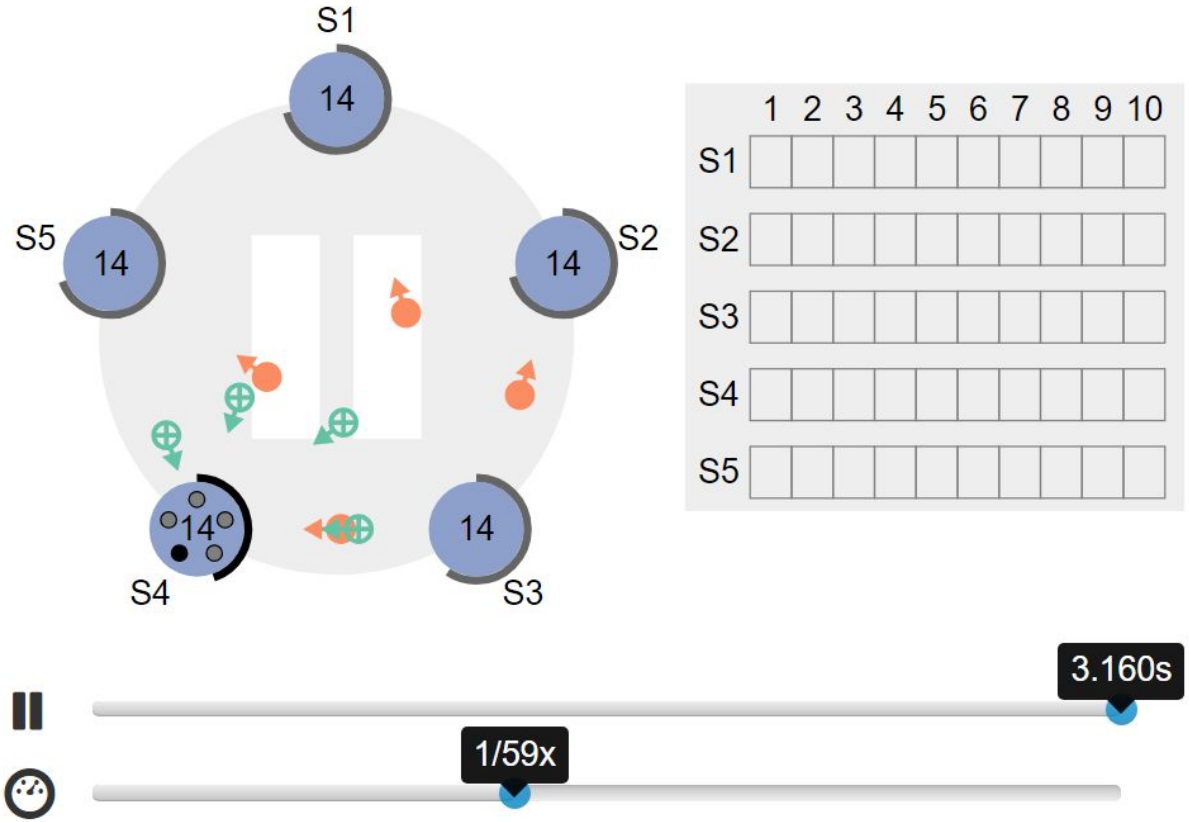
Then start your client



SER 321

RAFT

RAFT



Questions?

Survey:

https://bit.ly/asn_survey

20

Upcoming Events

SI Sessions:

- Wednesday September 20th 2023 6:00 pm MST

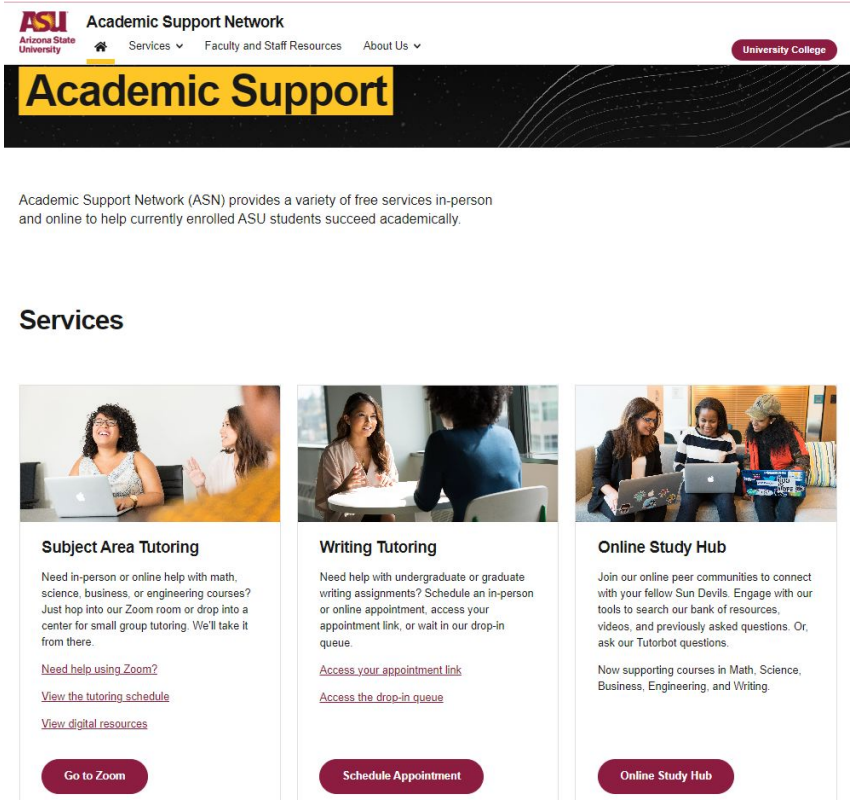
Review Sessions:

- TBD

More Questions?

Check out our other resources!

tutoring.asu.edu

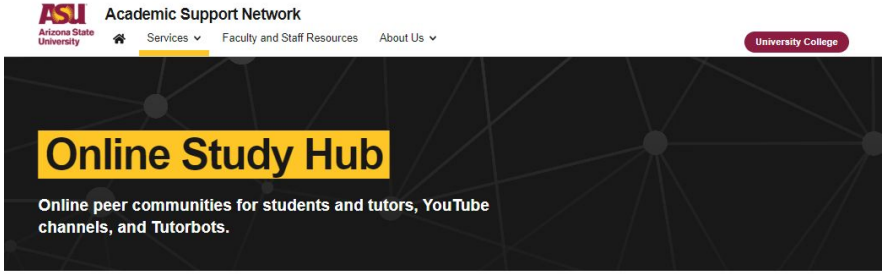


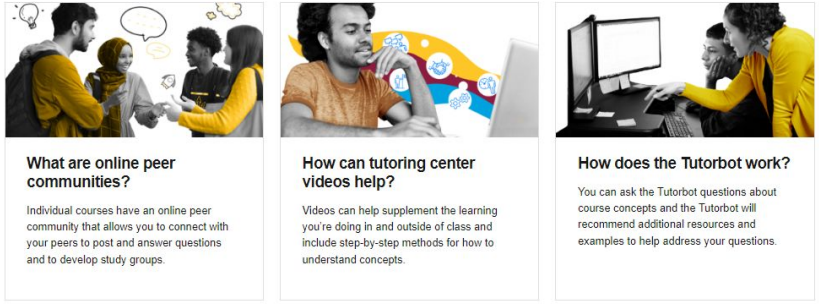
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

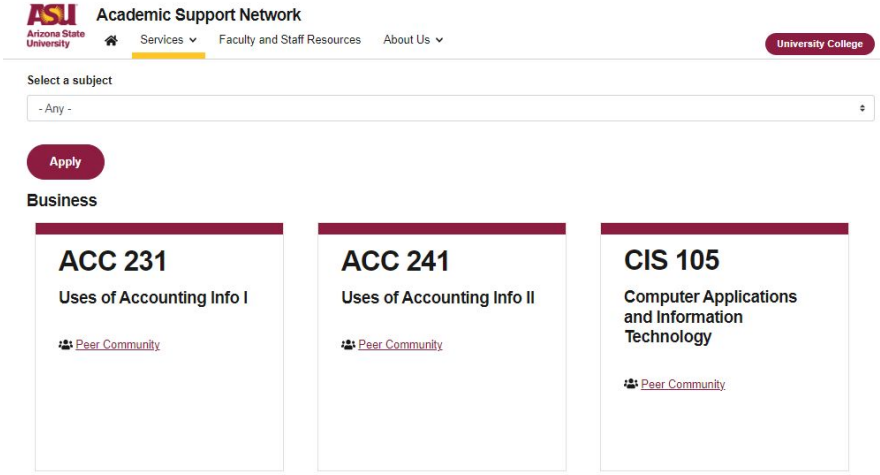
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub







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

Additional Resources

RAFT

Examples Repo

- SimplePeerToPeer