SER 321 B Session

SI Session

Thursday, November 7th 2024

7:00 pm - 8:00 pm MST

Agenda

- Protocol Tips
- JSON Review
- HTTP Response Matching
- Serialization
- Threads!

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
Protocol Tips

Stay Organized!

Try to emulate the structure in 3-1

Format your Markdown!

Assign 3-1 Starter Code

SER 321

Protocol Tips

Table of Contents

```
<!-- TOC -->
  * [Protocol:](#protocol-)
    * [Echo:](#echo-)
    * [Add:](#add-)
    * [AddMany:](#addmany-)
    * [Roller:](#roller-)
    * [Inventory:](#inventory-)
    * [General error responses:]
<!-- TOC -->
```

```
## Protocol: ##

### Echo: ###

Request:

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

General response:
```

Protocol:

Echo:

Request:

```
{
    "type" : "echo",
    "data" : <String>
}
```

General response:

Assign 3-1 Starter Code

SER 321

Protocol Tips

Table of Contents

```
<!-- TOC -->
  * [Protocol:](#protocol-)
    * [Echo:](#echo-)
    * [Add:](#add-)
    * [AddMany:](#addmany-)
    * [Roller:](#roller-)
    * [Inventory:](#inventory-)
    * [General error responses:]
<!-- TOC -->
```

```
## Protocol: ##

### Echo: ###

Request:

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

####General response:####
```

Protocol:

Echo:

Request:

```
{
    "type" : "echo", -- ty
    "data" : <String>  --
}
```

####General response:####

Spaces are important!

Assign 3-1 Starter Code

SER 321

Protocol Tips

Table of Contents

```
<!-- TOC -->
  * [Protocol:](#protocol-)
    * [Echo:](#echo-)
      * [Request:](#request-)
      * [General response:](#gen
    * [Add:](#add-)
    * [AddMany:](#addmany-)
    * [Roller:](#roller-)
    * [Inventory:](#inventory-)
    * [General error responses:]
<!-- TOC -->
```

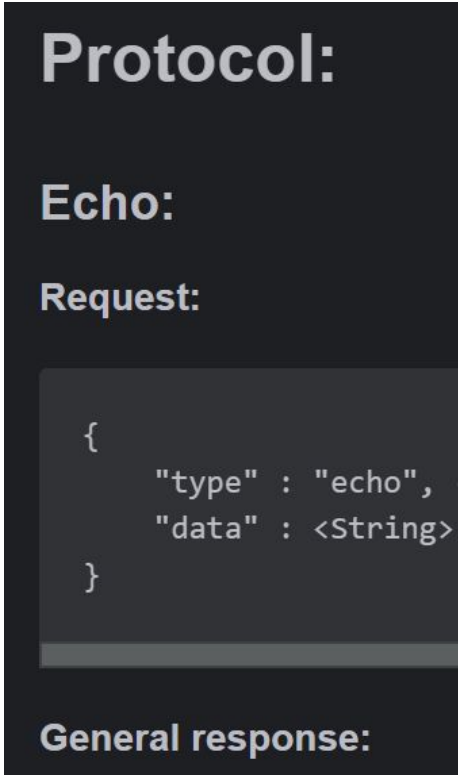
```
## Protocol: ##

### Echo: ###

#### Request: ####

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

#### General response: ####
```



Assign 3-1 Starter Code

SER 321

JSON

Which of the following would be a valid response?

```
{
    "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>
}
```

Assign 3-1 Starter Code

SER 321

JSON

Which of the following would be a valid response?

```
{
    "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

Why are the others invalid?

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

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

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

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

Assign 3-1 Starter Code

SER 321

JSON

Which of the following would be a valid response?

```
{
    "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”,
    “ok” : false,
    “echo” : <String>
}
```

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

C.
```
{
    “type” : “echo”,
    “ok” : false
}
```

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

Assign 3-1 Starter Code

Which of the following would be a valid response?

SER 321

JSON

```
{
    "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

Why are the others invalid?

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

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

C.
```
{
    “type” : “echo”,
    “ok” : false
}
```

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

SER 321

HTTP Matching

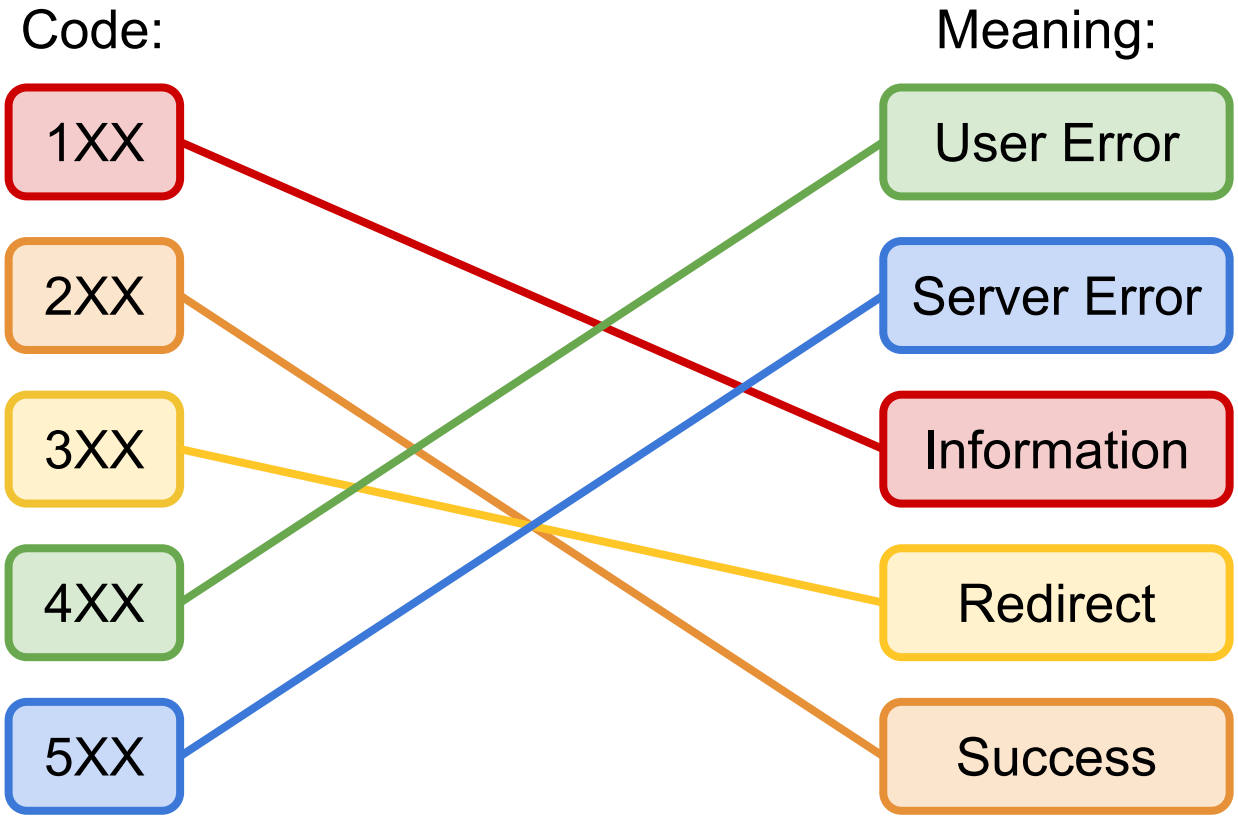
Match the HTTP response code with its meaning:

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:



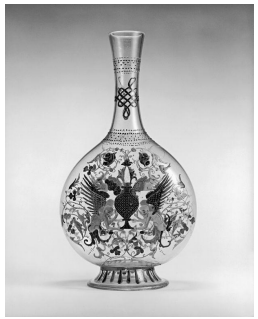
SER 321
Serialization

What is *serialization*?

"Translating data structures or object states for storage or transmission"

SER 321
Serialization

What is *serialization*?



Data

"Translating data structures or object states for storage or transmission"

SER 321
Serialization

What is *serialization*?

Serialized Data

"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

SER 321
Serialization

Binary

Text

Two main approaches for storing the content…

What about the data format?

JSON

Java Object Serialization

Protocol Buffers

XML

SER 321
Serialization

Binary

Text

Who uses *TEXT*?

Text

JSON

Java Object Serialization

Protocol Buffers

Text

XML

SER 321
Serialization

Binary

Text

What does this imply?

Who uses ***BINARY***?

Text JSON

Binary Java Object Serialization

Binary Protocol Buffers

Text XML

SER 321
Serialization

Streams and their types

Generic Superclass

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

Buffered Stream

Bytes

Data Stream

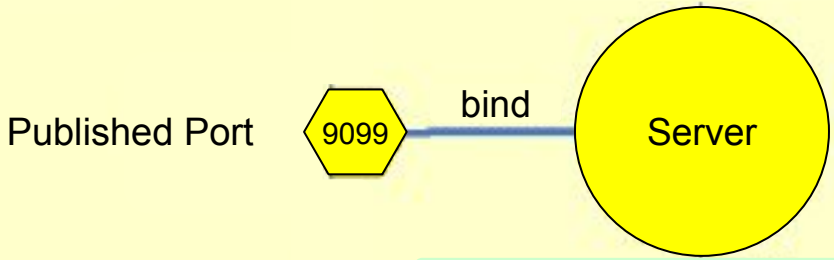
Primitive DATA Types

Object Stream

Java Objects

SER 321
Sockets!

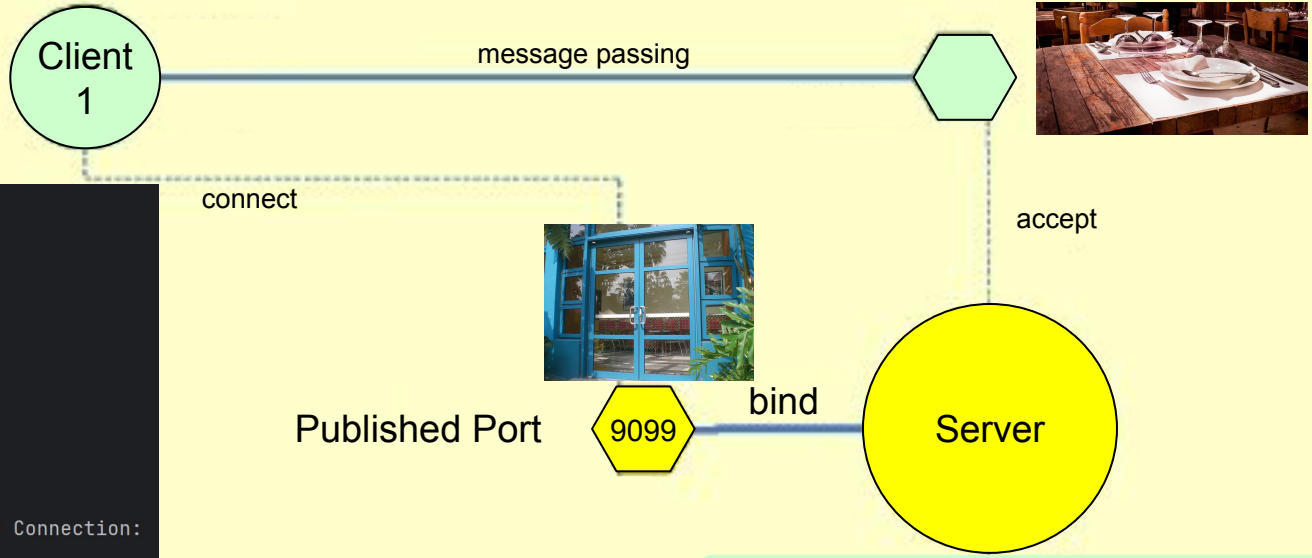
```
> Task :runServer
Server ready for connections
Server is listening on port: 9099
-----
Values of the ServerSocket Object:
Inet Address: 0.0.0.0/0.0.0.0
Local Port: 9099
Server waiting for a connection
Server connected to client
-----
Values of the Client Socket Object after Connection:
        Inet 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
System Layout

You have two systems…

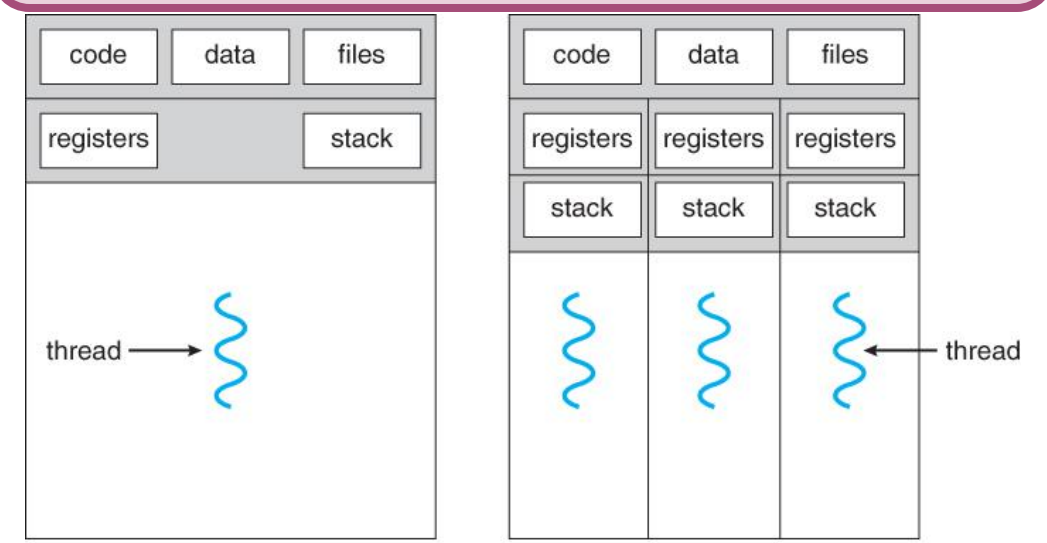
How can we test our server with multiple clients?

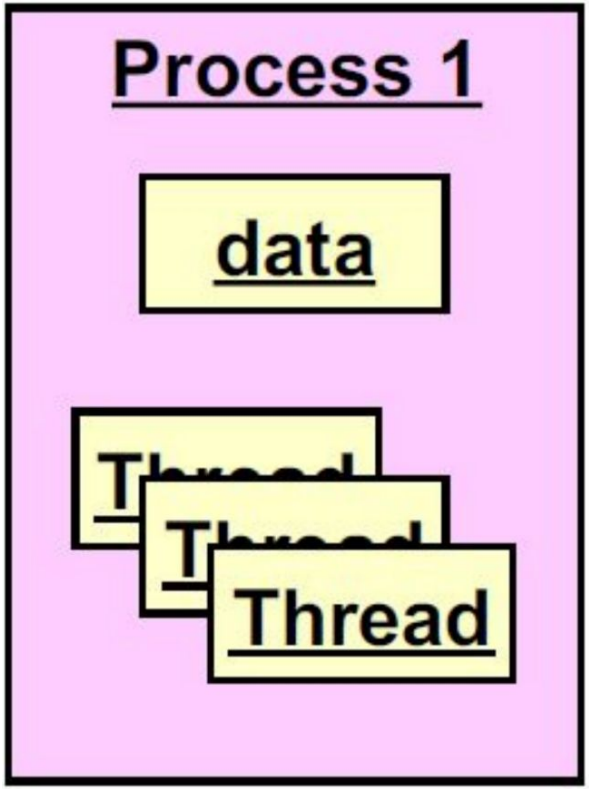
?

SER 321
Threads

What does that imply?

Remember that they exist *within* the parent process





SER 321
Threading Pitfalls

Race Condition	A thread never gains access to the resource it needs
Starvation	A thread is only able to acquire some of the resources it needs
Deadlock	More than one thread accesses a single resource at the same time

SER 321
Threading Pitfalls

Race Condition	A thread never gains access to the resource it needs
Starvation	A thread is only able to acquire some of the resources it needs
Deadlock	More than one thread accesses a single resource at the same time

NetworkDeadlock

SER 321
Threading Pitfalls

As the project name implies, we encounter a ***deadlock***.

But what happened?

Client

```
class SockClient {
    public static void main (String args[]) throws Exception {
        Socket      sock = new Socket( host: "localhost", port: 8888);    //Any IP name

        ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
        ObjectOutputStream out = new ObjectOutputStream(sock.getOutputStream());

        String s = (String) in.readObject();
        out.writeObject("Back at you");

        in.close();
        out.close();
        sock.close();
    }
}
```

Server

```
class SockServer {
    public static void main (String args[]) throws Exception {

        int count = 0;
        ServerSocket    serv = new ServerSocket( port: 8888);

        Socket sock = serv.accept();

        ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
        ObjectOutputStream out = new ObjectOutputStream(sock.getOutputStream());

        String s = (String) in.readObject();
        System.out.println("Received " + s);
        out.writeObject("Back at you");
        System.out.println("Received " + s);

        in.close();
        out.close();
        sock.close();
    }
}
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
server
<=========----> 75% EXECUTING [1m 33s]
> :server
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
client
Starting a Gradle Daemon, 1 busy and 1 stopped Daemons could not be reused, us
e --status for details
<=========----> 75% EXECUTING [53s]
> :client
```

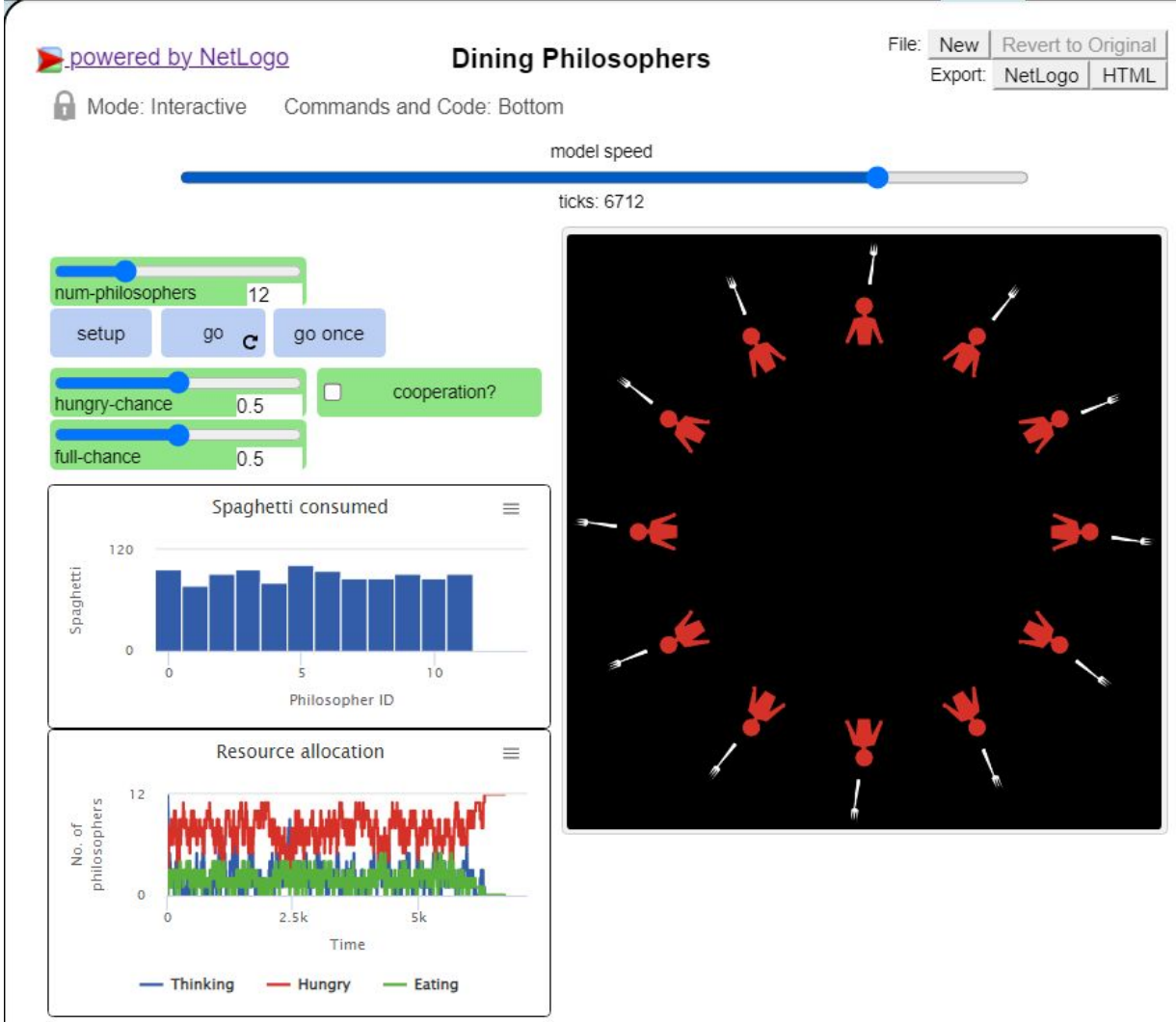
Dining Philosophers

SER 321
Threading Pitfalls

What does *Spaghetti Consumed* represent?

What does *Thinking* represent?

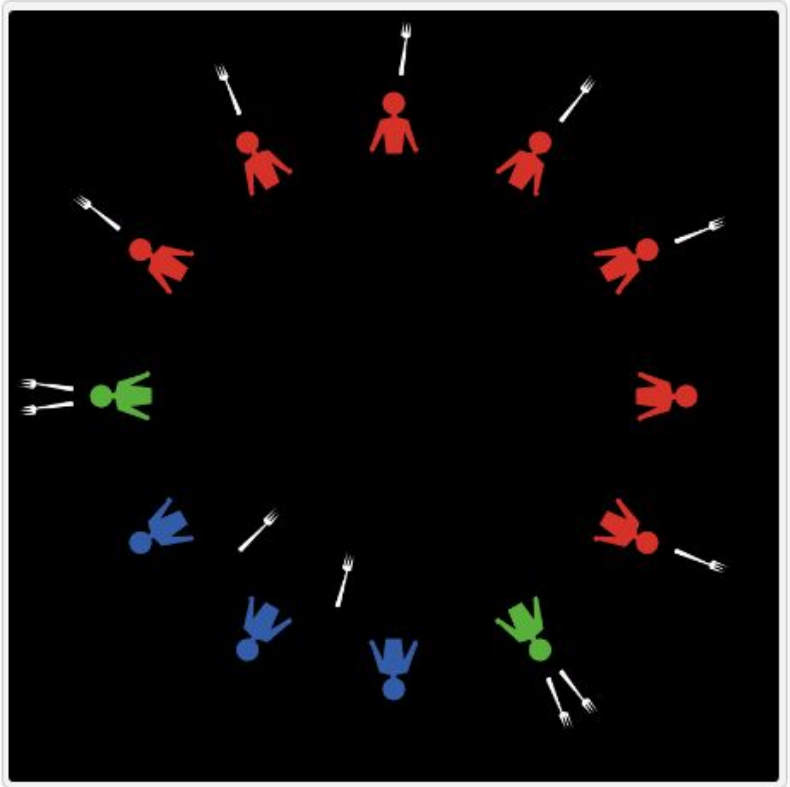
What does *Hungry* represent?



SER 321

Threading Pitfalls

Can we take a guess at what is happening here?



Austin Walter's Traffic Comparison

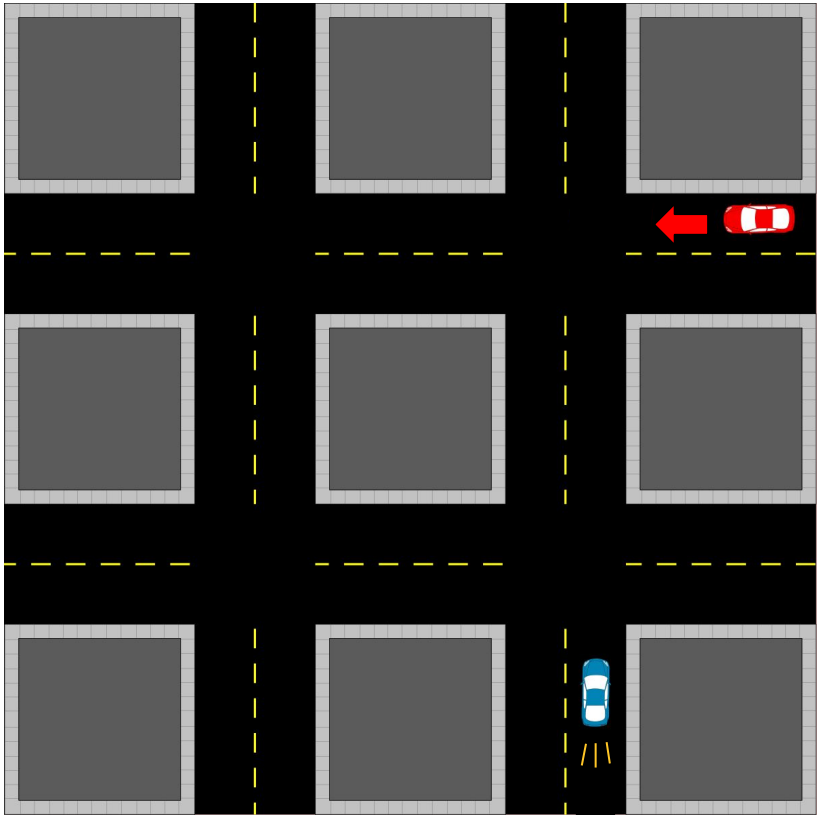
SER 321

Threading Pitfalls

Race Condition

Crash

More than one thread accesses a single resource at once



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

Race Condition

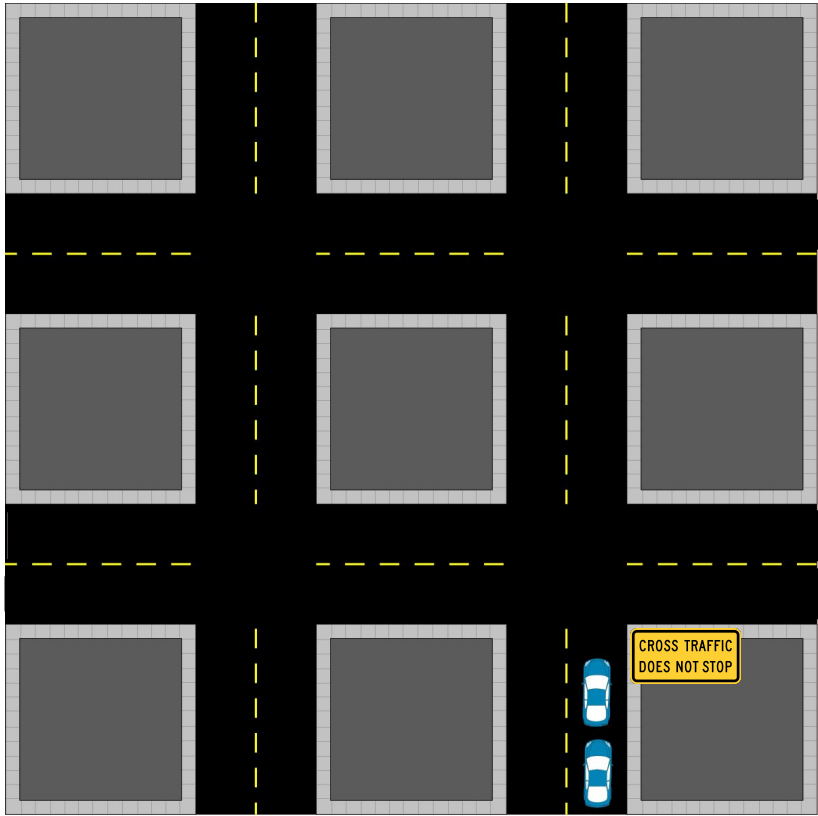
Crash

More than one thread accesses a single resource at once

Starvation

Cross Traffic

A thread never gains access to the resource it needs



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

Race Condition — **Crash**

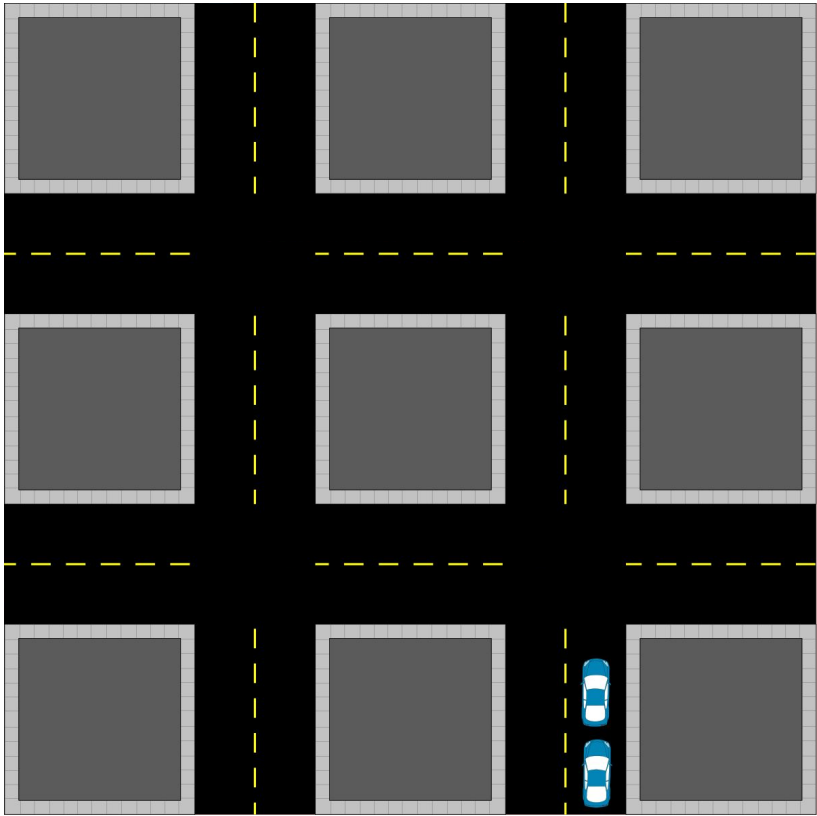
More than one thread accesses a single resource at once

Starvation — **Cross Traffic**

A thread never gains access to the resource it needs

Deadlock — **Gridlock**

A thread is only able to acquire some of the needed resources



SER 321

Scratch Space

Upcoming Events

SI Sessions:

- ~~Sunday, November 10th at 7:00 pm MST~~ **CANCELLED - Happy Veteran's Day!**
- Tuesday, November 12th at 10:00 am MST
- Thursday, November 14th at 7:00 pm MST

Review Sessions:

- Sunday, December 1st at 7:00 pm MST - **2 hour Review Session**
- Tuesday, December 3rd at 10:00 am MST - **Q&A Session**

Questions?

Survey:

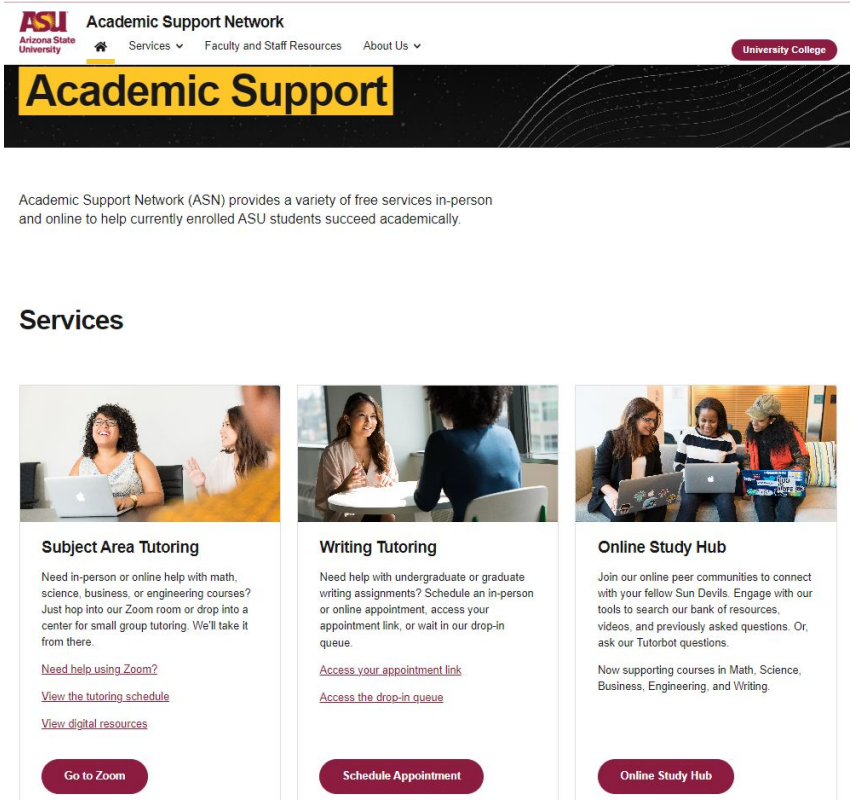
https://asuasn.info/ASNSurvey

37

More Questions?

Check out our other resources!

tutoring.asu.edu

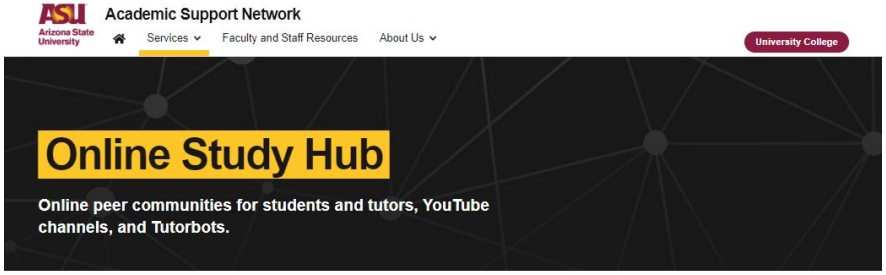


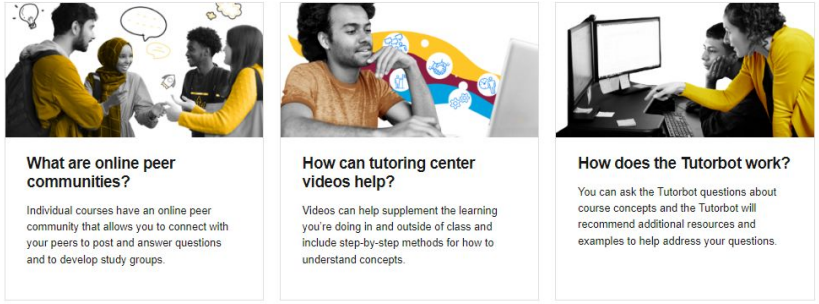
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

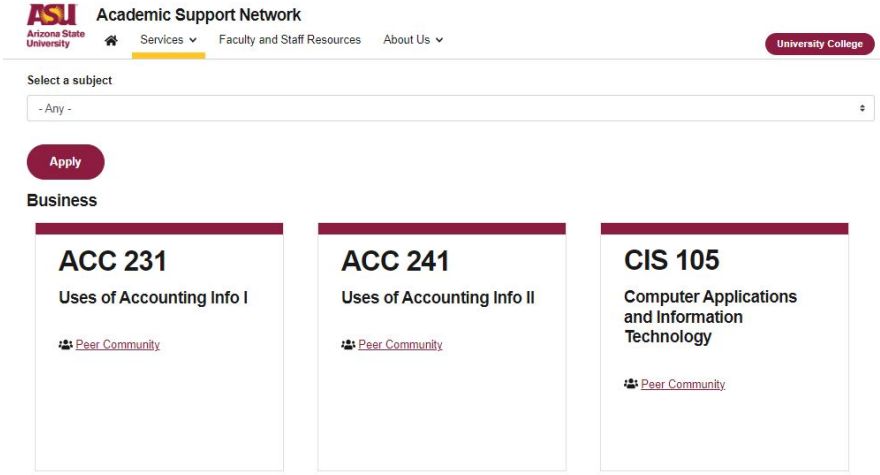
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



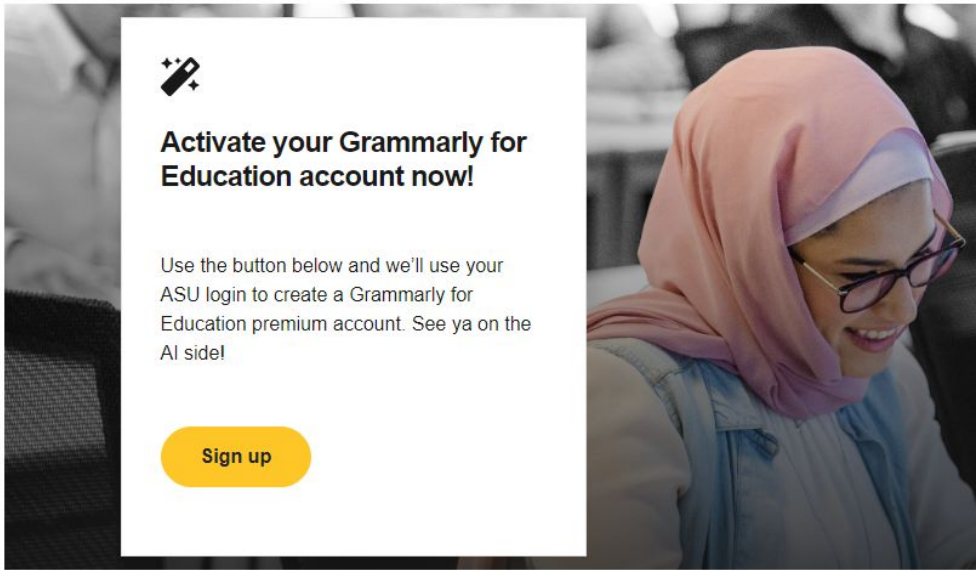




Don't forget to check out the Online Study Hub for additional resources!

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
- Dining Philosophers Interactive
- Austin G Walters Traffic Comparison