SER 321 B Session

SI Session

Thursday, November 14th 2024

7:00 pm - 8:00 pm MST

Agenda

Protobuf PSA

Threading Pitfalls

Concurrency Structures

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

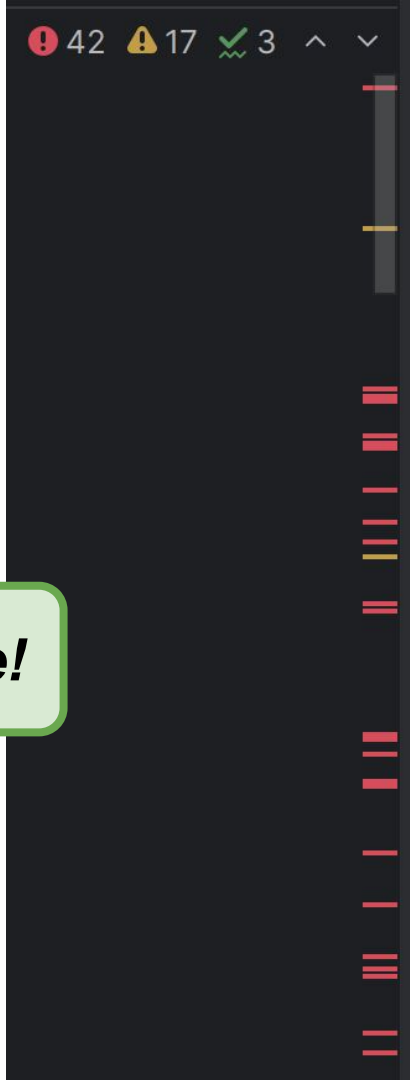
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321
Protobufs

Quick PSA for Protobufs!

Check out the recording for the discussion!

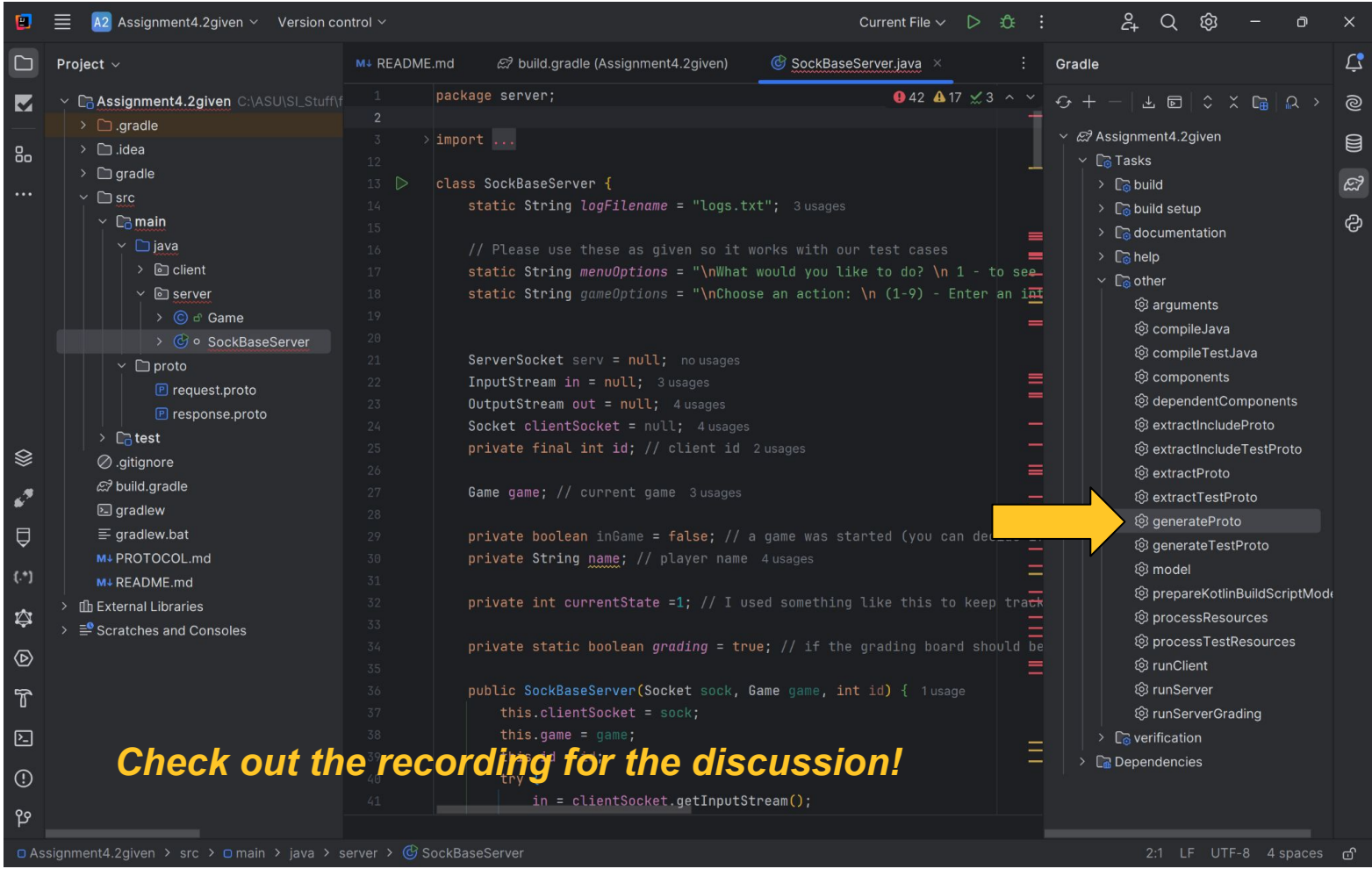
You must generate the Protobufs for use!



4-2 Starter Code

SER 321

Protobufs

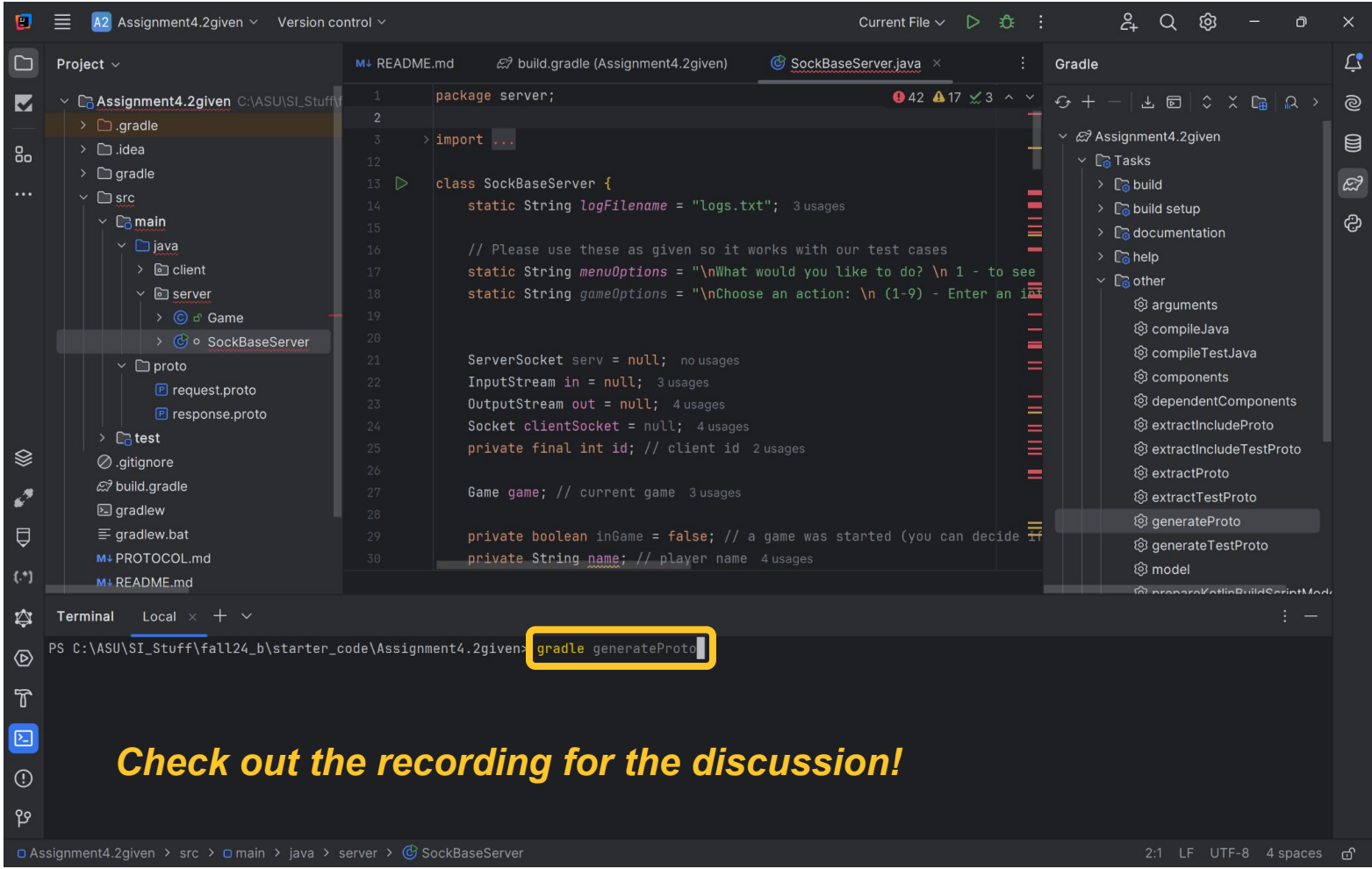


Check out the recording for the discussion!

4-2 Starter Code

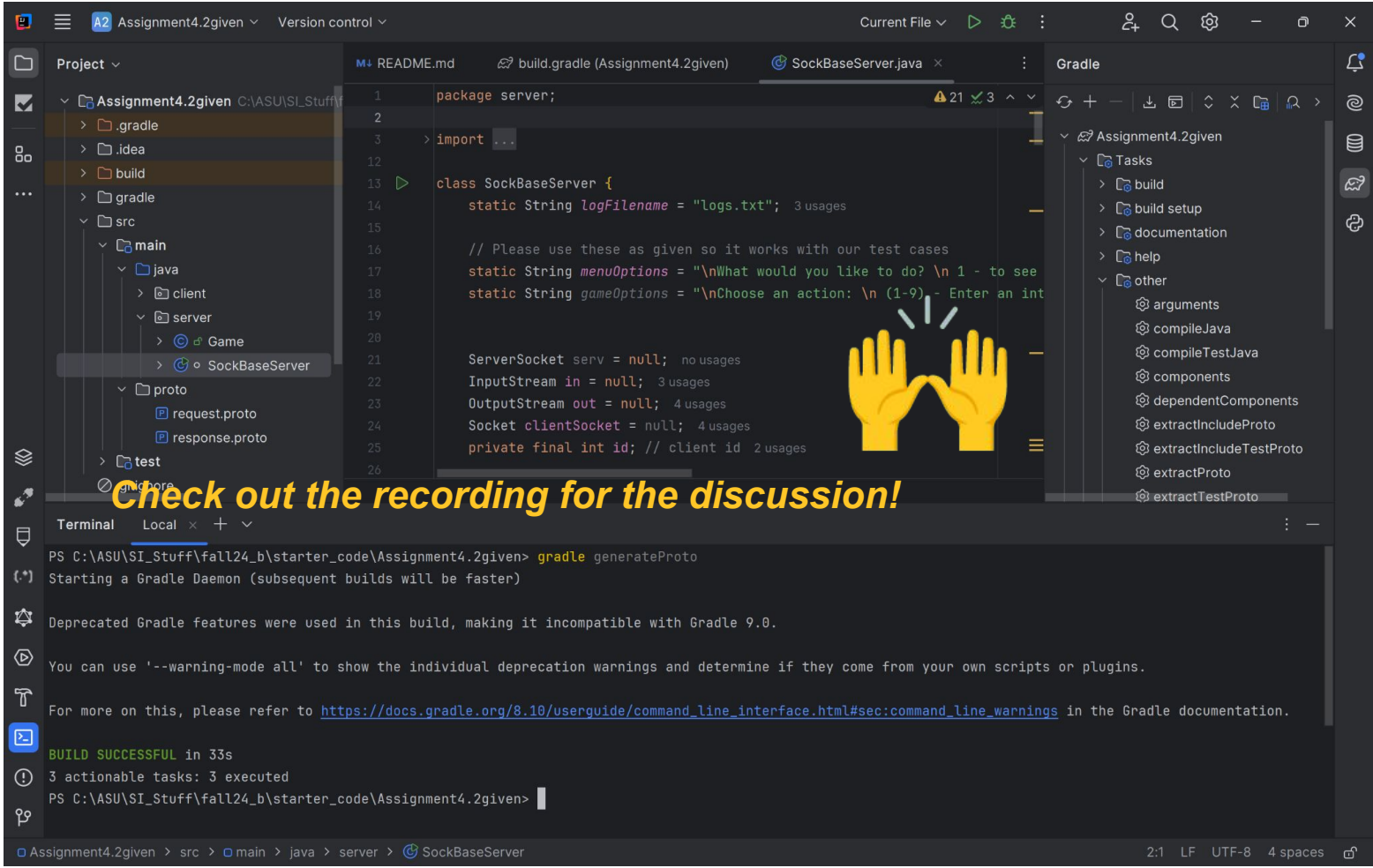
SER 321

Protobufs



4-2 Starter Code

SER 321
Protobufs



4-2 Starter Code

Check out the recording for the discussion!

SER 321
Protobufs

Options for Message Creation

```
Response.newBuilder()
 .setResponseType(Response.ResponseType.GREETING)
 .setMessage("Hello " + name + " and welcome to a simple game of Sudoku.")
 .setMenuoptions(menuOptions)
 .setNext(currentState)
 .build();
```

SockBaseServer

Create the message in a single statement

Create the message in increments

```
Request.Builder req = Request.newBuilder();

switch (response.getResponseType()) {
    case GREETING:
        System.out.println(response.getMessage());
        req = chooseMenu(req, response);
        break;
```

SockBaseClient - main

4-2 Starter Code

Check out the recording for the discussion!

SER 321
Protobufs

Options for Message Creation

```
static Request.Builder chooseMenu(Request.Builder req, Response response) throws IOException {
    while (true) {
        System.out.println(response.getMenuoptions());
        System.out.print("Enter a number 1-3: ");
        BufferedReader stdin = new BufferedReader(new InputStreamReader(System.in));
        String menu_select = stdin.readLine();
        System.out.println(menu_select);
        switch (menu_select) {
            // needs to include the other requests
            case "2":
                // this is not a complete START request!! Just as examp
                req.setOperationType(Request.OperationType.START);
                return req;
            default:
                System.out.println("\nNot a valid choice, p
                break;
        }
    }
}
```

SockBaseClient - chooseMenu

Need one more step…

```
Request.Builder req = Request.newBuilder();

switch (response.getResponseType()) {
    case GREETING:
        System.out.println(response.getMessage());
        req = chooseMenu(req, response);
        break;
```

SockBaseClient - main

```
req.build().writeDelimitedTo(out);
```

4-2 Starter Code

Check out the recording for the discussion!

SER 321
Protobufs

Parsing Messages

GETTERS!

```
System.out.println("Got a response: " + response.toString());
```

```
switch (response.getResponseType()) {
    case GREETING:
        System.out.println(response.getMessage());
        req = chooseMenu(req, response);
        break;
```

Fetch a single value

Fetch a *repeated* value

```
for (Entry lead: response3.getLeaderList()){
    System.out.println(lead.getName() + ": " + lead.getPoints());
}
```

Austin Walter's Traffic Comparison

SER 321

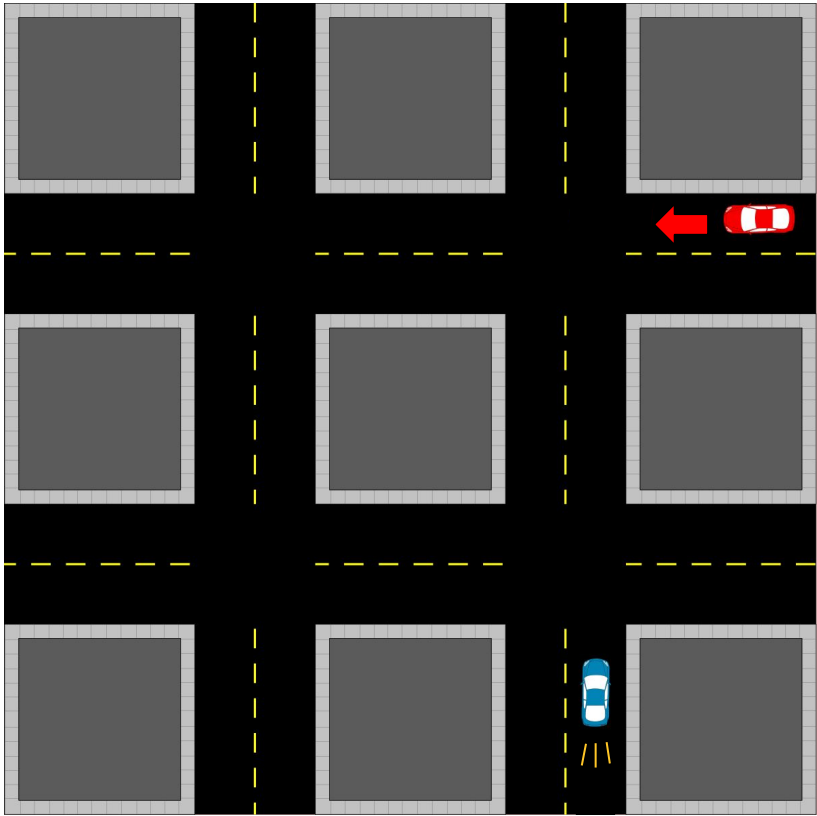
Threading Pitfalls

Check out the recording for the discussion!

Race Condition

Crash

More than one thread accesses a single resource at once



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

Check out the recording for the discussion!

Race Condition

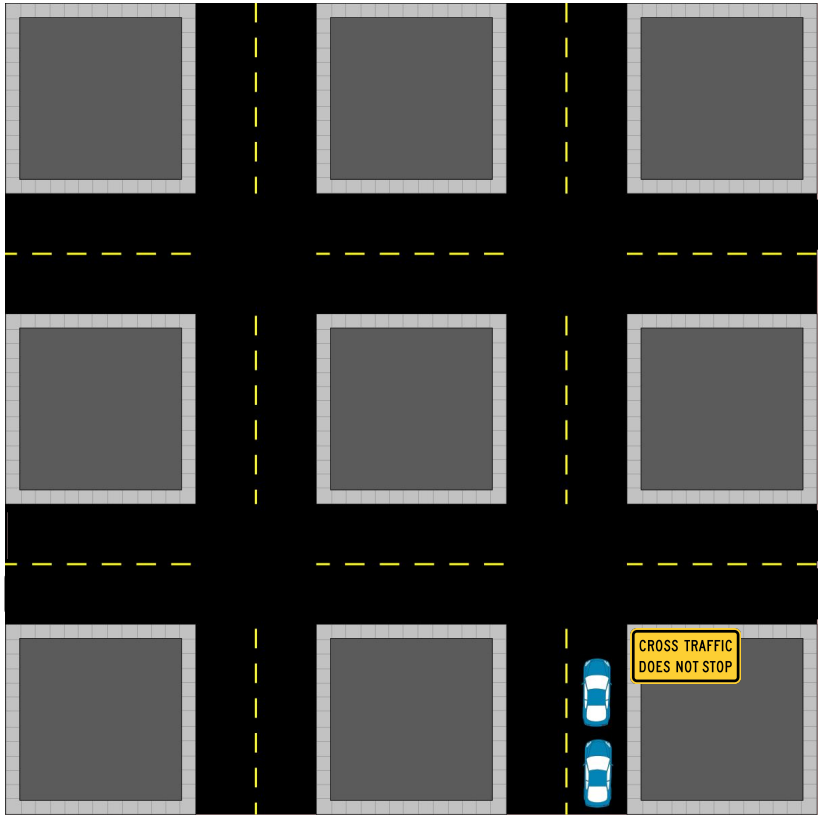
Crash

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

Check out the recording for the discussion!

Race Condition — **Crash**

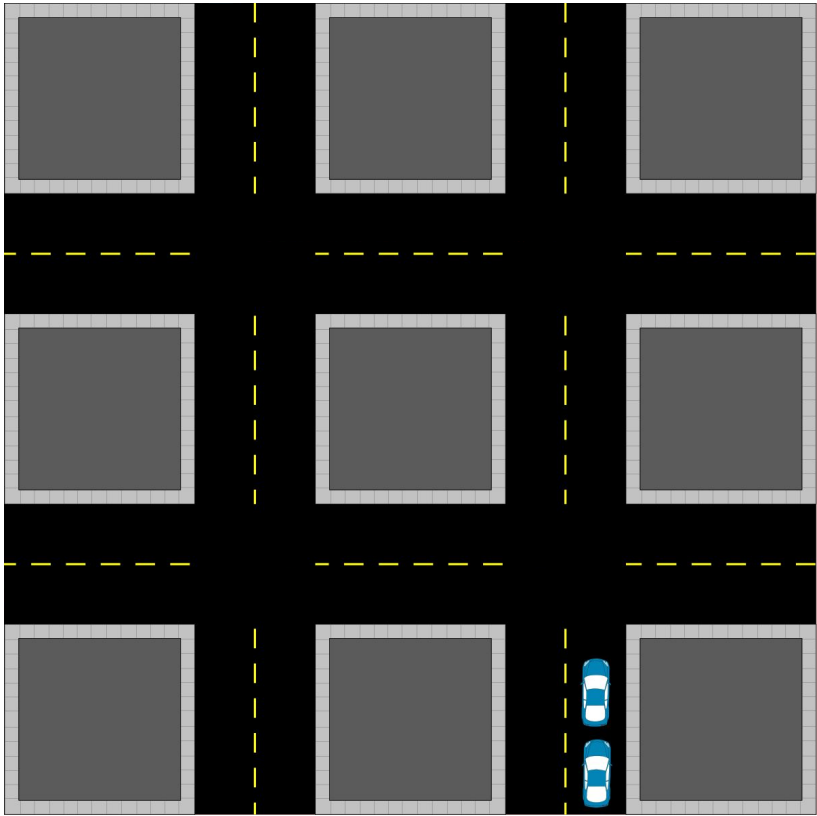
More than one thread accesses a single resource at once

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

Concurrency Structures

Check out the recording for the discussion!

Can we name some concurrency structures?

Atomic Operations & Variables

Locks

Semaphores

Monitors

SER 321

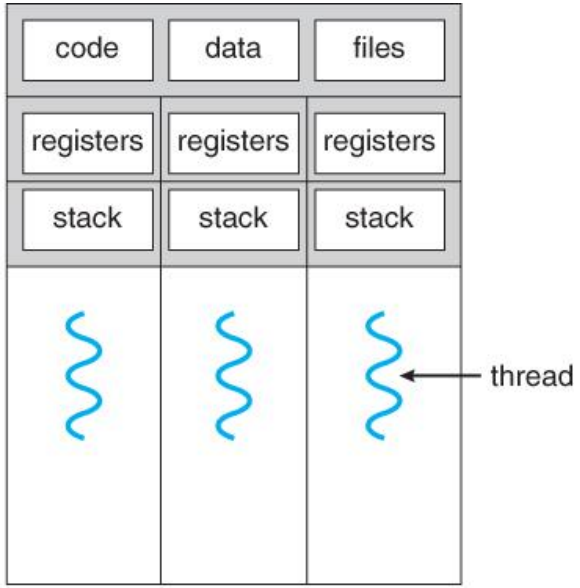
Concurrency Structures

Check out the recording for the discussion!

Atomic Operations & Variables

Recall *registers*…

Ensures updates are immediately visible for the local copy in *each thread*




```
main:
    pushq   %rbp
    movq    %rsp, %rbp
    subq    $48, %rsp
    call    __main
    movl    $5, -4(%rbp)
    movl    $12, -8(%rbp)
    movl    -4(%rbp), %eax
    addl    $7, %eax
    movl    %eax, -12(%rbp)
    movl    -8(%rbp), %edx
    movl    -12(%rbp), %eax
    addl    %edx, %eax
    movl    %eax, -16(%rbp)
    movl    -16(%rbp), %eax
    movl    %eax, %edx
    leaq    .LC0(%rip), %rax
    movq    %rax, %rcx
    call    printf
    movl    $0, %eax
    addq    $48, %rsp
    popq    %rbp
    ret
```

Check out the recording for the discussion!

SER 321

Concurrency Structures

Pros and Cons?

Locks

Acquire the Lock

- Open & Enter
- Close & Lock

Release the Lock → Unlock & Exit

Check out the recording for the discussion!

SER 321

Concurrency Structures

How am I different from a lock?

Semaphores

More than one stall!

Acquire Lock ⇨ Open & Enter; Close & Lock

Release Lock ⇨ Unlock & Exit

Semaphores support ***more than one*** acquirer

When would that be beneficial?

Check out the recording for the discussion!

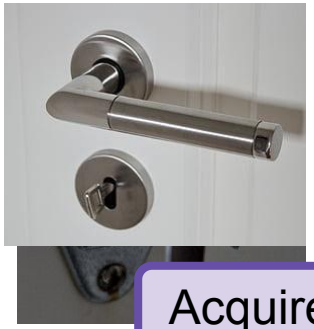
SER 321

Concurrency Structures

Pros and Cons?

Monitors

You lock the main door instead!



Acquire Lock ⇨ Open & Enter

Close & Lock

Covers the ***entire object***

Release Lock ⇨ Unlock & Exit

Check out the recording for the discussion!

SER 321
Concurrency Structures

RECAP

- Atomic Operations & Variables
  - ***YOU*** control the locks directly
- Locks
  - ***YOU*** control the locks directly
- Semaphores
  - ***YOU*** control the locks directly
- Monitors
  - Locks managed for you

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Sunday, November 17th at 7:00 pm MST
- Tuesday, November 19th at 10:00 am MST
- Thursday, November 21st 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:

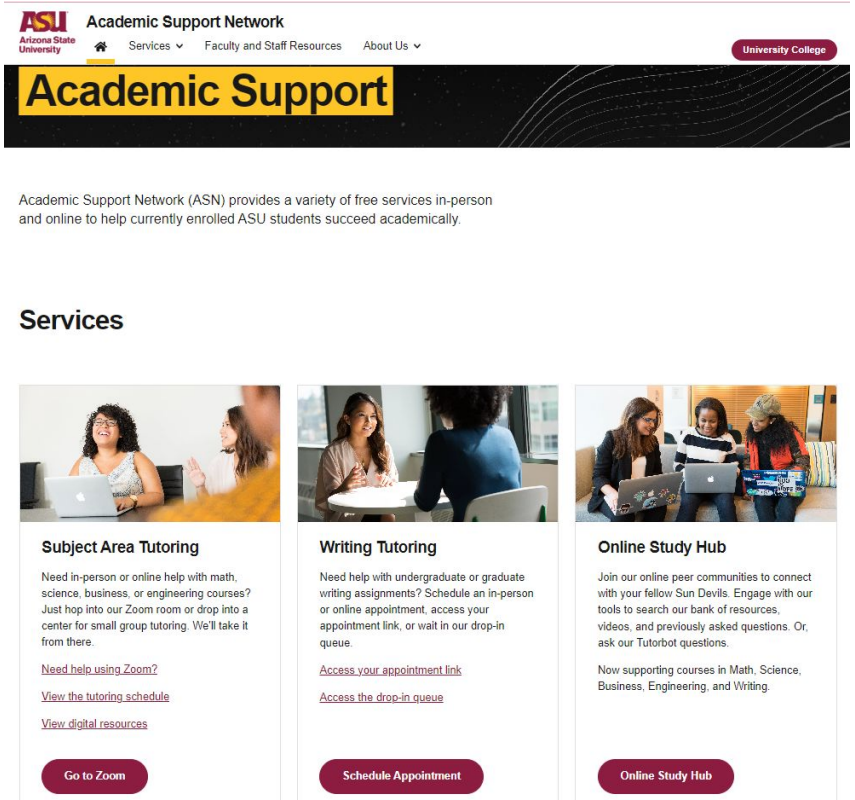
https://asuasn.info/ASNSurvey

23

More Questions?

Check out our other resources!

tutoring.asu.edu

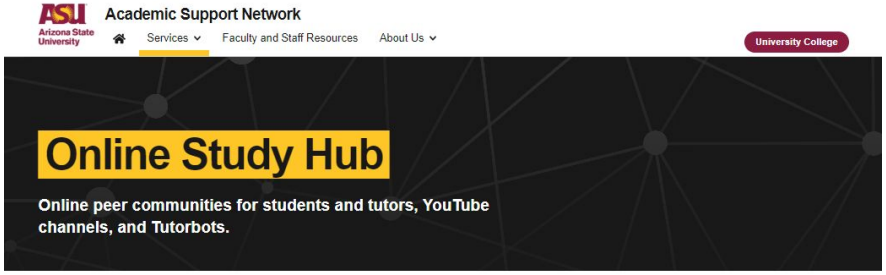


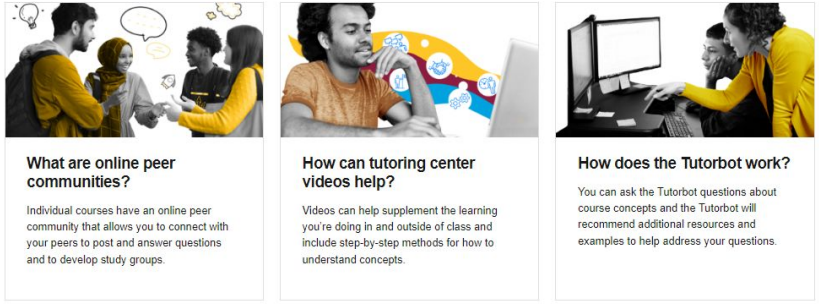
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

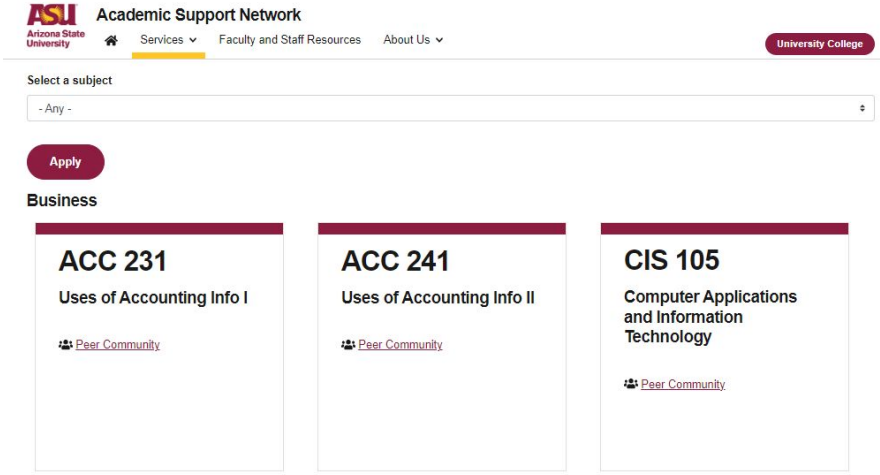
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



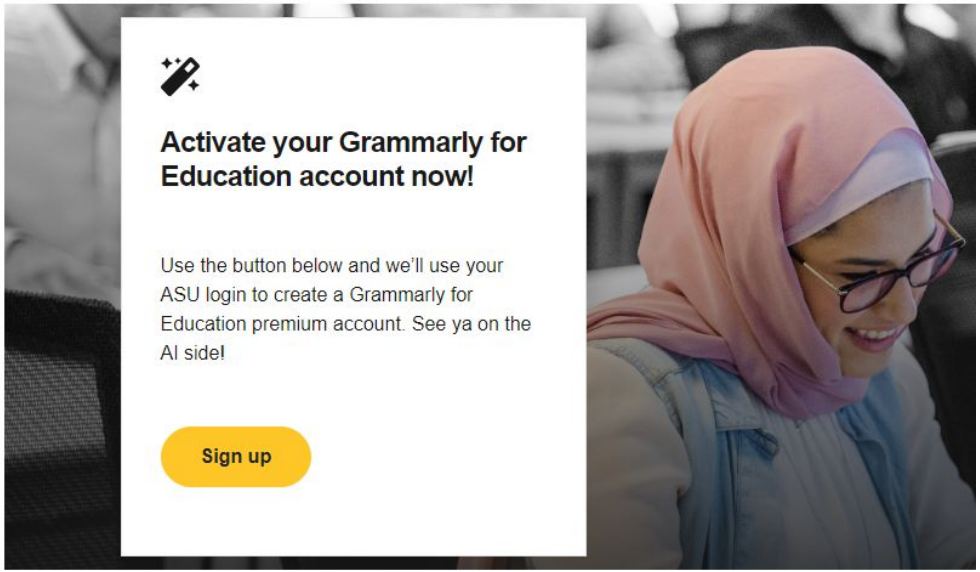




Don't forget to check out the Online Study Hub for additional resources!

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