

SER 321 B Session

SI Session

Sunday, April 13th 2025

7:00 pm - 8:00 pm MST

Agenda



Connections

Gradle Task Structure Review

Handling Multiple Clients

Why we thread our Server

How to thread our Server

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

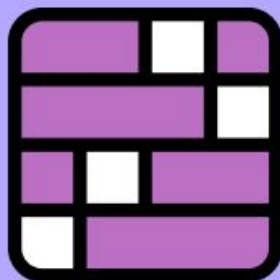
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321
Connections

[Connections!](#)

*Check out the recording to
watch our game, or click
the link to play yourself!*

The New York Times **Games**



Connections

SER 321

Gradle Review

Which of the following will run the main method in `/java/taskone/Server.java` with `gradle runTask1` ?

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

```
task runServer(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'taskone.Server.runTask1'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

A.

```
task runServer(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'taskone.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

B.

```
task1 runServer(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'taskone.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

C.

```
task runTask1(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'taskone.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

D.

SER 321

Gradle Review

Which of the following will run the main method in `/java/tasktwo/Server.java` with `gradle runTask2` ?

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

```
task runTask2(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'taskone.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

A.

```
task runTask2(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'tasktwo.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

B.

```
task2 runServer(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'tasktwo.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

C.

```
task runServer(type: JavaExec) {
    group 'server'
    description 'Creates Server socket waits for messages'

    classpath = sourceSets.main.runtimeClasspath

    main = 'tasktwo.Server'
    standardInput = System.in

    args 8000;
    if (project.hasProperty('port')) {
        args(project.getProperty('port'));
    }
}
```

D.

SER 321

Gradle Review

Which of the following will run the main method in `/java/taskone/Client.java` with `gradle runClient` ?

```
task runClient(type: JavaExec) {
    group 'client'
    description 'Creates client socket sends a message to the server'

    classpath = sourceSets.main.runtimeClasspath
    standardInput = System.in

    main = 'taskone.Client'
    standardInput = System.in

    if (project.hasProperty("host") && project.hasProperty('port')) {
        args(project.getProperty('host'), project.getProperty('port'));
    }
}
```

A.

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

```
task runClient(type: JavaExec) {
    group 'client'
    description 'Creates client socket sends a message to the server'

    classpath = sourceSets.main.runtimeClasspath
    standardInput = System.in

    main = 'taskone.Client'
    standardInput = System.in

    args("localhost", 8000);
    if (project.hasProperty("host") && project.hasProperty('port')) {
        args(project.getProperty('host'), project.getProperty('port'));
    }
}
```

B.

```
task runClient(type: JavaExec) {
    group 'client'
    description 'Creates client socket sends a message to the server'

    classpath = sourceSets.main.runtimeClasspath
    standardInput = System.in

    main = 'taskone.Client'
    standardInput = System.in

    if (project.hasProperty("host") && project.hasProperty('port')) {
        args(project.getProperty('host'), project.getProperty('port'));
    } else if (project.hasProperty("host")) {
        args(project.getProperty('host'), 8000);
    } else if (project.hasProperty("port")) {
        args("localhost", project.getProperty('port'))
    } else {
        args("localhost", 8000);
    }
}
```

C.

```
task runClient(type: JavaExec) {
    group 'client'
    description 'Creates client socket sends a message to the server'

    classpath = sourceSets.main.runtimeClasspath
    standardInput = System.in

    main = 'taskone.Client'
    standardInput = System.in

    if (project.hasProperty('host') && project.hasProperty('port')) {
        args(project.getProperty('host'), project.getProperty('port'));
    } else if (project.hasProperty('host')) {
        args(project.getProperty('host'), 8000);
    } else if (project.hasProperty('port')) {
        args('localhost', project.getProperty('port'));
    }
}
```

D.

SER 321

Single Threaded Server

What will happen if there are two clients?

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
Server ready for a connection
Server waiting for a connection
<=====--> 75% EXECUTING [20s]
```

```
> :SocketServer
```

Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit):
```

```
<=====--> 75% EXECUTING [14s]
```

```
> :SocketClient
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

Client 2

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
Server ready for a connection
Server waiting for a connection
<=====--> 75% EXECUTING [53s]
```

```
> :SocketServer
```

Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit):
```

```
<=====--> 75% EXECUTING [47s]
```

```
> :SocketClient
```

```
Hello!
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details
```

```
<=====--> 75% EXECUTING [15s]
```

```
> :SocketClient
```

Client 2

SER 321

Single Threaded Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
```

```
Server ready for a connection
```

```
Server waiting for a connection
```

```
Received the String Hello!
```

```
<=====--> 75% EXECUTING [1m 12s]
```

```
> :SocketServer
```

```
█
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
> Task :SocketClient
```

```
Please enter a String to send to the Server (enter
"exit" to quit):
```

```
<<====<=====--> 75% EXECUTING [59s] P
```

```
Please enter a Number to send to the Server (enter
0 to quit):
```

```
<=====--> 75% EXECUTING [1m 6s]
```

```
> :SocketClient
```

```
9
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details
```

```
<=====--> 75% EXECUTING [35s]
```

```
> :SocketClient
```

```
█
```

Server

Client 1

Client 2

SER 321**Single Threaded Server**

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Hello!
Received the Integer 9
<=====--> 75% EXECUTING [1m 27s]
> :SocketServer
█
```

Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketClient
```

```
> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit):
<=====--> 75% EXECUTING [59s]      P
Please enter a Number to send to the Server (enter
0 to quit):
<=====--> 75% EXECUTING [1m 18s]    9
and Hello! ... Got it!
Please enter a String to send to the Server (enter
"exit" to quit):
<=====--> 75% EXECUTING [1m 21s]
> :SocketClient
█
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketClient
```

```
Starting a Gradle Daemon, 2 busy and 4 stopped Daemons could not be reused, use --status for details
<=====--> 75% EXECUTING [49s]
> :SocketClient
█
```

Client 2

SER 321

Single Threaded Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Hello!
Received the Integer 9
<=====----> 75% EXECUTING [1m 55s]
> :SocketServer
█
```

Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketClient
```

```
> Task :SocketClient
Please enter a String to send to the Server (enter "exit" to quit):
<=====----> 75% EXECUTING [59s] P
Please enter a Number to send to the Server (enter 0 to quit):
<=====----> 75% EXECUTING [1m 18s] 9
and Hello! ... Got it!
Please enter a String to send to the Server (enter "exit" to quit):
<=====----> 75% EXECUTING [1m 49s]
> :SocketClient
exit█
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\JavaSimpleSock> gradle socketClient
```

```
Starting a Gradle Daemon, 2 busy and 4 stopped Daemons could not be reused, use --status for details
<=====----> 75% EXECUTING [1m 18s]
> :SocketClient
█
```

Client 2

What do we think will happen?

SER 321**Single Threaded Server**

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Hello!
Received the Integer 9
Received the String exit
Received the Integer 0
Server waiting for a connection
<=====--> 75% EXECUTING [2m 15s]
> :SocketServer
█
```

Server

```
and Hello! ... Got it!
Please enter a String to send to the Server (enter
"exit" to quit)":
<=====--> 75% EXECUTING [2m 3s]
xitingketClient
```

Deprecated Gradle features were used in this build
, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individ-
ual deprecation warnings and determine if they come
from your own scripts or plugins.

See https://docs.gradle.org/7.4.2/userguide/command_line_interface.html#sec:command_line_warnings

BUILD SUCCESSFUL in 2m 5s

2 actionable tasks: 1 executed, 1 up-to-date

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock>
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details
```

```
> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit)":
<=====--> 75% EXECUTING [1m 37s]
> :SocketClient
█
```

Client 2

SER 321**Single Threaded Server**

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer
```

```
> Task :SocketServer
```

```
Server ready for a connection
```

```
Server waiting for a connection
```

```
Received the String Hello!
```

```
Received the Integer 9
```

```
Received the String exit
```

```
Received the Integer 0
```

```
Server waiting for a connection
```

```
<=====--> 75% EXECUTING [2m 45s]
```

```
> :SocketServer
```

```
█
```

```
and Hello! ... Got it!
```

```
Please enter a String to send to the Server (enter
"exit" to quit):
```

```
<=====--> 75% EXECUTING [2m 3s]           e
xitingketClient
```

```
Deprecated Gradle features were used in this build
, making it incompatible with Gradle 8.0.
```

```
You can use '--warning-mode all' to show the indiv
idual deprecation warnings and determine if they c
ome from your own scripts or plugins.
```

```
See https://docs.gradle.org/7.4.2/userguide/command\_line\_interface.html#sec:command\_line\_warnings
```

```
BUILD SUCCESSFUL in 2m 5s
```

```
2 actionable tasks: 1 executed, 1 up-to-date
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> █
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
```

```
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details
```

```
> Task :SocketClient
```

```
Please enter a String to send to the Server (enter
"exit" to quit):
```

```
<=====--> 75% EXECUTING [2m 7s]
```

```
> :SocketClient
```

```
Hello!
```

Server

Client 1

Client 2

SER 321

Single Threaded Server

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer

> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Hello!
Received the Integer 9
Received the String exit
Received the Integer 0
Server waiting for a connection
Received the String Hello!
<=====--> 75% EXECUTING [3m 7s]
> :SocketServer
█
```

Server

```
and Hello! ... Got it!
Please enter a String to send to the Server (enter
"exit" to quit"):
<=====--> 75% EXECUTING [2m 3s]
xitingketClient

Deprecated Gradle features were used in this build
, making it incompatible with Gradle 8.0.

You can use '--warning-mode all' to show the individ
ual deprecation warnings and determine if they c
ome from your own scripts or plugins.

See https://docs.gradle.org/7.4.2/userguide/command
line\_interface.html#sec:command\_line\_warnings

BUILD SUCCESSFUL in 2m 5s
2 actionable tasks: 1 executed, 1 up-to-date
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> █
```

Client 1

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details

> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit"):
<=====--> 75% EXECUTING [2m 24s]
Please enter a Number to send to the Server (enter
0 to quit"):
<=====--> 75% EXECUTING [2m 30s]
> :SocketClient
77 █
```

Client 2

SER 321

Single Threaded Server

Why?



```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets
\JavaSimpleSock> gradle socketServer

> Task :SocketServer
Server ready for a connection
Server waiting for a connection
Received the String Hello!
Received the Integer 9
Received the String exit
Received the Integer 0
Server waiting for a connection
Received the String Hello!
<=====--> 75% EXECUTING [3m 7s]
> :SocketServer
█
```

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.

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sock
ets\JavaSimpleSock> gradle socketClient
Starting a Gradle Daemon, 2 busy and 4 stopped Dae
mons could not be reused, use --status for details

> Task :SocketClient
Please enter a String to send to the Server (enter
"exit" to quit"):
<=====--> 75% EXECUTING [2m 24s]    P
lease enter a Number to send to the Server (enter
0 to quit"):
<=====--> 75% EXECUTING [2m 30s]
> :SocketClient
77█
```

Check out the recording
for the discussion!

Server

Client 1

Client 2

SER 321

Threaded Server

Given the standard server socket steps...

Ideas on how we could introduce threads?

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

Check out the recording for the discussion!

SER 321

Threaded Server

Given the standard server socket steps...

Ideas on how we could introduce threads?

1. Define Params

2. Create Socket

3-5. Mark Socket to Listen

6. Wait for Connection

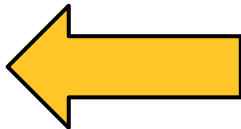
7. Handle Client Connection

8. Close Client Connection

9. Continue Listening

Why do we send the *client socket* to the thread?

7. Send Client Socket to thread



Check out the recording for the discussion!

SER 321 Threads

*Check out the
recording for the
discussion!*

1. Define Params

2. Create Socket

3-5. Mark Socket to Listen

6. Wait for Connection

7. Send Client **Socket** to Thread

8. Close Client Connection

9. Continue Listening

1

2 & 3-5

9

6

7

8

```
public static void main(String args[]) throws IOException {
    Socket sock = null;
    int id = 0;
    try {
        if (args.length != 1) {
            System.out.println
                ("Usage: gradle ThreadedSockServer --args=<port num>");
            System.exit(0);
        }
        int portNo = Integer.parseInt(args[0]);
        if (portNo <= 1024)
            portNo = 8888;
        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();
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (sock != null) sock.close();
    }
}
```

SER 321 Threads

```
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();
        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");
            }
        }
    }
}
```

Client
A

Server

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

```
public static void main(String args[]) throws IOException {
    Socket sock = null;
    int id = 0;
    try {
        if (args.length != 1) {
            System.out.println
                ("Usage: gradle ThreadedSockServer --args=<port num>");
            System.exit( code: 0);
        }
        int portNo = Integer.parseInt(args[0]);
        if (portNo <= 1024)
            portNo = 8888;
        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();
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (sock != null) sock.close();
    }
}
```

SER 321 Threads

```
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();
        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");
            }
        }
    }
}
```

Client
A

Server

Client
B

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

```
public static void main(String args[]) throws IOException {
    Socket sock = null;
    int id = 0;
    try {
        if (args.length != 1) {
            System.out.println
                ("Usage: gradle ThreadedSockServer --args=<port num>");
            System.exit( code: 0);
        }
        int portNo = Integer.parseInt(args[0]);
        if (portNo <= 1024)
            portNo = 8888;
        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();
        }
    } catch (Exception e) {
        e.printStackTrace();
    } finally {
        if (sock != null) sock.close();
    }
}
```

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Tuesday, April 15th at 10:00 am MST
- Thursday, April 17th at 7:00 pm MST
- Sunday, April 20th at 7:00 pm MST

Review Sessions:

- Sunday, April 27th at **6:00 pm MST - 2 hour Exam Review Session**
- Tuesday, April 29th, at 10:00 am MST - **Q&A Session**

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

 **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

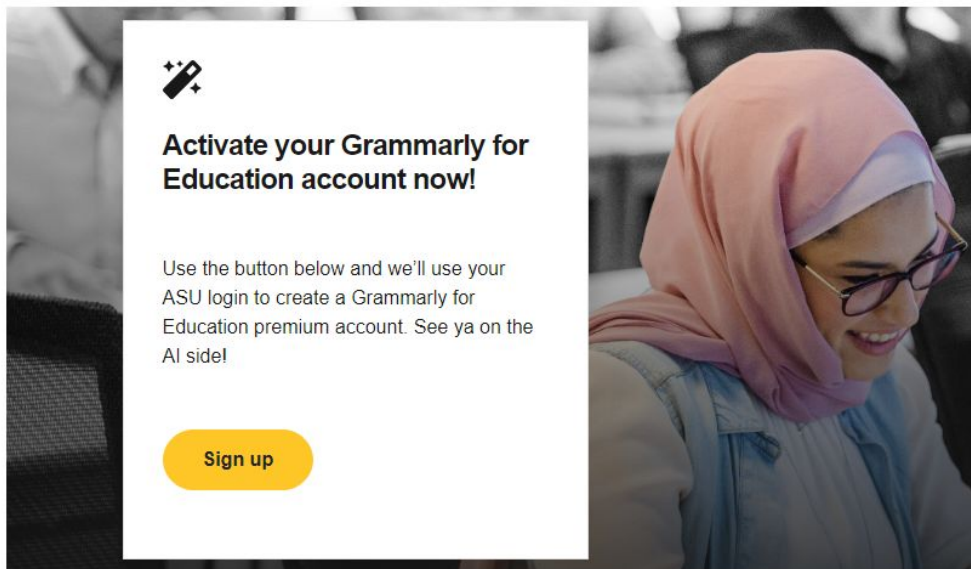
Computer Applications and Information Technology

 [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!



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](#)