SER 321 B Session

SI Session

Tuesday, October 29th 2024

10:00 am - 11:00 am MST

Agenda

JSON Practice

Client and Server Identification

Robustness

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

javax.swing API

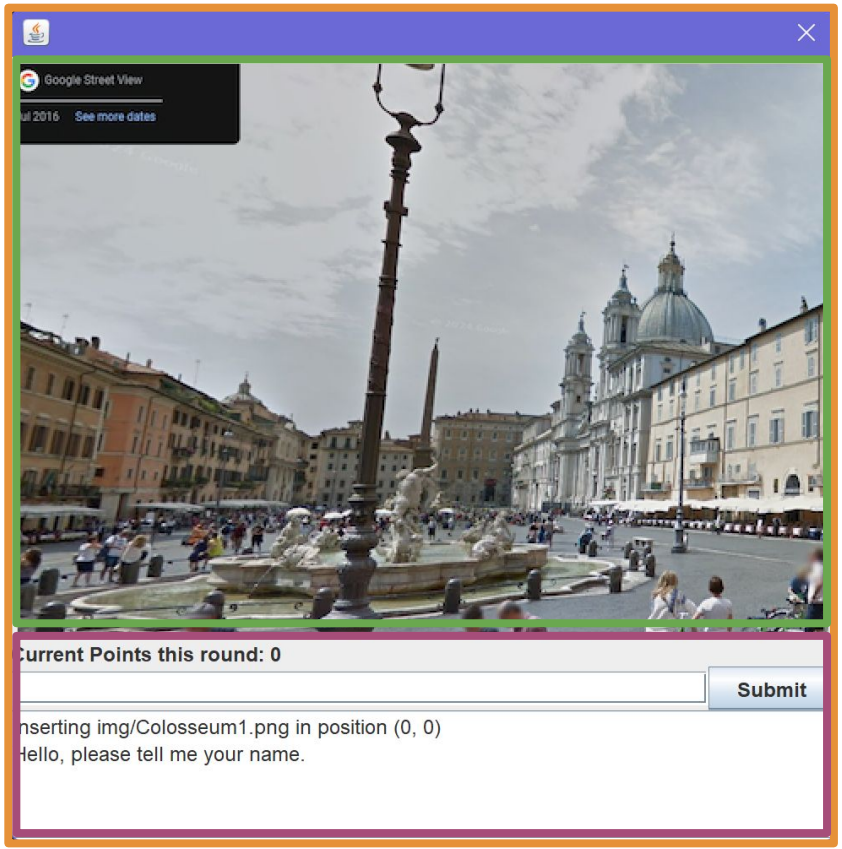
Swing Tutorials

SER 321

Assignment 3-2 GUI

🌟 GUI Walkthrough 🌟

Frame



picPanel

outputPanel

SER 321
Assignment 3-2 GUI

ClientGui
Points of Interest!

Line	Item
58	Constructor
86	Calling newGame()
114	show()
125	newGame()
139	insertImage()
(154)	appendOutput()
165	submitClicked()

Check out the recording for the discussion!

SER 321
Client/Server

Think Fast - Client or Server?

```
String host = args[0];
Socket server = new Socket(host, port);
```

Check out the recording for the discussion!

SER 321
Client/Server

Think Fast - Client or Server?

```
Socket clientSock;
ServerSocket sock = new ServerSocket(port);
```

Check out the recording for the discussion!

SER 321
Client/Server

Think Fast - Client or Server?

```
try {
  sock = new Socket(host,  port: 8888);
  OutputStream out = sock.getOutputStream();
  ObjectOutputStream os = new ObjectOutputStream(out);
  os.writeObject( message);
  os.writeObject( number);
  os.flush();
```

Check out the recording for the discussion!

```
  ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
  String i = (String) in.readObject();
  System.out.println(i);
  sock.close();
} catch (Exception e) {e.printStackTrace();}
```

SER 321
Client/Server

Think Fast - Client or Server?

```
try {
  ServerSocket serv = new ServerSocket( port: 8888);
  for (int rep = 0; rep < 3; rep++){
    sock = serv.accept();
    ObjectInputStream in = new ObjectInputStream(sock.getInputStream());

    String s = (String) in.readObject();
    System.out.println("Received the String "+s);
    Integer i = (Integer) in.readObject();
    System.out.println("Received the Integer "+ i);
          Check out the recording for the discussion!
    OutputStream out = sock.getOutputStream();
    ObjectOutputStream os = new ObjectOutputStream(out);
    os.writeObject("Got it!");
    os.flush();
  }
} catch(Exception e) {e.printStackTrace();}
```

SER 321

Making your Code Robust

What do we mean when we say “make sure your code is robust”?

Error Handling

Check out the recording for the discussion!

Check out the recording for the discussion!

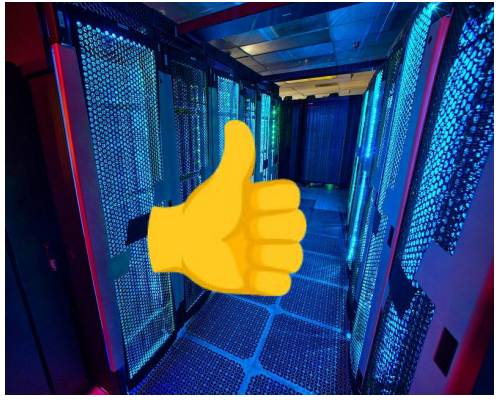
SER 321

Making your Code Robust

What do we mean when we say “make sure your code is robust”?

You

Buddy



Check out the recording for the discussion!

Sockets/Echo_Java

SER 321

Making your Code Robust

What do we mean when we say “make sure your code is robust”?

```
PS C:\ASU\SER321\examples_repo\ser321examp
les\Sockets\Echo_Java> gradle runServer

> Task :runServer
Server ready for connections
Server waiting for a connection
Server connected to client
<==========----> 75% EXECUTING [24m 44s]
> :runServer
```

```
PS C:\ASU\SER321\examples_repo\ser321example
Starting a Gradle Daemon, 1 busy and 3 stopp

> Task :runClient
Connected to server at localhost:9099
String to send>
<==========----> 75% EXECUTING [24m 25s]
> :runClient
```

Ctrl + C

What do you think will happen?

Check out the recording for the discussion!

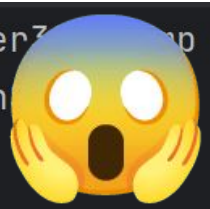
Sockets/Echo_Java

SER 321

Making your Code Robust

We crashed the server!

```
PS C:\ASU\SER321\examples_repo\ser3...p
les\Sockets\Echo_Java> gradle run
```



```
Server ready for connections
Server waiting for a connection
Server connected to client
java.net.SocketException: Connection rese
        at java.base/sun.nio.ch.NioSocketImpl.implRead(NioSocketImpl.java:320)
        at java.base/sun.nio.ch.NioSocketImpl.read(NioSocketImpl.java:347)
        at java.base/sun.nio.ch.NioSocketImpl$1.read(NioSocketImpl.java:800)
        at java.base/java.net.Socket$SocketInputStream.read(Socket.java:966)
        at Server.main(Server.java:48)

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

```
PS C:\ASU\SER321\examples_repo\ser321example
Starting a Gradle Daemon, 1 busy and 3 stopp
```

```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\Echo_Java> g
ient
Starting a Gradle Daemon, 1 busy and 3 stopped Daemons could not b
se --status for details

> Task :runClient
Connected to server at localhost:9099
String to send>
<=========----> 75% EXECUTING [24m 43s]
> :runClient
Terminate batch job (Y/N)? y
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\Echo_Java>
```

SER 321

Making your Code Robust

What happened?

Sockets/Echo_Java

```
while(true) {
        System.out.println("Server waiting for a connection");
        clientSock = sock.accept(); // blocking wait
        PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
        InputStream input = clientSock.getInputStream();
        System.out.println("Server connected to client");
        int numr = input.read(clientInput, off: 0, bufLen);
        while (numr != -1) {
          String received = new String(clientInput, offset: 0, numr);
          System.out.println("read from client: " + received);
          out.println(received);
          numr = input.read(clientInput, off: 0, bufLen);
        }
        input.close();
        clientSock.close();
        System.out.println("Socket Closed.");
}
```

We saw this…

Check out the recording for the discussion!

Then got a SocketException stacktrace…

SER 321

Making your Code Robust

We just threw the error to the console and quit!

Sockets/Echo_Java

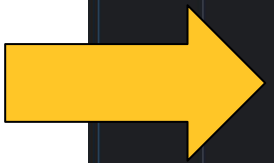
```
try {
        if (args.length != 1) {...}
        int port = -1;
        try {...} catch (NumberFormatException nfe) {...}
        Socket clientSock;
        ServerSocket sock = new ServerSocket(port);
        System.out.println("Server ready for connections");

        int bufLen = 1024;
        byte clientInput[] = new byte[bufLen]; // up to 1024 bytes in a message.
        while(true) {
                System.out.println("Server waiting for a connection");
                clientSock = sock.accept(); // blocking wait
                PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
                InputStream input = clientSock.getInputStream();
                System.out.println("Server connected to client");
                int numr = input.read(clientInput, off: 0, bufLen);
                while (numr != -1) {
                  String received = new String(clientInput, offset: 0, numr);
                  System.out.println("read from client: " + received);
                  out.println(received);
                  numr = input.read(clientInput, off: 0, bufLen);
                }
                input.close();
                clientSock.close();
                System.out.println("Socket Closed.");
        }
} catch(Exception e) {
        e.printStackTrace();
}
```

Check out the recording for the discussion!

SER 321

Making your Code Robust

What can we do to keep our server from crashing?



Sockets/Echo_Java

```
try {
        if (args.length != 1) {...}
        int port = -1;
        try {...} catch (NumberFormatException nfe) {...}
        Socket clientSock;
        ServerSocket sock = new ServerSocket(port);
        System.out.println("Server ready for connections");

        int bufLen = 1024;
        byte clientInput[] = new byte[bufLen]; // up to 1024 bytes in a message.
        while(true) {
                System.out.println("Server waiting for a connection");
                clientSock = sock.accept(); // blocking wait
                PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
                InputStream input = clientSock.getInputStream();
                System.out.println("Server connected to client");
                int numr = input.read(clientInput, off: 0, bufLen);
                while (numr != -1) {
                  String received = new String(clientInput, offset: 0, numr);
                  System.out.println("read from client: " + received);
                  out.println(received);
                  numr = input.read(clientInput, off: 0, bufLen);
                }
                input.close();
                clientSock.close();
                System.out.println("Socket Closed.");
        }
} catch(Exception e) {
        e.printStackTrace();
}
```

Check out the recording for the discussion!

Check out the recording for the discussion!

Sockets/Echo_Java

SER 321

Making your Code Robust

```
while(true) {
        System.out.println("Server waiting for a connection");
        clientSock = sock.accept(); // blocking wait
        PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
        InputStream input = clientSock.getInputStream();
        System.out.println("Server connected to client");
        int numr = input.read(clientInput, off: 0, bufLen);
        while (numr != -1) {
          String received = new String(clientInput, offset: 0, numr);
          System.out.println("read from client: " + received);
          out.println(received);
          numr = input.read(clientInput, off: 0, bufLen);
        }
        input.close();
        clientSock.close();
        System.out.println("Socket Closed.");
}
```

Check out the recording for the discussion! Sockets/Echo_Java

SER 321

Making your Code Robust

```
while (true) {
    System.out.println("Server waiting for a connection");
    clientSock = sock.accept(); // blocking wait
    PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
    InputStream input = clientSock.getInputStream();
    System.out.println("Server connected to client");
    int numr = -1;
    try {
        numr = input.read(clientInput, off: 0, bufLen);
    } catch (SocketException e) {
        System.out.println("Client disconnected.");
        break;
    }

    while (numr != -1) {
        String received = new String(clientInput, offset: 0, numr);
        System.out.println("read from client: " + received);
        out.println(received);
        numr = input.read(clientInput, off: 0, bufLen);
    }
    input.close();
    clientSock.close();
    System.out.println("Socket Closed.");
}
```

Check out the recording for the discussion! Sockets/Echo_Java

SER 321

Making your Code Robust

What other things can we do to keep our server running?

Error Handling!

```
while (true) {
    System.out.println("Server waiting for a connection");
    clientSock = sock.accept(); // blocking wait
    PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
    InputStream input = clientSock.getInputStream();
    System.out.println("Server connected to client");
    int numr = -1;
    try {
        numr = input.read(clientInput, off: 0, bufLen);
    } catch (SocketException e) {
        System.out.println("Client disconnected.");
        break;
    }
```

Sockets/SimpleProtocolWithSomeErrorHandling

```
    while (numr != -1) {
        String received = new String(clientInput, offset: 0, numr);
        System.out.println("read from client: " + received);
        out.println(received);
        numr = input.read(clientInput, off: 0, bufLen);
    }
    input.close();
    clientSock.close();
    System.out.println("Socket Closed.");
}
```

Sockets/SimpleProtocolWithSomeErrorHandling

SER 321

Making your Code Robust

```
static JSONObject testField(JSONObject req, String key, String type){
  JSONObject res = new JSONObject();
  // field does not exist
  if (!req.has(key)){
    res.put("ok", false);
    res.put("message", "Field " + key +
            " does not exist in request");
    return res;
  }
  System.out.println(req.get(key).getClass().getName());
  // field does not have correct type
  if (!req.get(key).getClass().getName().equals(type)){
    res.put("message", "Field " + key +
            " needs to be of type: " + type);
    res.put("ok", false);
    return res.put("ok", false);
  } else {
    return res.put("ok", true);
  }
}
```

```
while (true){
  System.out.println("Server waiting for a connection");
  sock = serv.accept(); // blocking wait
  in = new ObjectInputStream(sock.getInputStream());
  OutputStream out = sock.getOutputStream();
  os = new DataOutputStream(out);
  String s = (String) in.readObject();
  JSONObject req = new JSONObject(s);

  JSONObject res =
          testField(req, key: "type", type: "java.lang.String");
  if (!res.getBoolean( key: "ok")) {
    overandout(res);
    continue;
  }
  // check which request it is (could also be a switch statement)
  if (req.getString( key: "type").equals("echo")) {
    res = echo(req);
  } else if (req.getString( key: "type").equals("add")) {
    res = add(req);
  } else if (req.getString( key: "type").equals("addmany")) {
    res = addmany(req);
  } else {
    res = wrongType(req);
  }
  overandout(res);
}
```

Check out the recording for the discussion!

Sockets/SimpleProtocolWithSomeErrorHandling

SER 321

Making your Code Robust

```
while (true){
  System.out.println("Server waiting for a connection");
  sock = serv.accept(); // blocking wait
  in = new ObjectInputStream(sock.getInputStream());
  OutputStream out = sock.getOutputStream();
  os = new DataOutputStream(out);
  String s = (String) in.readObject();
```

```
static JSONObject testField(JSONObject req, String key, String type){
  JSONObject res = new JSONObject(
  // field does not exist
  if (!req.has(key)){
    res.put("ok", false);
    res.put("message", "Field " +
            " does not exist in re
    return res;
  }
  System.out.println(req.get(key).getClass().getName());
  // field does not have correct type
  if (!req.get(key).getClass().getName().equals(type)){
    res.put("message", "Field " + key +
            " needs to be of type: " + type);
    res.put("ok", false);
    return res.put("ok", false);
  } else {
    return res.put("ok", true);
  }
}
```

```
static JSONObject wrongType(JSONObject req){  1 usage
  JSONObject res = new JSONObject();
  res.put("ok", false);
  res.put("message", "Type " + req.getString( key: "type") + " not supported.");
  return res;
}
```

Check out the recording for the discussion!

```
  // check which request it is (could also be a switch statement)
  if (req.getString( key: "type").equals("echo")) {
    res = echo(req);
  } else if (req.getString( key: "type").equals("add")) {
    res = add(req);
  } else if (req.getString( key: "type").equals("addmany")) {
    res = addmany(req);
  } else {
    res = wrongType(req);
  }
  overandout(res);
}
```

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Thursday, October 31st 2024 at 7:00 pm MST
- Sunday, November 3rd 2024 at 7:00 pm MST
- Tuesday, November 5th 2024 at 10:00 am MST

Review Sessions:

- TBD

Don’t forget about Daylight Savings!

Questions?

Survey:

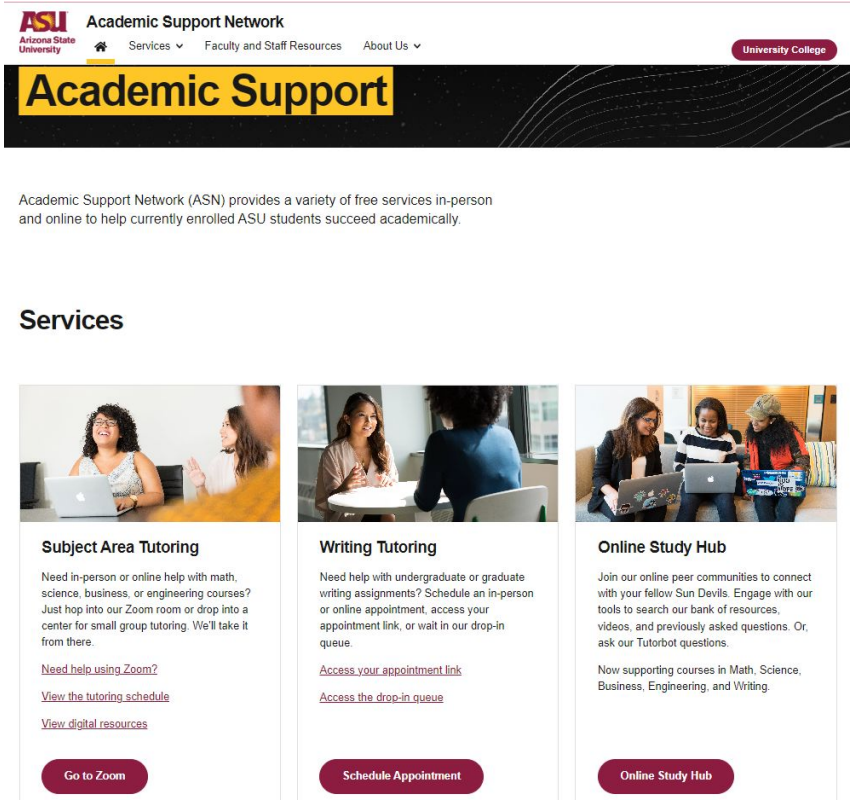
https://asuasn.info/ASNSurvey

25

More Questions?

Check out our other resources!

tutoring.asu.edu

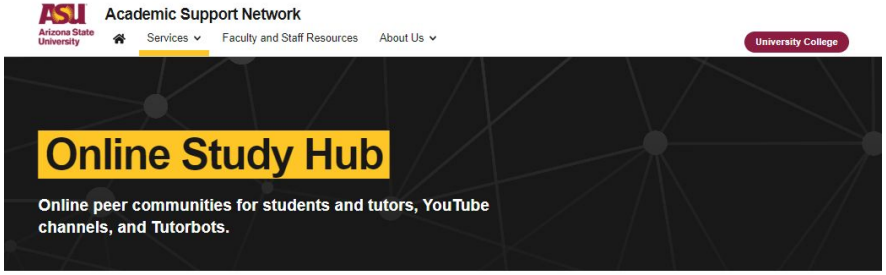


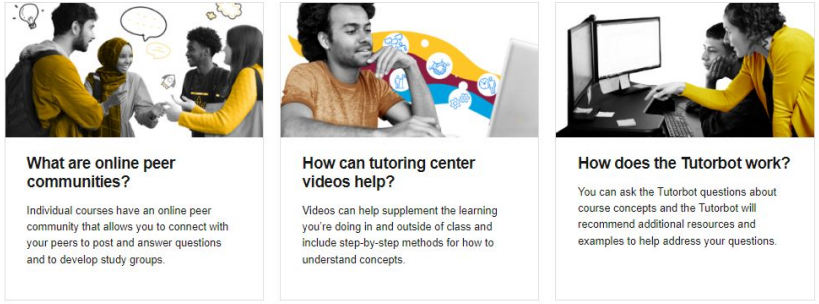
1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

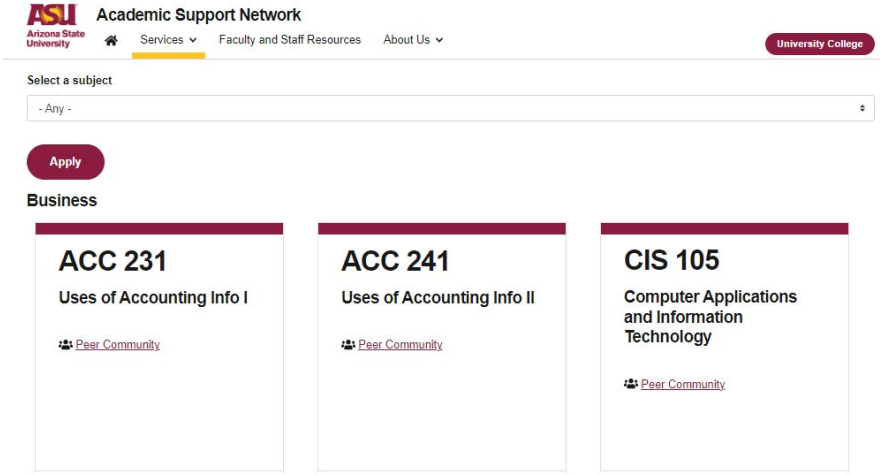
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



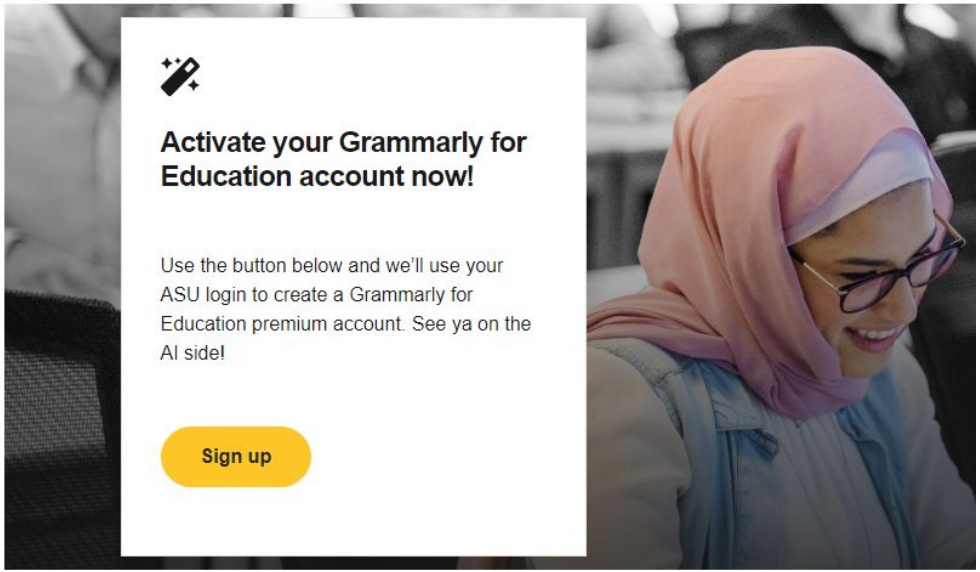




Don't forget to check out the Online Study Hub for additional resources!

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