SER 321 B Session

SI Session

Tuesday, April 1st 2025

10:00 am - 11:00 am MST

Agenda

- Connections
- JSON Recognition
- TCP v. UDP Matching
- Making your Code Robust
- Sockets & Client-Server Intro

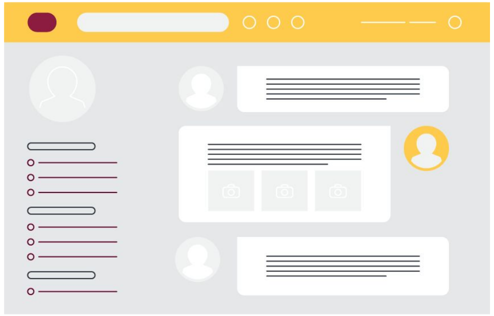
SI Session Expectations

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!

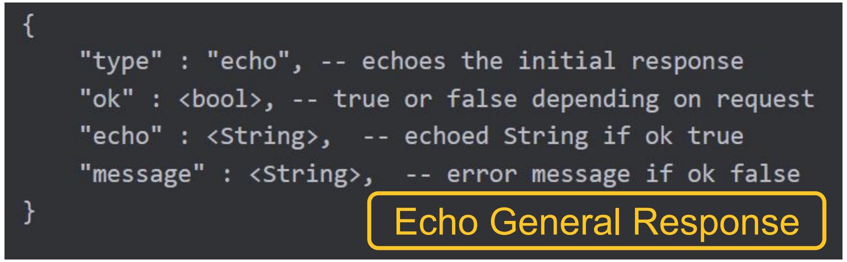
Check out the recording for the discussion!

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>
}
```

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

Check out the recording for the discussion and solution!

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

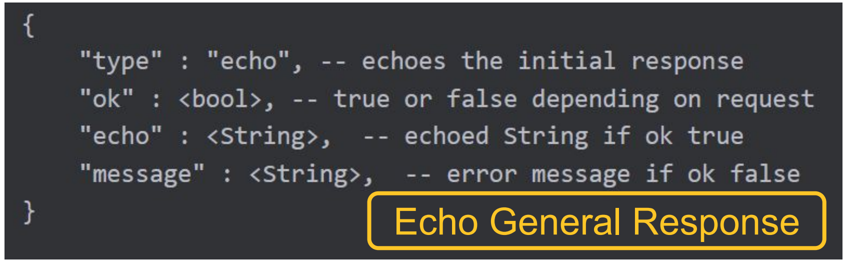
D.
```
{
    “type” : “echo”,
    “ok” : true,
    “echo” : <String>
}
```

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>
}
```

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

Check out the recording for the discussion and solution!

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

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

SER 321

TCP vs. UDP Matching

Unreliable

Check out the recording for the discussion and solution!

TCP

OR

UDP

SER 321

TCP vs. UDP Matching

Connection-Oriented

Check out the recording for the discussion and solution!

TCP

OR

UDP

SER 321

TCP vs. UDP Matching

Uses Streams

Check out the recording for the discussion and solution!

TCP

OR

UDP

SER 321
TCP vs. UDP Matching

Has Less Overhead

Check out the recording for the discussion and solution!

TCP

OR

UDP

SER 321

TCP vs. UDP Matching

Has Less Overhead

Check out the recording for the discussion and solution!

TCP	OR	UDP
Reliable		Unreliable
Connection-Oriented		Connectionless
Uses Streams		Uses Datagrams
Has More Overhead		Has Less Overhead

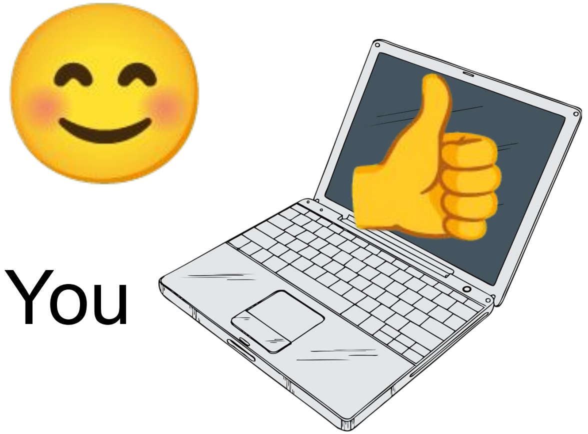
What do we mean when we say “make sure your code is robust”?

Error Handling

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”?



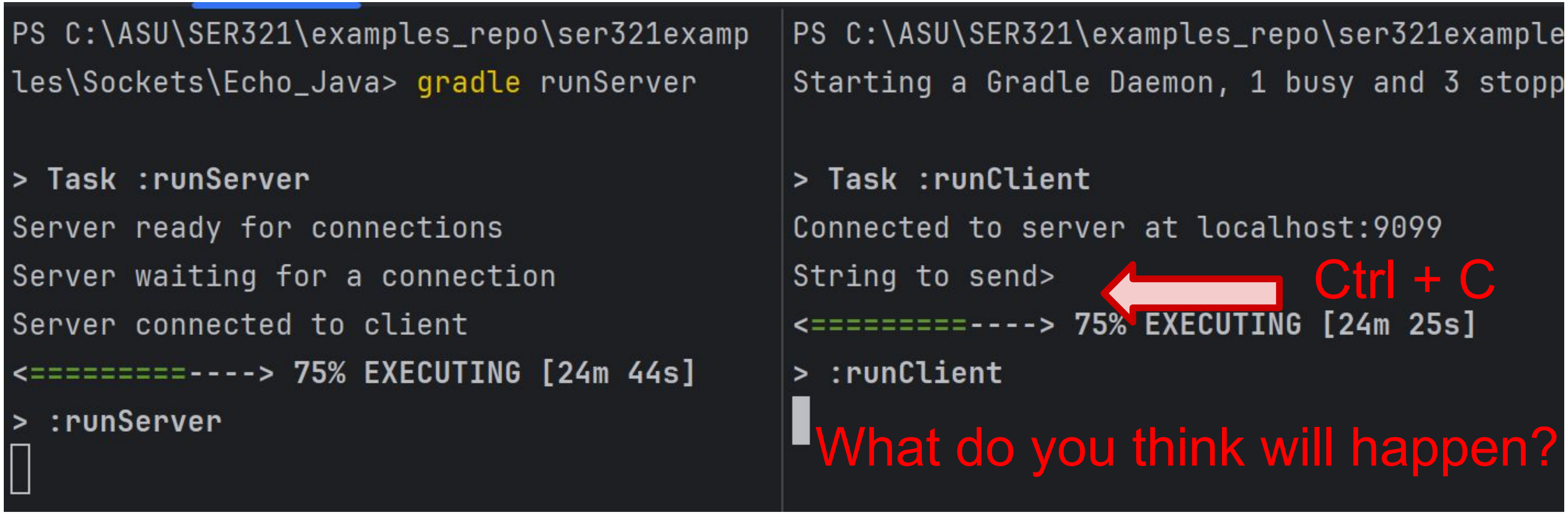
```
An error has occurred. To continue:
Press Enter to return to Windows, or
Press CTRL+ALT+DEL to restart your computer. If you do this,
you will lose any unsaved information in all open applications.
Error: 0E : 016F : BFF9B3D4
        Press any key to continue _
```

Sockets/Echo_Java

SER 321
Making your Code Robust

Check out the recording for the discussion!

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
```

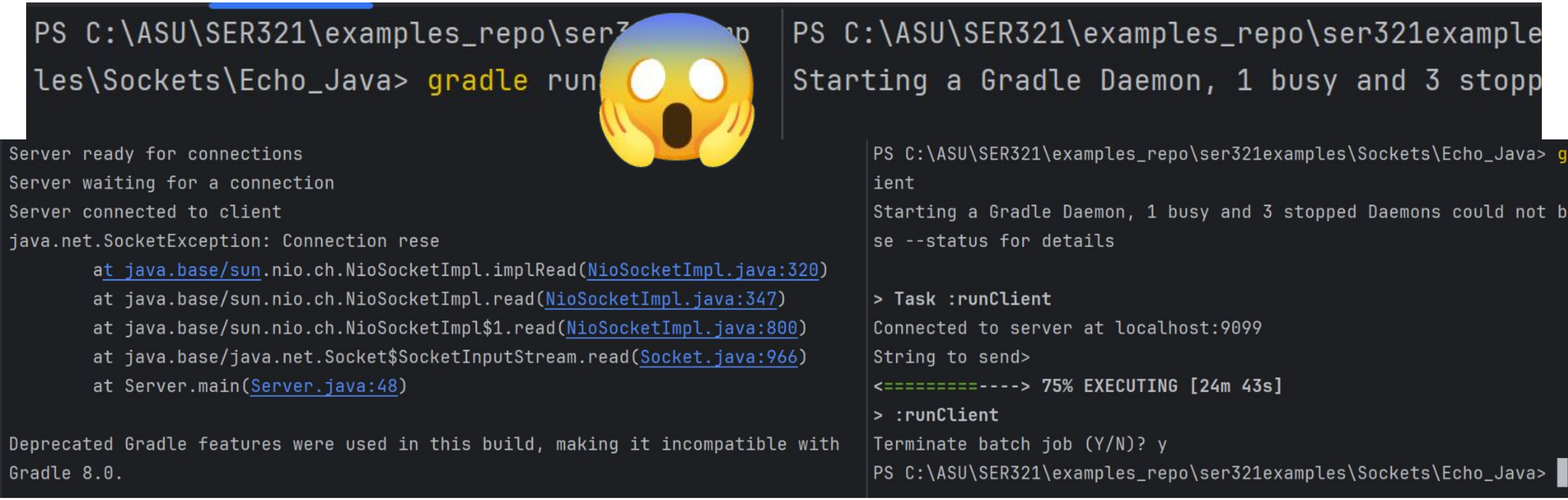
Sockets/Echo_Java

SER 321

Making your Code Robust

Check out the recording for the discussion!

We crashed the server!



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

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

```
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.
```

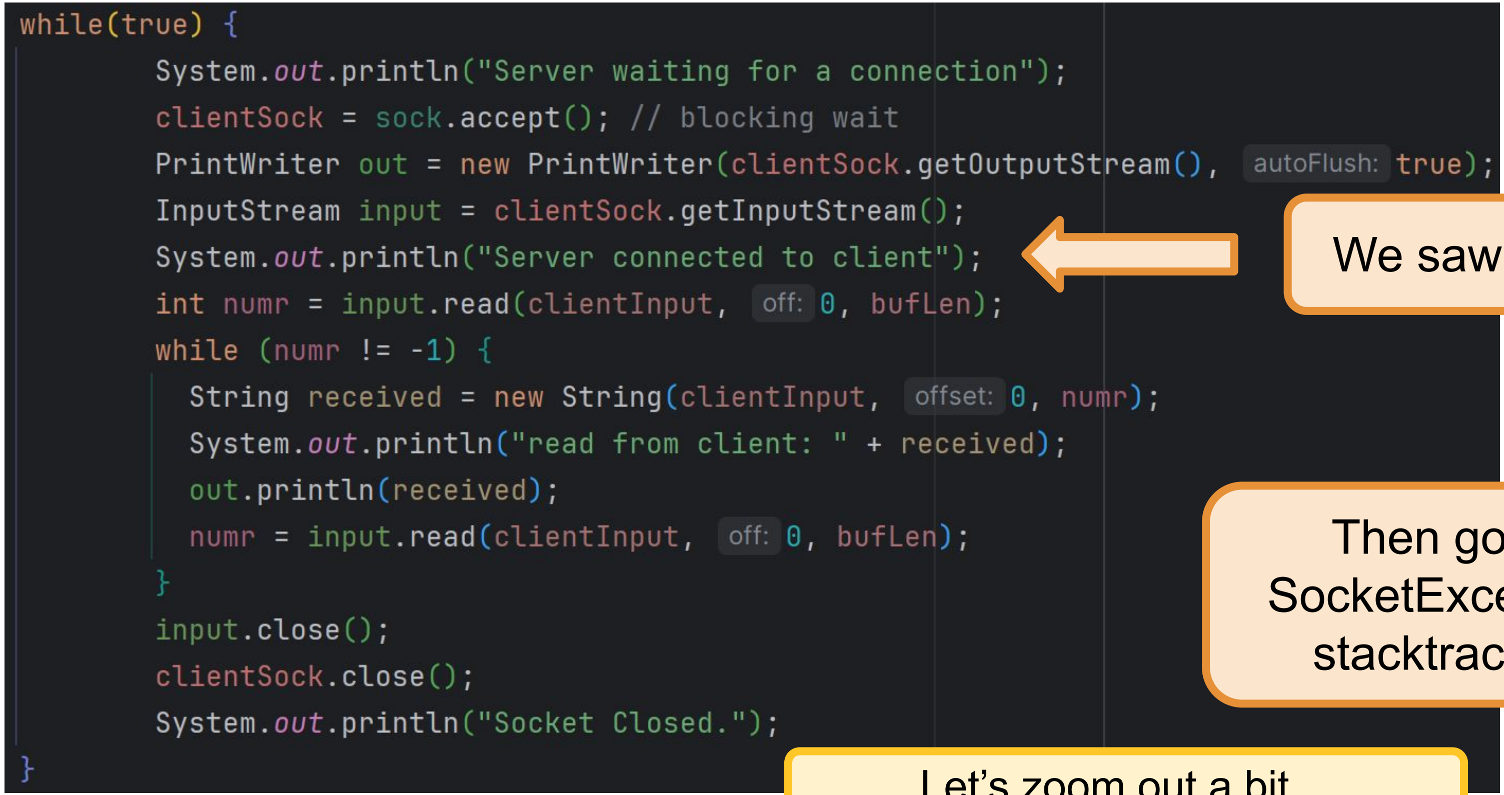
```
PS C:\ASU\SER321\examples_repo\ser321examples\Sockets\Echo_Java> g
ient
Starting a Gradle Daemon, 1 busy and 3 stopped Daemons could not b
se --status for details

> 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…

Then got a SocketException stacktrace…

Let's zoom out a bit…

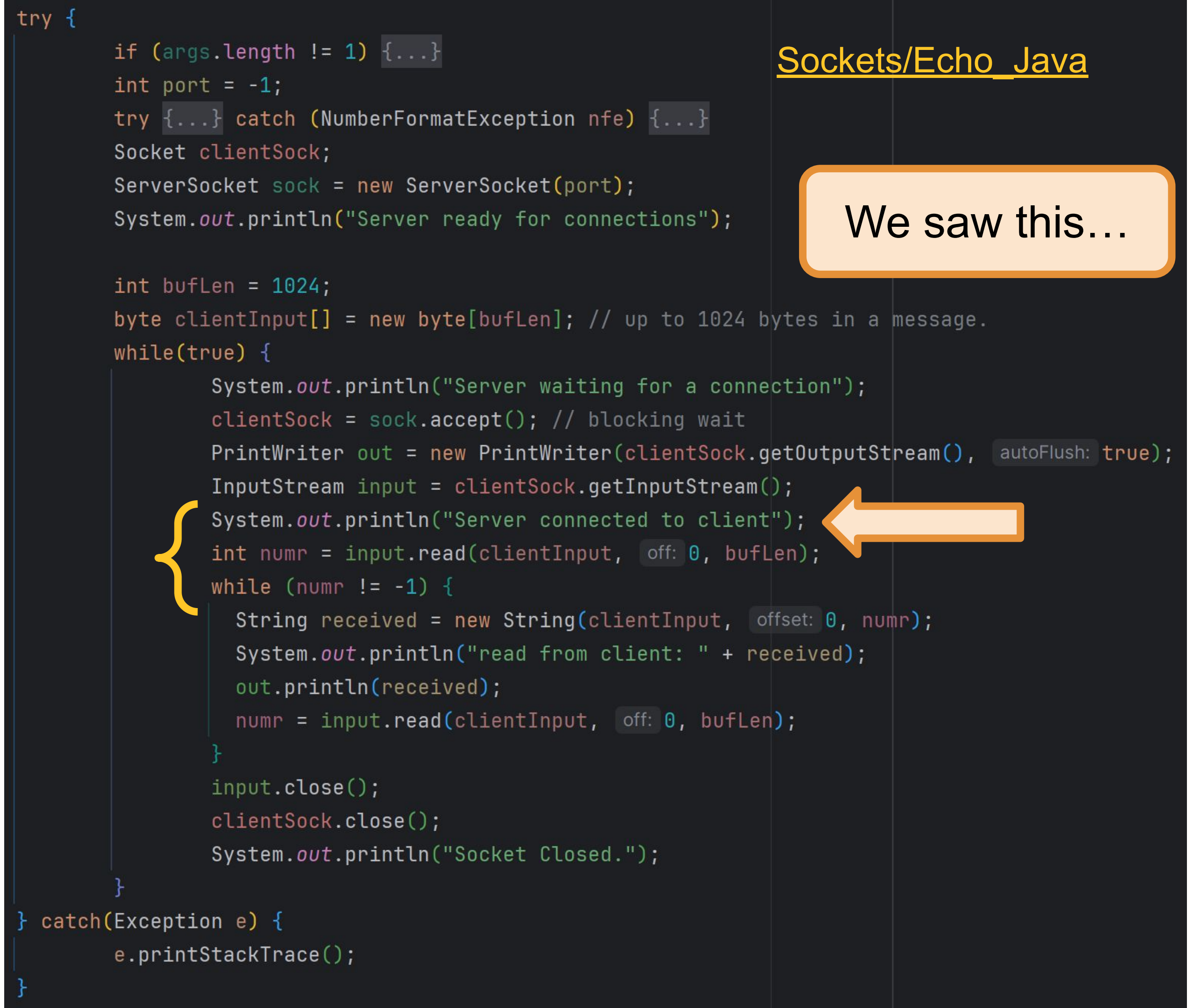
SER 321
Making your Code Robust

This *assumes* we read from the stream with no problems

If we have a problem, we just throw the error to the console and quit!

Sockets/Echo_Java

We saw this…



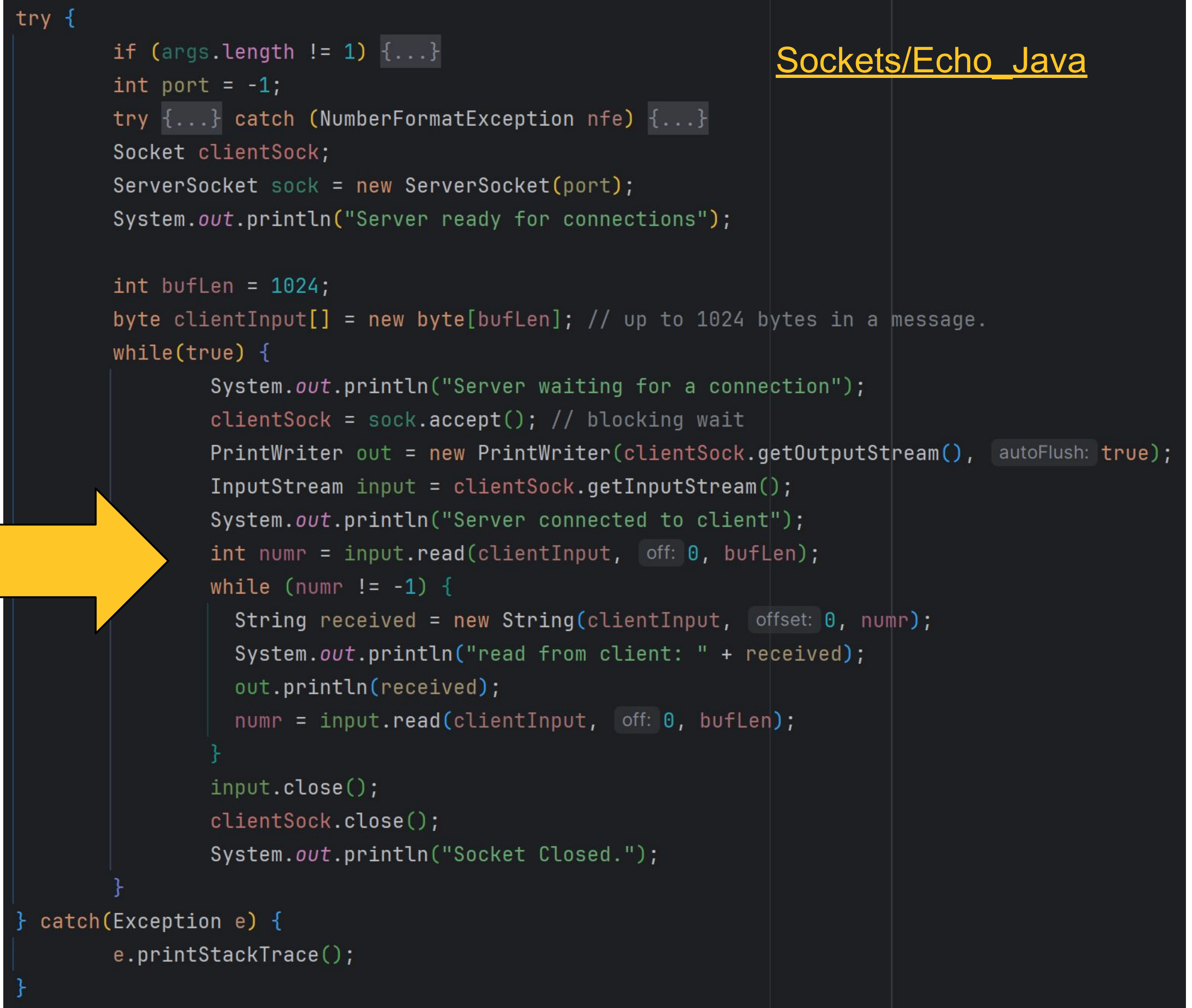
```
try {
        if (args.length != 1) {...}
        int port = -1;
        try {...} catch (NumberFormatException nfe) {...}
        Socket clientSock;
        ServerSocket sock = new ServerSocket(port);
        System.out.println("Server ready for connections");

        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();
}
```

SER 321

Making your Code Robust

What can we do to keep our server from crashing?

Error Handling!



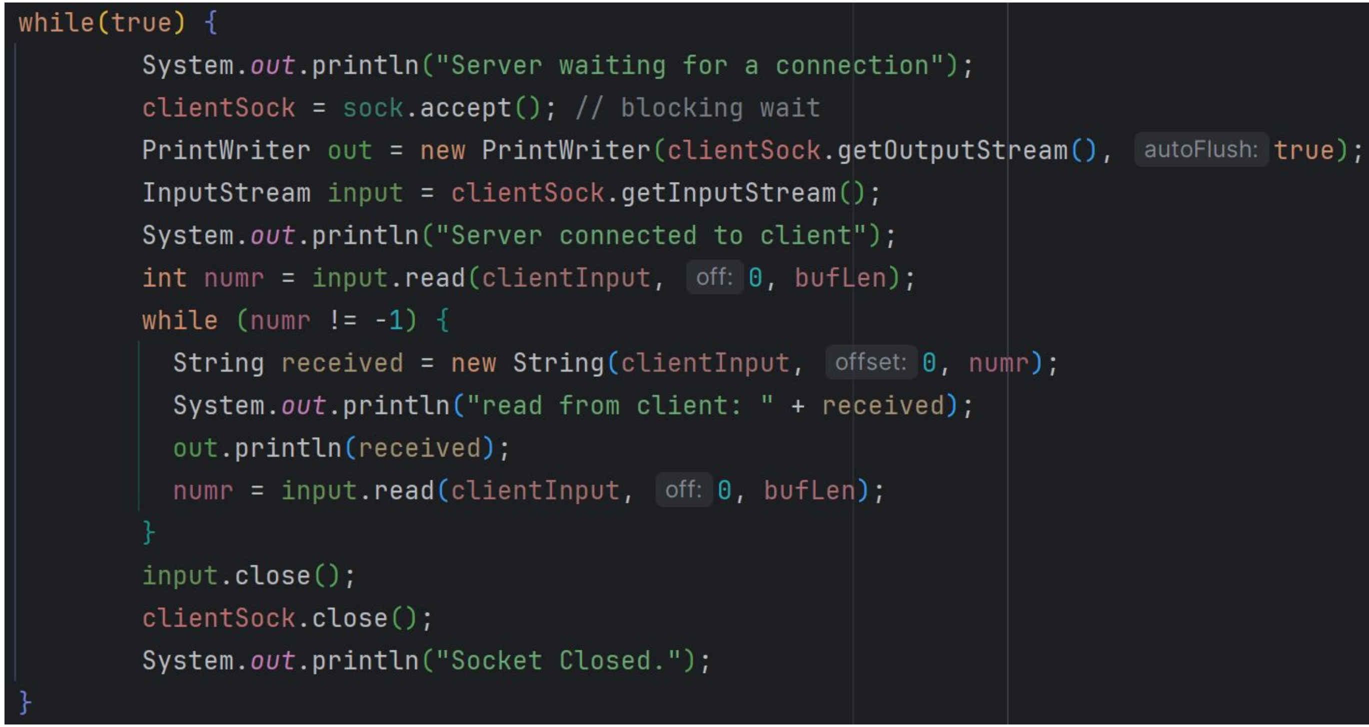
Sockets/Echo_Java

```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();
}
```

Sockets/Echo_Java

SER 321
Making your Code Robust

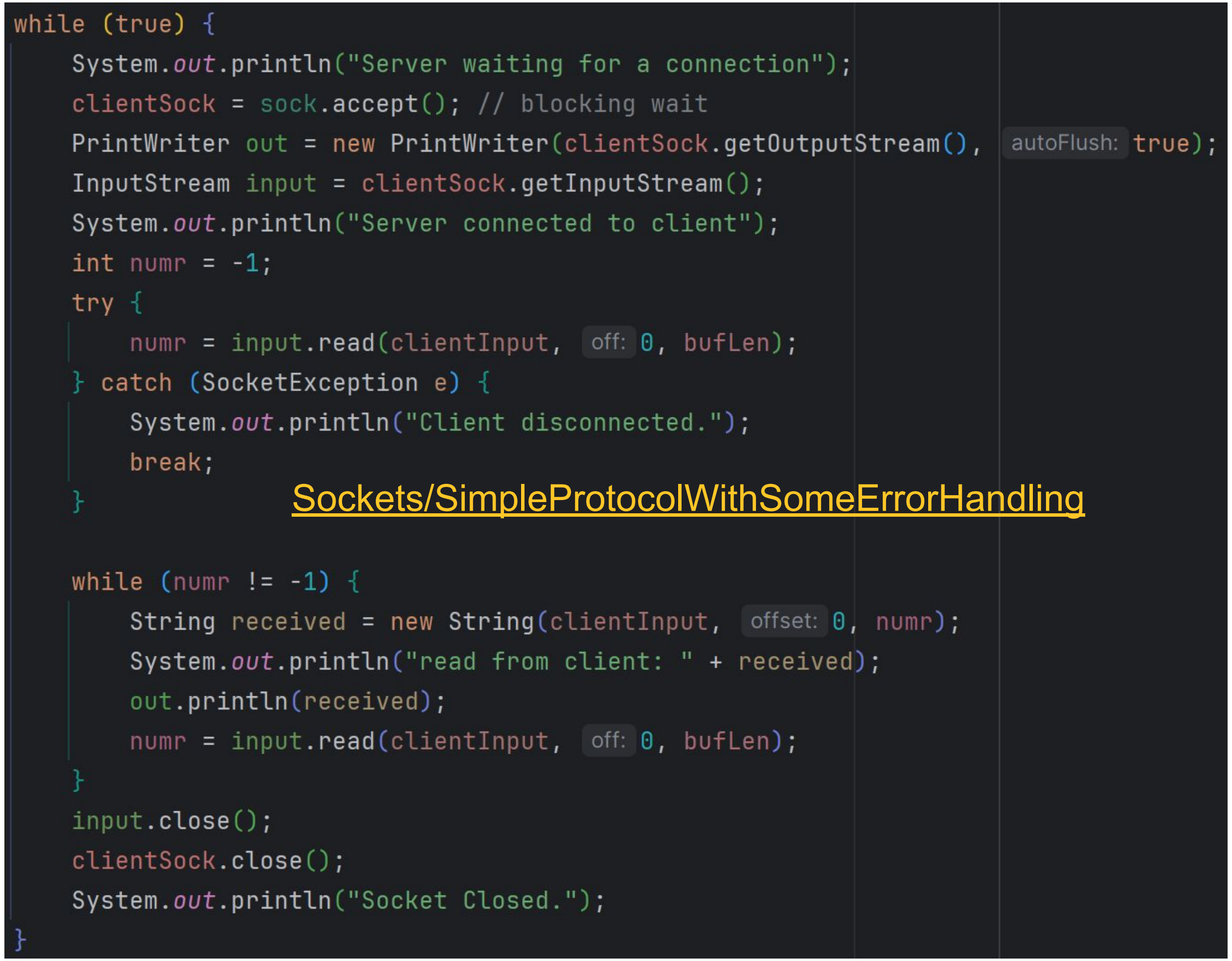
Check out the recording for the discussion!



```
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.");
}
```

Sockets/Echo_Java

SER 321
Making your Code Robust



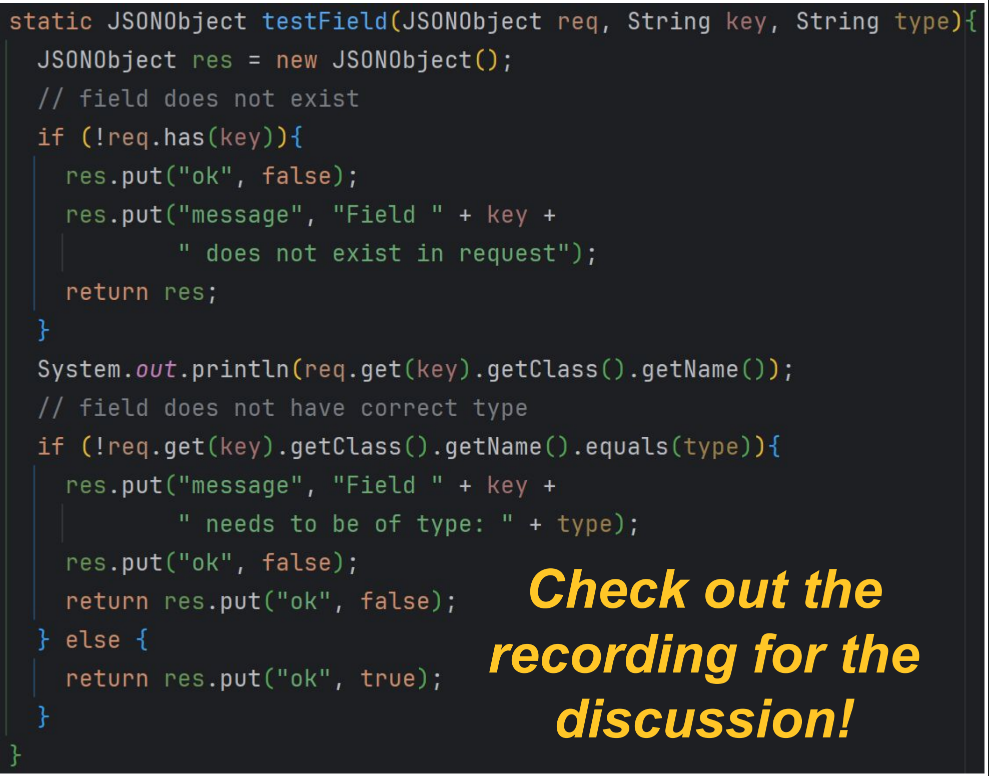
```
while (true) {
    System.out.println("Server waiting for a connection");
    clientSock = sock.accept(); // blocking wait
    PrintWriter out = new PrintWriter(clientSock.getOutputStream(), autoFlush: true);
    InputStream input = clientSock.getInputStream();
    System.out.println("Server connected 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.");
}
```

Sockets/SimpleProtocolWithSomeErrorHandling

Sockets/SimpleProtocolWithSomeErrorHandling

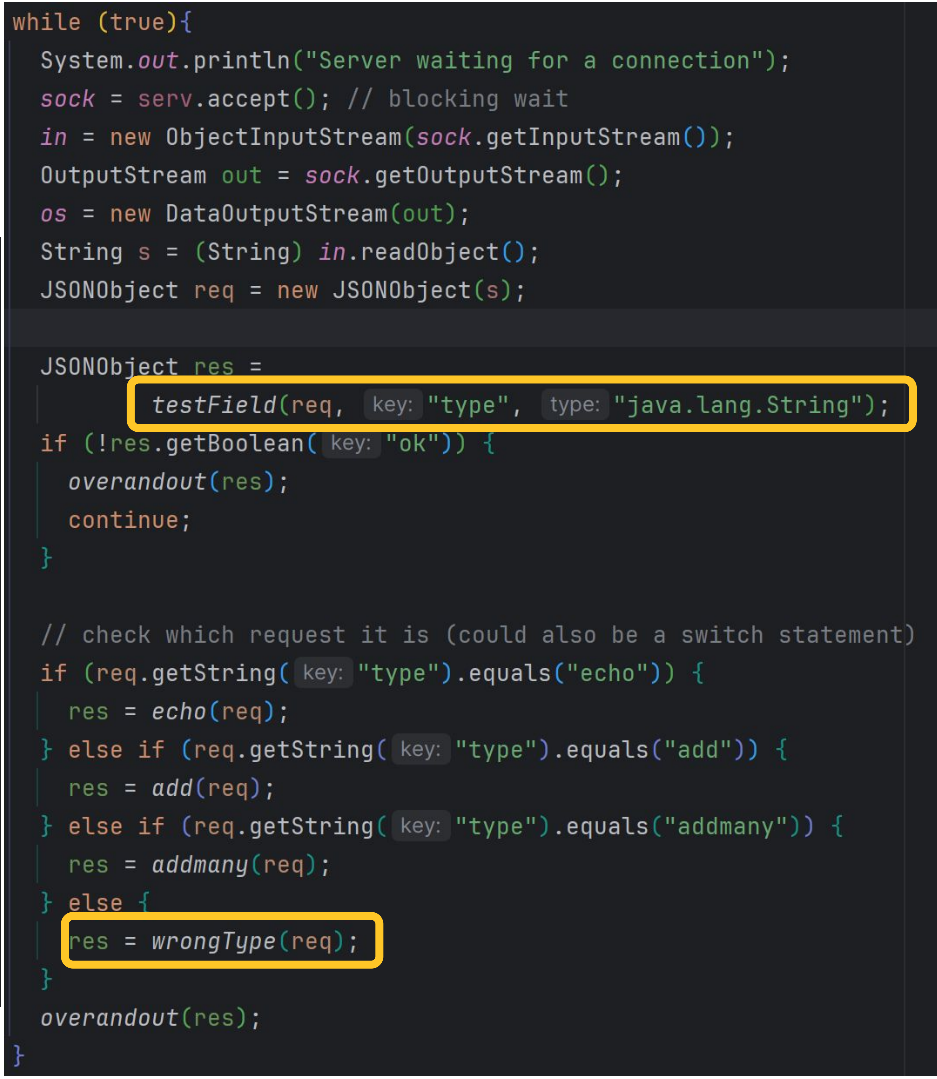
SER 321

Making your Code Robust



```java
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);
  }
}
```

Check out the recording for the discussion!

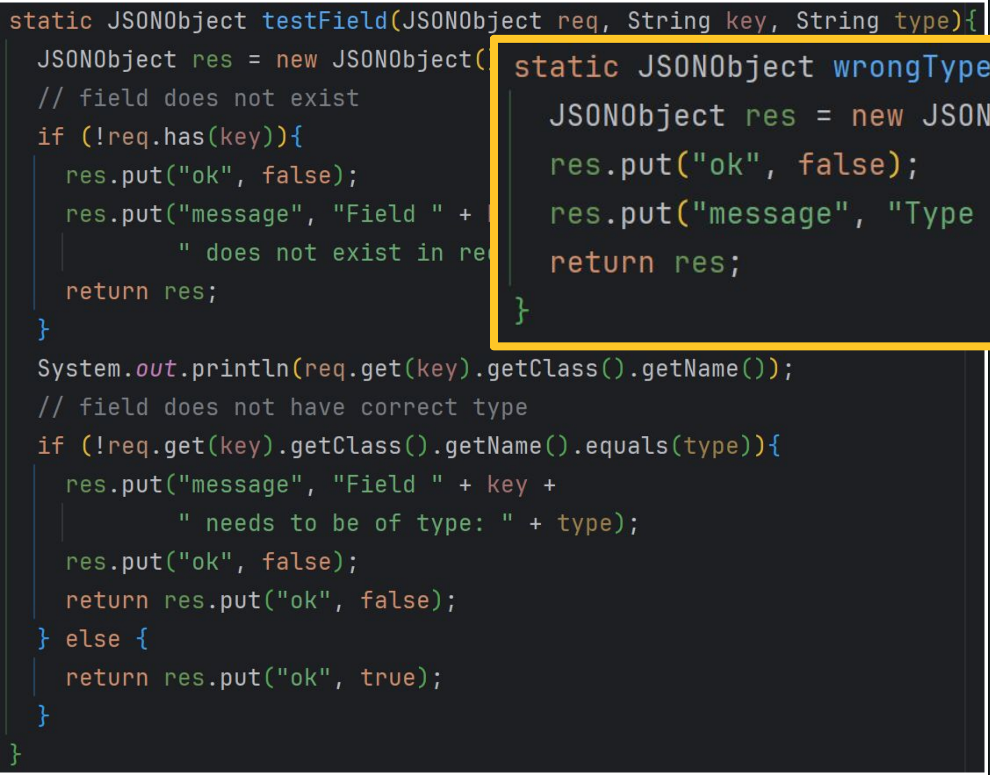


```java
while (true){
  System.out.println("Server waiting for a connection");
  sock = serv.accept(); // blocking wait
  in = new ObjectInputStream(sock.getInputStream());
  OutputStream out = sock.getOutputStream();
  os = new DataOutputStream(out);
  String s = (String) in.readObject();
  JSONObject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

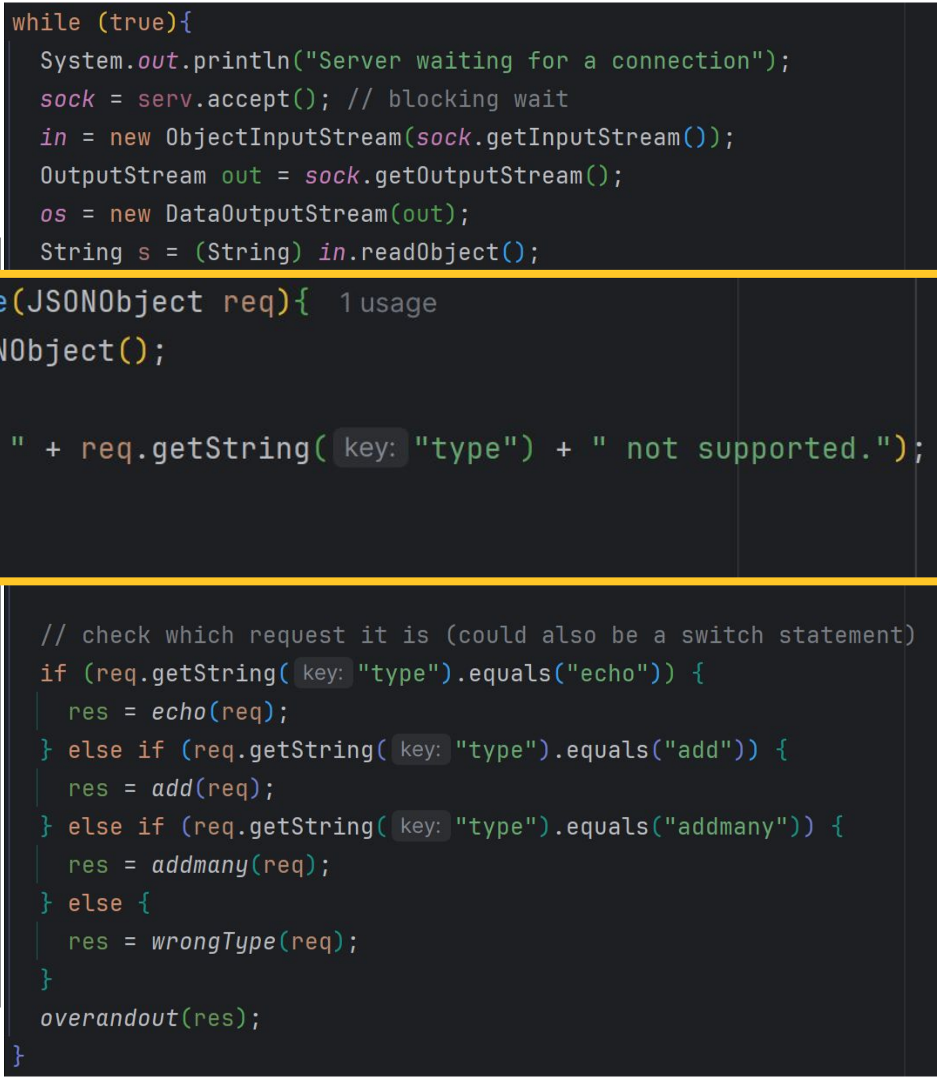
Sockets/SimpleProtocolWithSomeErrorHandling

SER 321

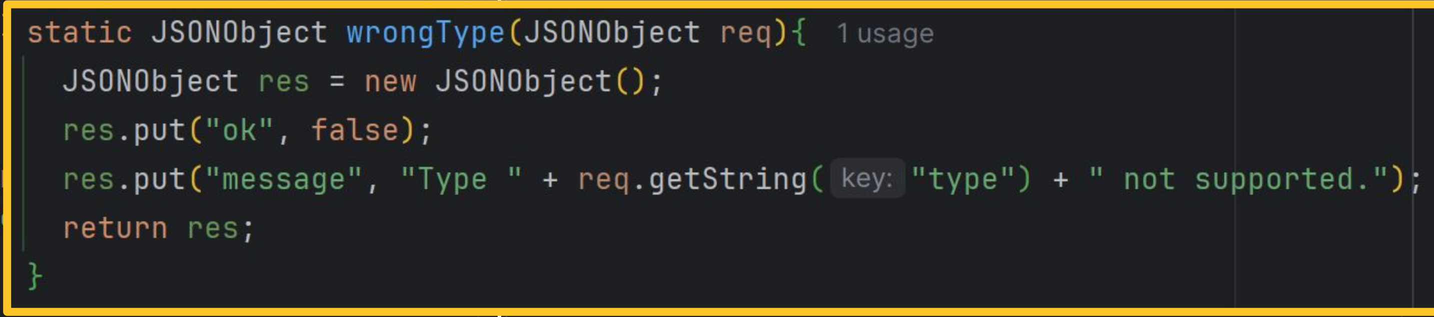
Making your Code Robust



```java
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);
  }
}
```



```java
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();

  // 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);
}
```



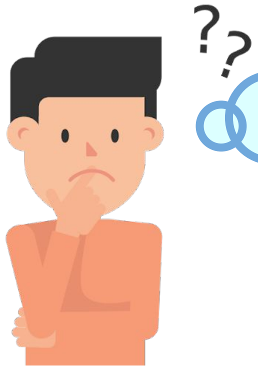
```java
static JSONObject wrongType(JSONObject req){  1 usage
  JSONObject res = new JSONObject();
  res.put("ok", false);
  res.put("message", "Type " + req.getString( key: "type") + " not supported.");
  return res;
}
```

SER 321

Sockets!

What do we need for a client/server connection?

Check out the recording for the discussion!

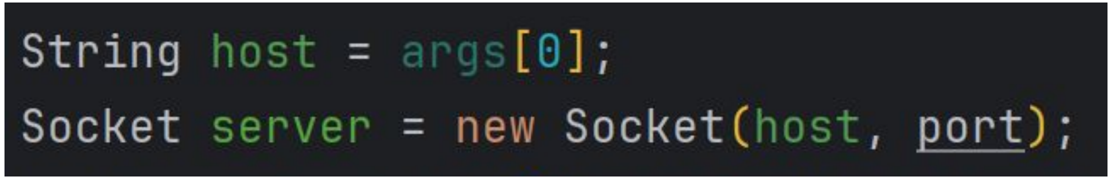


Sockets/Echo_Java

SER 321
Client/Server

Think Fast - Client or Server?

Check out the recording for the discussion and solution!

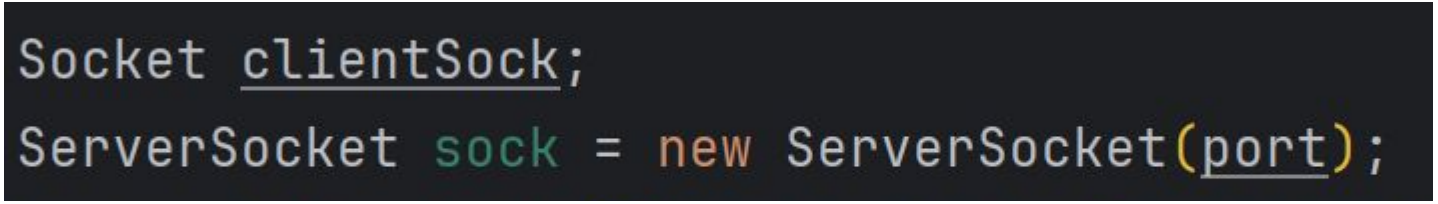


```
String host = args[0];
Socket server = new Socket(host, port);
```

SER 321
Client/Server

Think Fast - Client or Server?

Check out the recording for the discussion and solution!

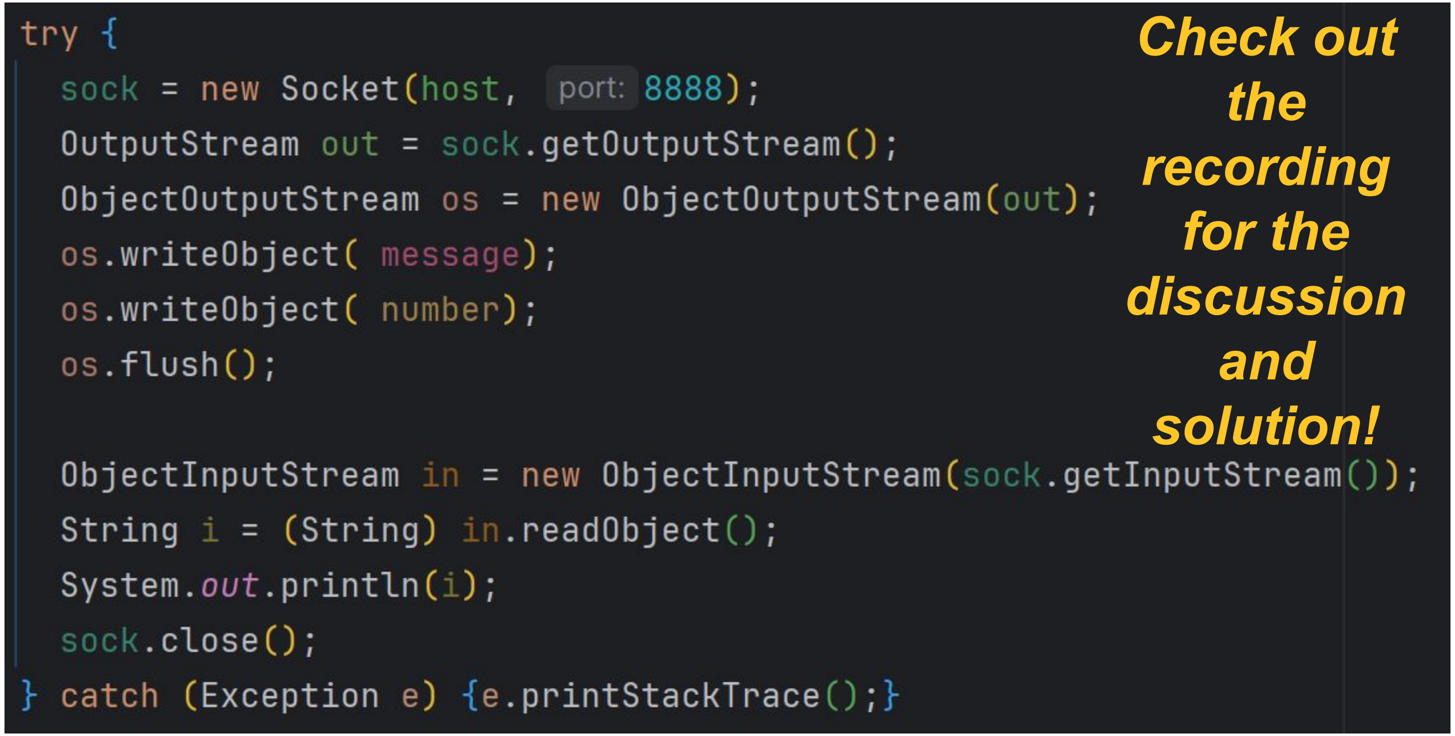


```
Socket clientSock;
ServerSocket sock = new ServerSocket(port);
```

Sockets/JavaSimpleSock2

SER 321
Client/Server

Think Fast - Client or Server?



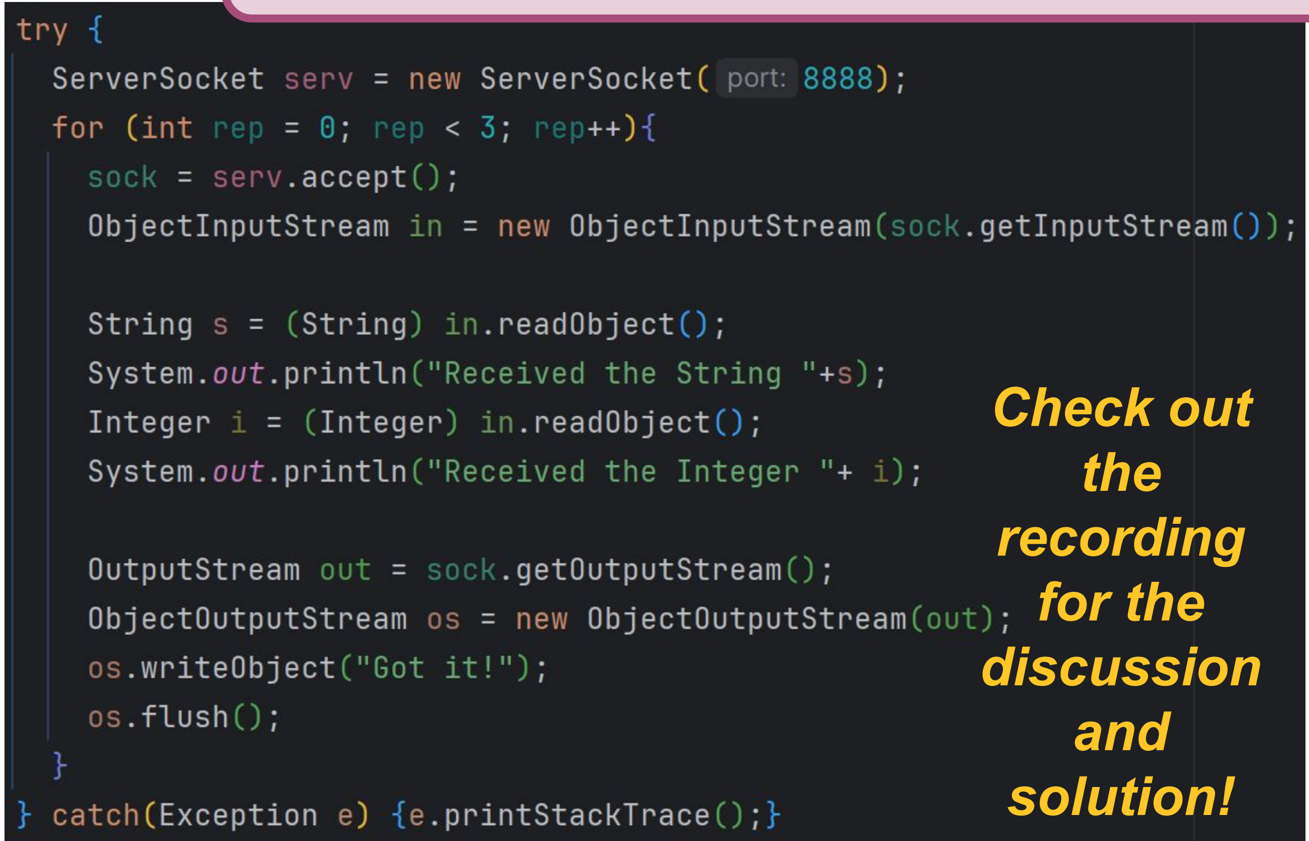
```
try {
    sock = new Socket(host, port: 8888);
    OutputStream out = sock.getOutputStream();
    ObjectOutputStream os = new ObjectOutputStream(out);
    os.writeObject( message);
    os.writeObject( number);
    os.flush();

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

Check out the recording for the discussion and solution!

Sockets/JavaSimpleSock2

SER 321
Client/Server

Think Fast - Client or Server?



```java
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);

      OutputStream out = sock.getOutputStream();
      ObjectOutputStream os = new ObjectOutputStream(out);
      os.writeObject("Got it!");
      os.flush();
    }
} catch(Exception e) {e.printStackTrace();}
```

Check out the recording for the discussion and solution!

SER 321

Scratch Space

Upcoming Events

SI Sessions:

- Thursday, April 3rd at 7:00 pm MST
- Sunday, April 6th at 7:00 pm MST
- Tuesday, April 8th at 10:00 am 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:

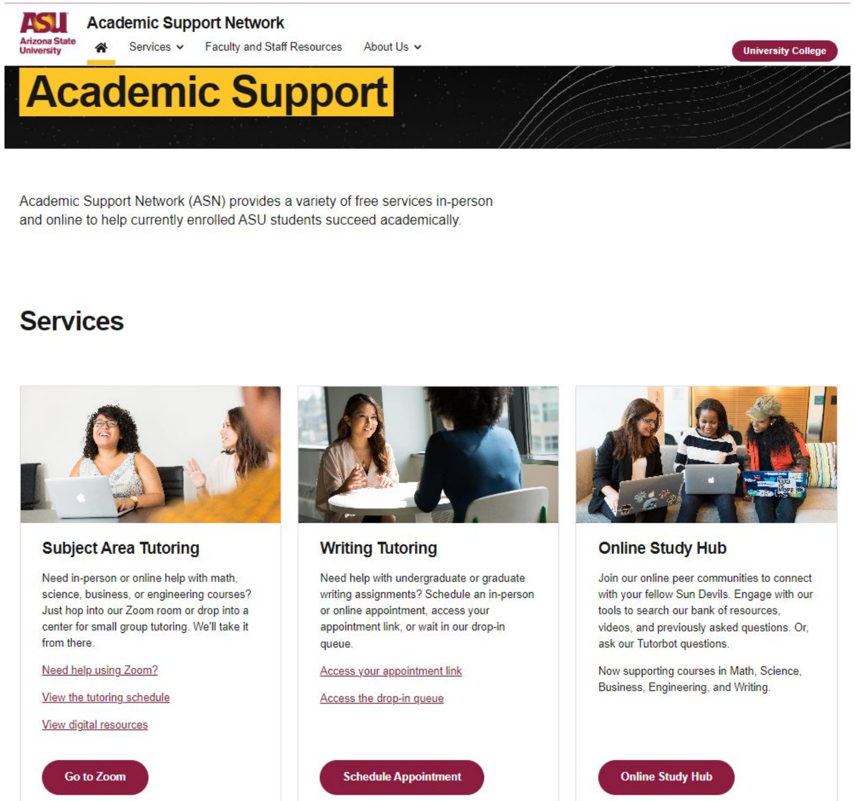
https://asuasn.info/ASNSurvey

bitly

More Questions?

Check out our other resources!

tutoring.asu.edu



ASU Arizona State University

Academic Support Network

Services · Faculty and Staff Resources · About Us · University College

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

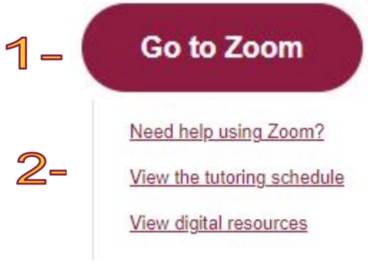
Schedule Appointment

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

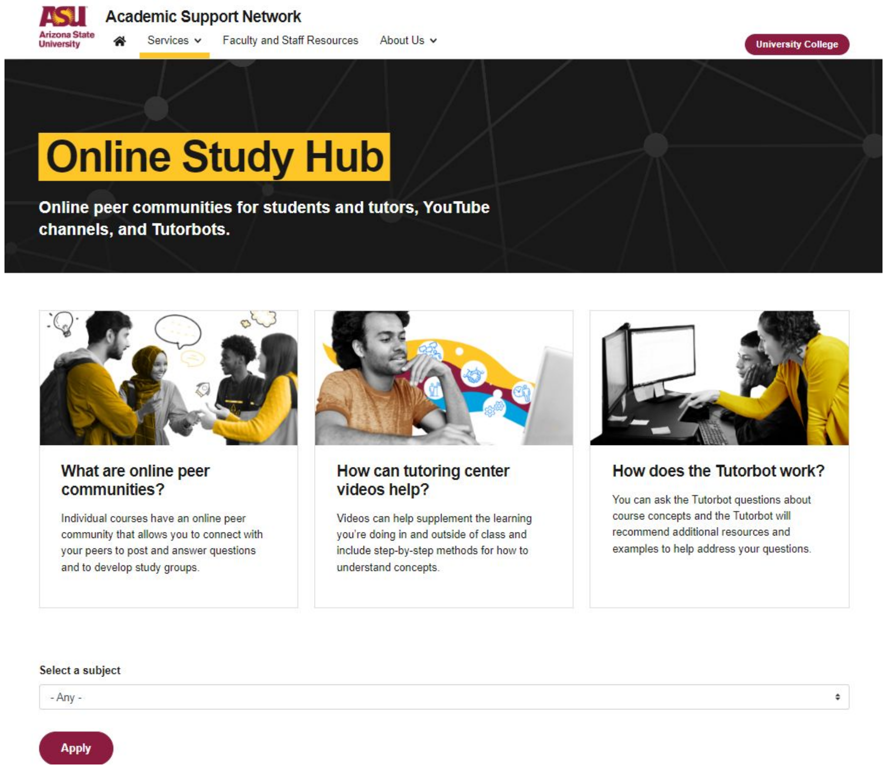


1. **Click on 'Go to Zoom' to log onto our Online Tutoring Center.**
2. **Click on 'View the tutoring schedule' to see when tutors are available for specific courses.**

More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



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.

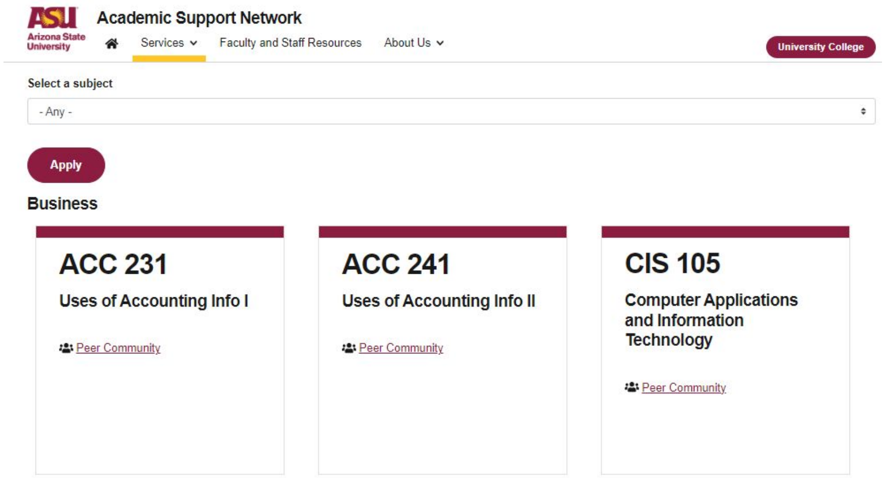
How can tutoring center videos help?

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 -



Select a subject: - Any -

Business

ACC 231 Uses of Accounting Info I — Peer Community

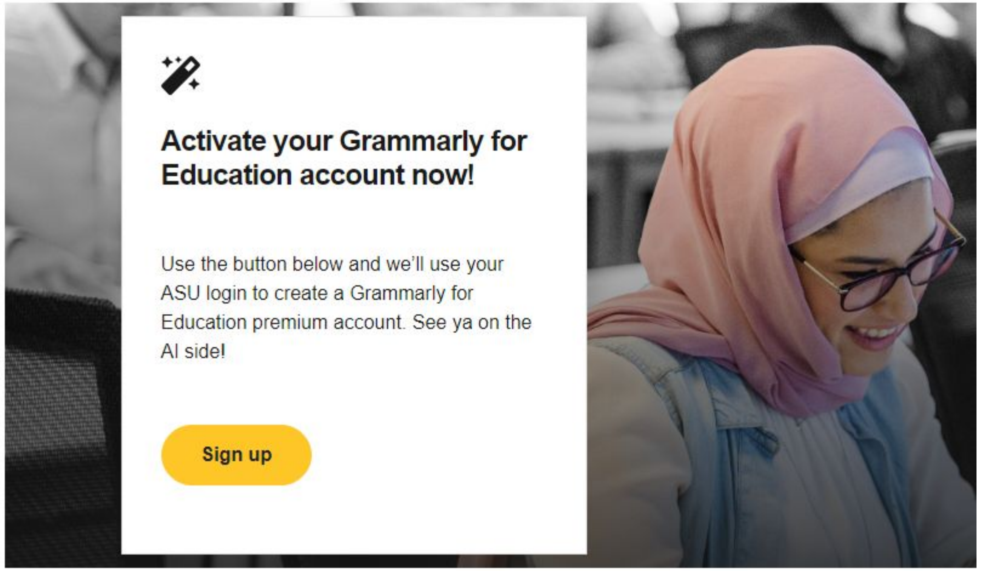
ACC 241 Uses of Accounting Info II — Peer Community

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!



Activate your Grammarly for Education account now!

Use the button below and we'll use your ASU login to create a Grammarly for Education premium account. See ya on the AI side!

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