

SER 321 C Session

SI Session

Monday, June 17th 2024

6:00 pm - 7:00 pm MST

Agenda



Protobuf PSA & Demo

Threads!

Single vs. Multi-Threaded Servers

Why We Care

Threading your Code

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

Quick PSA for Protobufs!

If you use IntelliJ, you must do this first!!

IntelliJ users!

- `gradle generateProto` will generate the protocol buffer files but may not be recognized by IntelliJ because it may not be able to locate them. Add this block of code into your `build.gradle` file →

```
sourceSets {  
    main {  
        java {  
            srcDirs 'build/generated/source/proto/main/java/proto'  
        }  
    }  
}
```

The file path needs to point to the generated Protobuf files so make sure the file path you are declaring in the `sourceSets` block matches the actual path to your generated Protobuf files for your project

- In the protobuf files, the `java_package` declaration determines the package name for the generated files and `java_outer_classname` determines this Protobuf's generated file name → in the given starter code's Protobuf files the package name is declared as "proto" so generated Protobuf files will be placed in a package called "proto" (under `build/generated/source/proto/main/java`)

4-2 Starter Code

SER 321

Protobufs

Step 1

Project structure:

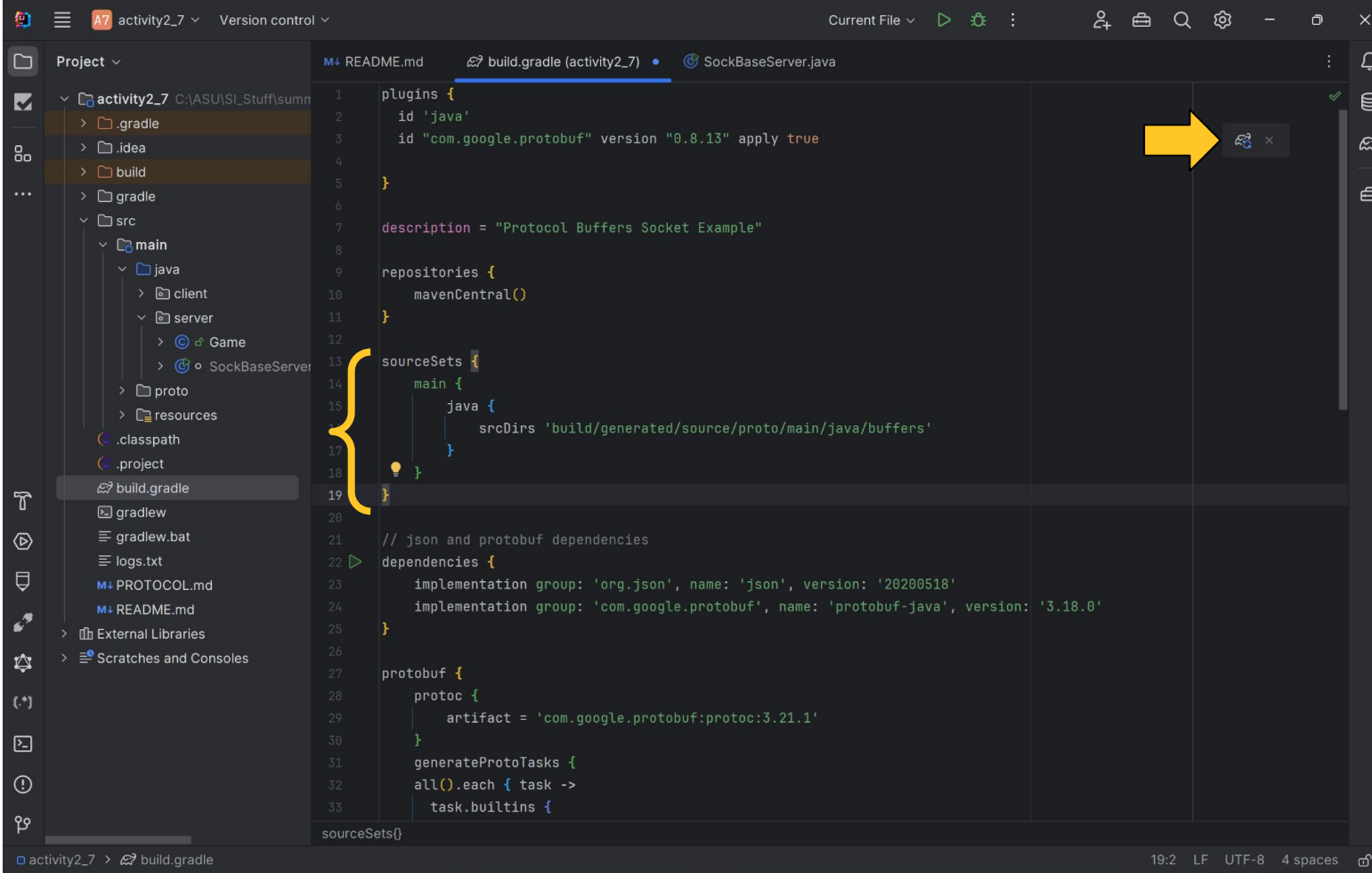
- activity2_7
 - .gradle
 - .idea
 - build
 - gradle
 - src
 - main
 - java
 - client
 - server
 - Game
 - SockBaseServer
 - proto
 - resources
 - .classpath
 - .project
 - build.gradle
 - gradlew
 - gradlew.bat
 - logs.txt
 - PROTOCOL.md
 - README.md
 - External Libraries
 - Scratches and Consoles

build.gradle (activity2_7):

```
1 plugins {  
2     id 'java'  
3     id "com.google.protobuf" version "0.8.13" apply true  
4 }  
5  
6  
7 description = "Protocol Buffers Socket Example"  
8  
9 repositories {  
10     mavenCentral()  
11 }  
12  
13 // json and protobuf dependencies  
14 dependencies {  
15     implementation group: 'org.json', name: 'json', version: '20200518'  
16     implementation group: 'com.google.protobuf', name: 'protobuf-java', version: '3.18.0'  
17 }  
18  
19 protobuf {  
20     protoc {  
21         artifact = 'com.google.protobuf:protoc:3.21.1'  
22     }  
23     generateProtoTasks {  
24         all().each { task ->  
25             task.builtins {  
26                 }  
27         }  
28     }  
29 }  
30 def host :String = 'localhost'  
31 def port :Integer = 9099  
32  
33 // task reading in the arguments if any given, if not the default from above will be used
```

SER 321
Protobufs

Step 1



4-2 Starter Code

SER 321

Protobufs

Step 2

The screenshot shows an IDE with the following components:

- Project View:** Shows the project structure with folders like `activity2_7`, `src`, `main`, `java`, `server`, and `Game`. The `SockBaseServer` class is highlighted in the `server` folder.
- Code Editor:** Displays the `SockBaseServer.java` file. The code includes package declarations, imports, and a class definition with static variables and a constructor.
- Gradle View:** Shows the `activity2_7` task list, including `build`, `build setup`, `documentation`, `help`, and `other` tasks. The `generateProto` task is highlighted.
- Terminal:** Shows the command `gradle generateProto` being executed in the `activity2_7` directory.

```
package server;

import ...

class SockBaseServer {
    static String logFilename = "logs.txt"; 3 usages

    ServerSocket serv = null; no usages
    InputStream in = null; 4 usages
    OutputStream out = null; 4 usages
    Socket clientSocket = null; 5 usages
    int port = 9099; // default port no usages
    Game game; 3 usages

    public SockBaseServer(Socket sock, Game game){ 1 usage
        this.clientSocket = sock;
        this.game = game;
        try {
            in = clientSocket.getInputStream();
            out = clientSocket.getOutputStream();
        }
    }
}
```

Terminal output:

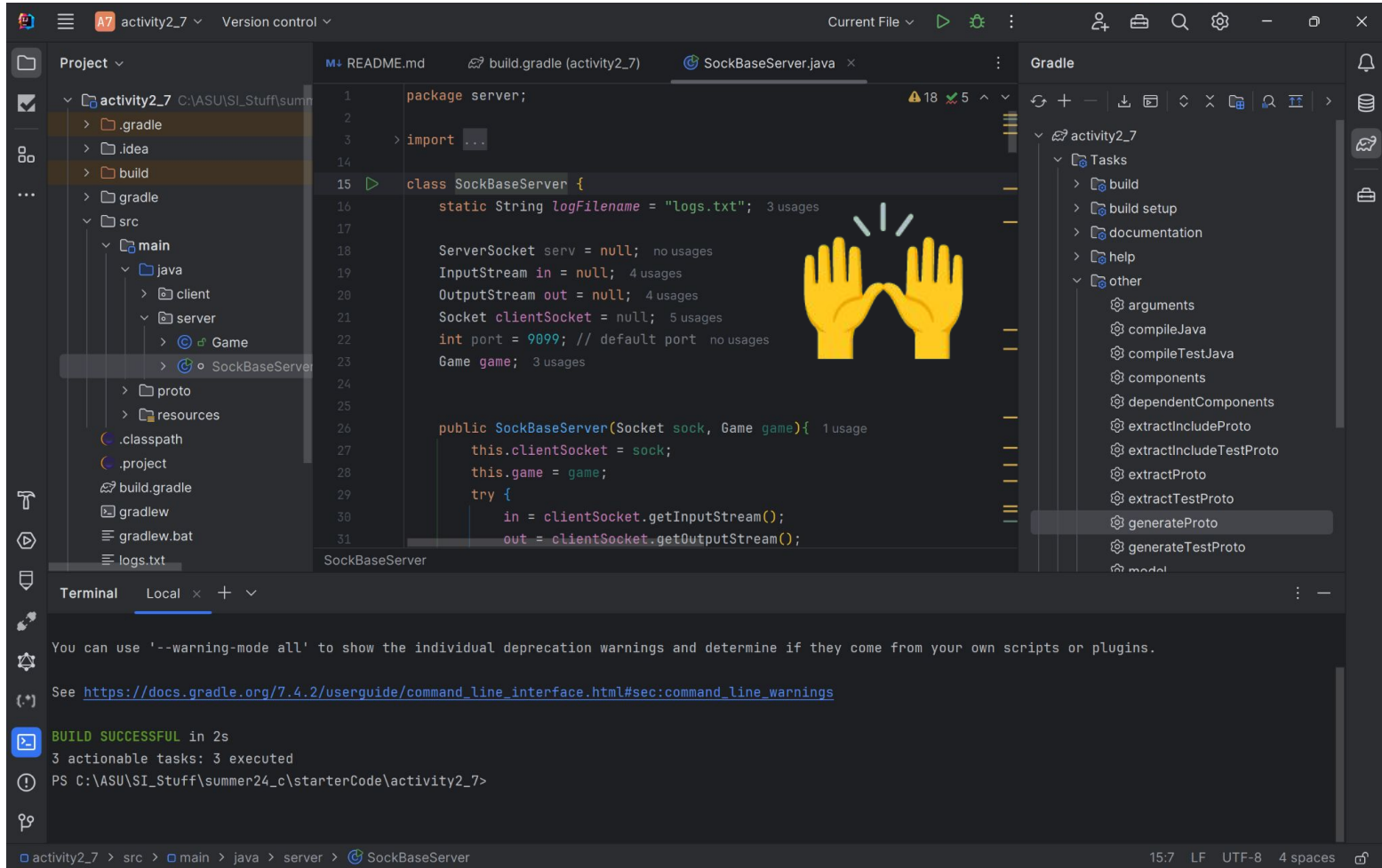
```
PS C:\ASU\SI_Stuff\summer24_c\starterCode\activity2_7> gradle generateProto
```


4-2 Starter Code

SER 321

Protobufs

Step 2



SER 321

Protobufs

Options for Message Creation

```
Response response2 = Response.newBuilder()  
    .setResponseType(Response.ResponseType.START)  
    .setBoard(game.getImage())  
    .setMessage("Select a row and column.")  
    .build();
```

Create the message in
a single statement

Create the message in
increments

```
Response.Builder res = Response.newBuilder()  
    .setResponseType(Response.ResponseType.LEADERBOARD);  
  
res.addLeader(leader);  
res.addLeader(leader2);  
Response response3 = res.build();
```

SER 321

Protobufs

Options for Parsing Messages

```
System.out.println("Type: " + response2.getResponse_type());  
System.out.println("Board: \n" + response2.getBoard());  
System.out.println("Task: \n" + response2.getMessage());
```

Fetch a single value

Fetch a *repeated* value

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

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!
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\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 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\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 [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 individual 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\Sockets\JavaSimpleSock>
```

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
```

```
> 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
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

Why?

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

- 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.

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

> 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

Threaded Server

Given the standard server socket steps...

Ideas on how we could introduce threads?

1. Define Params

2. Create Socket

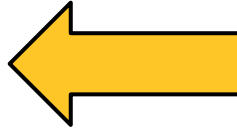
3-5. Mark Socket to Listen

6. Wait for Connection

7. Handle Client Connection

8. Close Client Connection

9. Continue Listening



7. Send Client Socket to thread

SER 321 Threads

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

```
// if it contains only numbers
if (validInput) {
    // convert to an integer
    index = Integer.valueOf(s);
    System.out.println("From client " + id + " get string " + index);
    if (index > -1 & index < buf.length) {
        // if valid, pull the line from the buffer array above and write it to socket
        out.writeObject(buf[index]);
    } else if (index == 5) {
        // fun surprise for mostly correct
        out.writeObject("Close but out of range: https://youtu.be/dQw4w9WgXcQ");
    } else {
        // really wrong
        out.writeObject("index out of range");
    }
}

// wait for next token from the user
s = (String) in.readObject();
}

// on close, clean up
System.out.println("Client " + id + " closed connection.");
in.close();
out.close();
conn.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

```
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

```
// if it contains only numbers
if (validInput) {
    // convert to an integer
    index = Integer.valueOf(s);
    System.out.println("From client " + id + " get string " + index);
    if (index > -1 & index < buf.length) {
        // if valid, pull the line from the buffer array above and write it to socket
        out.writeObject(buf[index]);
    } else if (index == 5) {
        // fun surprise for mostly correct
        out.writeObject("Close but out of range: https://youtu.be/dQw4w9WgXcQ");
    } else {
        // really wrong
        out.writeObject("index out of range");
    }
}

// wait for next token from the user
s = (String) in.readObject();
}

// on close, clean up
System.out.println("Client " + id + " closed connection.");
in.close();
out.close();
conn.close();
} catch (Exception e) {
    e.printStackTrace();
}
}
```

Client
B

Server

```
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);
        }

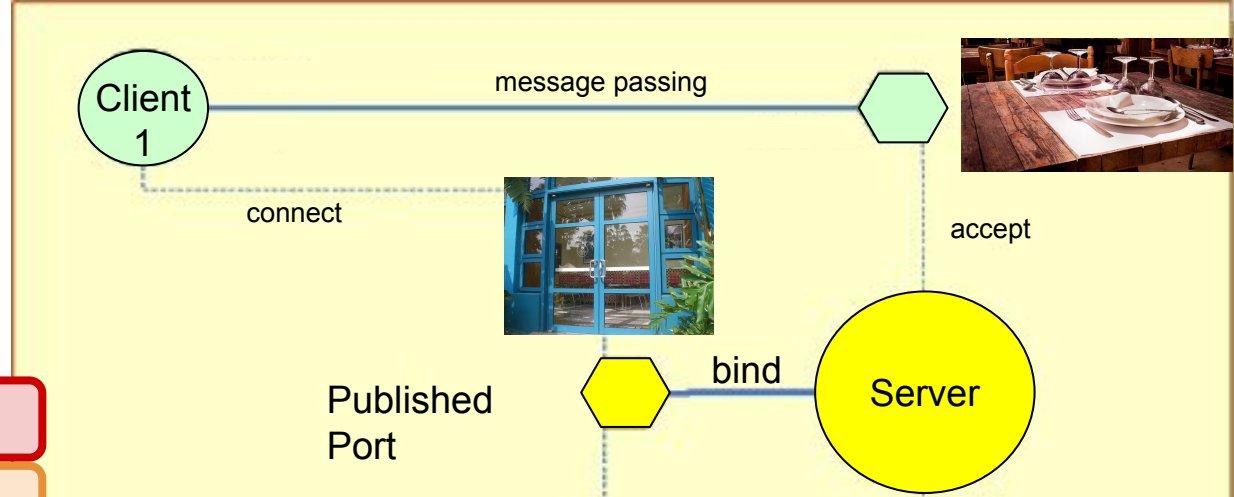
        int portNo = Integer.parseInt(args[0]);
        if (portNo <= 1024)
            portNo = 8888;
        ServerSocket serv = new ServerSocket(portNo);

        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

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

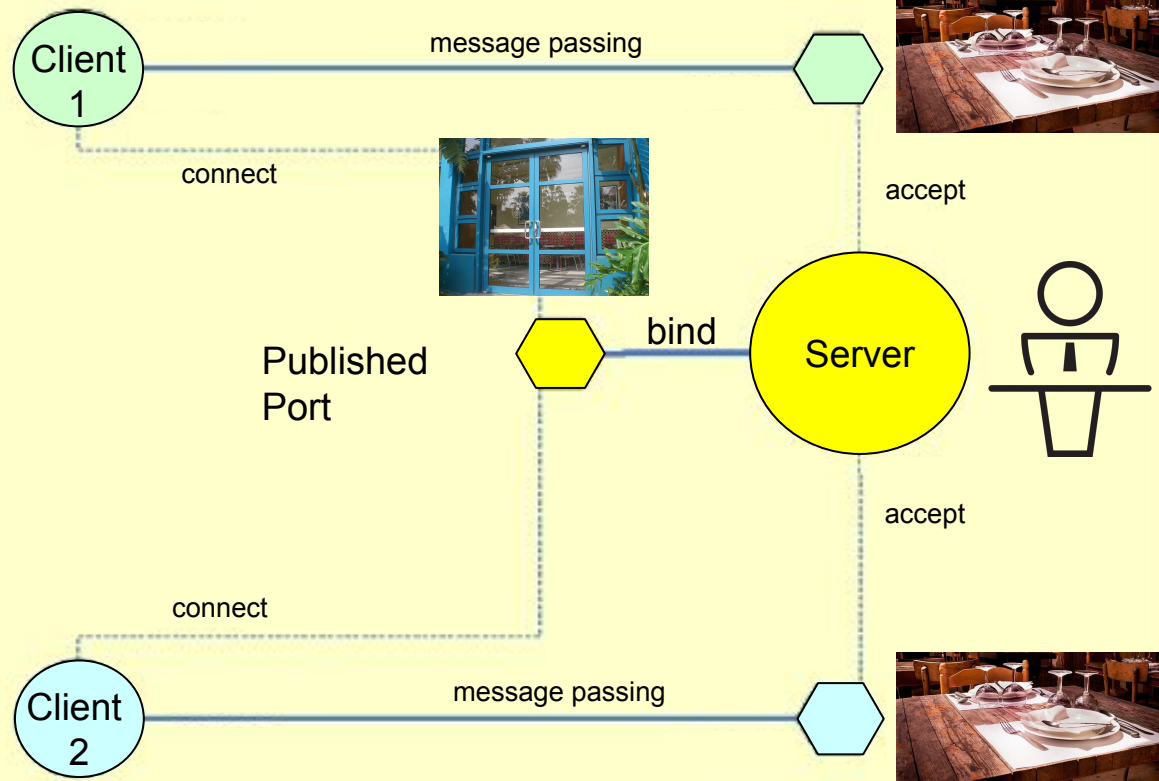


Why do we send the client **socket** to the thread?

SER 321

Threads

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



SER 321

Threaded Server

Now What?

Handle the Client just like before!

1. Define Params

2. Create Socket

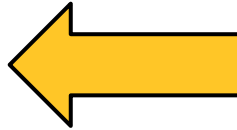
3-5. Mark Socket to Listen

6. Wait for Connection

7. Send Client **Socket** to Thread

8. Close Client Connection

9. Continue Listening



1

2

3

4

5

SER 321

Scratch Space

Questions?



Survey:

<http://bit.ly/ASN2324>



Upcoming Events

SI Sessions:

- Thursday, June 20th at 6:00 pm MST
- Sunday, June 23rd at 6:00 pm MST
- Monday, June 24th at 6:00 pm MST

Review Sessions:

- Review Session - **Wednesday**, July 3rd at 6:00 pm MST (2 hr Session)
- Q&A Session - Sunday, July 7th at 6:00 pm MST (Final Session)

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

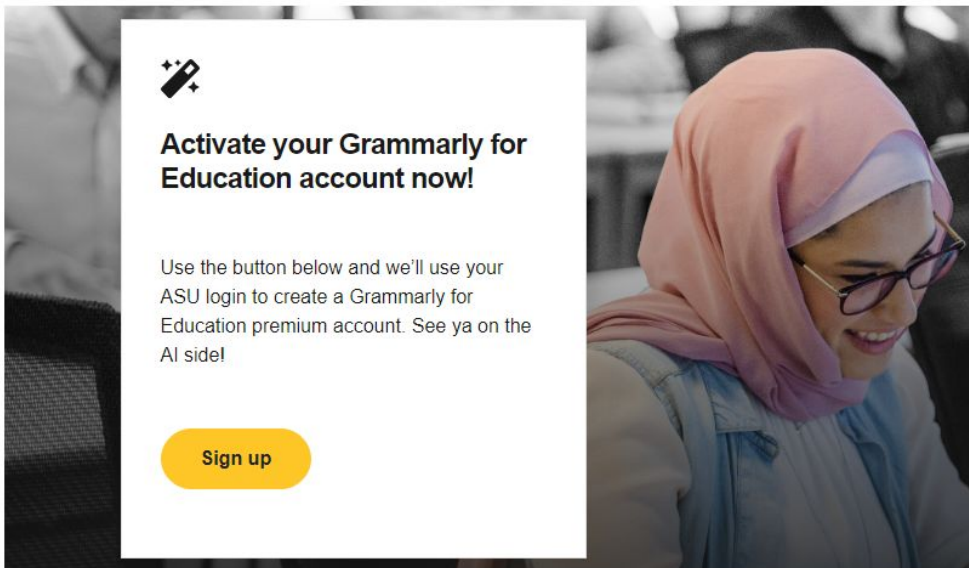
Computer Applications and Information Technology


 [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!





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!

[Sign up](#)



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](#)