SER 321 B Session

SI Session

Thursday, November 14th 2024

7:00 pm - 8:00 pm MST

Agenda

- Protobuf PSA
- Threading Pitfalls
- Concurrency Structures

SI Session Expectations

Thanks for coming to the **SER 321** SI session. We have a packed agenda and we are going to try to get through as many of our planned example problems as possible. This session will be recorded and shared with others.

- If after this you want to see additional examples, please visit the drop-in tutoring center.
- We will post the link in the chat now and at the end of the session.
  - tutoring.asu.edu
- Please keep in mind we are recording this session and it will be made available for you to review 24-48 hours after this session concludes.
- Finally, please be respectful to each other during the session.

Interact with us:

Zoom Features

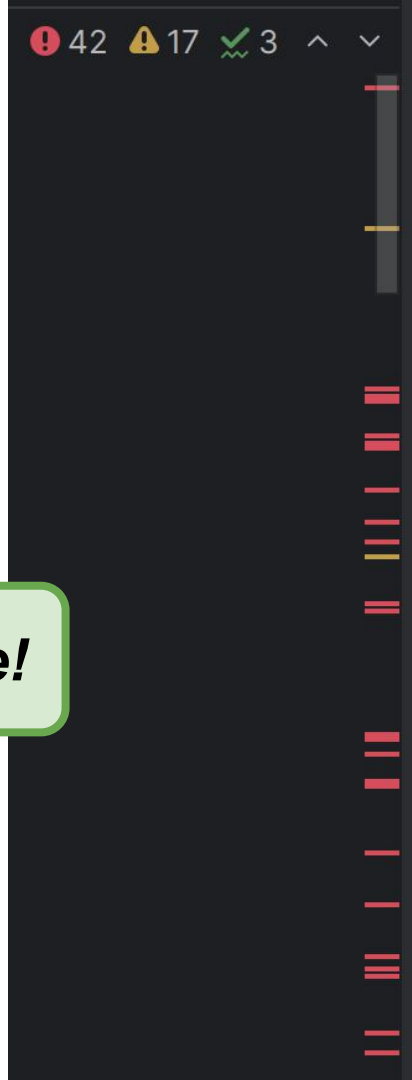
Zoom Chat

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

SER 321
Protobufs

Quick PSA for Protobufs!

You must generate the Protobufs for use!



4-2 Starter Code

SER 321

Protobufs

Assignment4.2given Version control Current File

Project

- Assignment4.2given C:\ASU\SI_Stuff\f
  - .gradle
  - .idea
  - gradle
  - src
    - main
      - java
        - client
        - server
          - Game
          - SockBaseServer
      - proto
        - request.proto
        - response.proto
    - test
  - .gitignore
  - build.gradle
  - gradlew
  - gradlew.bat
  - PROTOCOL.md
  - README.md
- External Libraries
- Scratches and Consoles

README.md build.gradle (Assignment4.2given) SockBaseServer.java

```
package server;

import ...

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

    // Please use these as given so it works with our test cases
    static String menuOptions = "\nWhat would you like to do? \n 1 - to see
    static String gameOptions = "\nChoose an action: \n (1-9) - Enter an int


    ServerSocket serv = null;  no usages
    InputStream in = null;  3 usages
    OutputStream out = null;  4 usages
    Socket clientSocket = null;  4 usages
    private final int id; // client id  2 usages

    Game game; // current game  3 usages

    private boolean inGame = false; // a game was started (you can de
    private String name; // player name  4 usages

    private int currentState =1; // I used something like this to keep track

    private static boolean grading = true; // if the grading board should be

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

42 17 3

Gradle

- Assignment4.2given
  - Tasks
    - build
    - build setup
    - documentation
    - help
    - other
      - arguments
      - compileJava
      - compileTestJava
      - components
      - dependentComponents
      - extractIncludeProto
      - extractIncludeTestProto
      - extractProto
      - extractTestProto
      - generateProto
      - generateTestProto
      - model
      - prepareKotlinBuildScriptMode
      - processResources
      - processTestResources
      - runClient
      - runServer
      - runServerGrading
    - verification
  - Dependencies

Assignment4.2given > src > main > java > server > SockBaseServer 2:1 LF UTF-8 4 spaces

4-2 Starter Code

SER 321

Protobufs

Assignment4.2given Version control Current File

Project

- Assignment4.2given C:\ASU\SI_Stuff\f
  - .gradle
  - .idea
  - gradle
  - src
    - main
      - java
        - client
        - server
          - Game
          - SockBaseServer
      - proto
        - request.proto
        - response.proto
    - test
  - .gitignore
  - build.gradle
  - gradlew
  - gradlew.bat
  - PROTOCOL.md
  - README.md

README.md build.gradle (Assignment4.2given) SockBaseServer.java

```
1   package server;
2
3   import ...
12
13  class SockBaseServer {
14      static String logFilename = "logs.txt";  3 usages
15
16      // Please use these as given so it works with our test cases
17      static String menuOptions = "\nWhat would you like to do? \n 1 - to see
18      static String gameOptions = "\nChoose an action: \n (1-9) - Enter an in
19
20
21      ServerSocket serv = null;  no usages
22      InputStream in = null;  3 usages
23      OutputStream out = null;  4 usages
24      Socket clientSocket = null;  4 usages
25      private final int id; // client id  2 usages
26
27      Game game; // current game  3 usages
28
29      private boolean inGame = false; // a game was started (you can decide if
30      private String name; // player name  4 usages
```

42 17 3

Gradle

- Assignment4.2given
  - Tasks
    - build
    - build setup
    - documentation
    - help
    - other
      - arguments
      - compileJava
      - compileTestJava
      - components
      - dependentComponents
      - extractIncludeProto
      - extractIncludeTestProto
      - extractProto
      - extractTestProto
      - generateProto
      - generateTestProto
      - model

Terminal Local

```
PS C:\ASU\SI_Stuff\fall24_b\starter_code\Assignment4.2given> gradle generateProto
```

Assignment4.2given > src > main > java > server > SockBaseServer 2:1 LF UTF-8 4 spaces

4-2 Starter Code

SER 321

Protobufs

Assignment4.2given Version control Current File

Project

- Assignment4.2given C:\ASU\SI_Stuff\f
  - .gradle
  - .idea
  - build
  - gradle
  - src
    - main
      - java
        - client
        - server
          - Game
          - SockBaseServer
      - proto
        - request.proto
        - response.proto
    - test
  - .gitignore

README.md build.gradle (Assignment4.2given) SockBaseServer.java

```
package server;

import ...

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

    // Please use these as given so it works with our test cases
    static String menuOptions = "\nWhat would you like to do? \n 1 - to see
    static String gameOptions = "\nChoose an action: \n (1-9) - Enter an int


    ServerSocket serv = null;  no usages
    InputStream in = null;  3 usages
    OutputStream out = null;  4 usages
    Socket clientSocket = null;  4 usages
    private final int id; // client id  2 usages
```

Gradle

- Assignment4.2given
  - Tasks
    - build
    - build setup
    - documentation
    - help
    - other
      - arguments
      - compileJava
      - compileTestJava
      - components
      - dependentComponents
      - extractIncludeProto
      - extractIncludeTestProto
      - extractProto
      - extractTestProto

Terminal Local

```
PS C:\ASU\SI_Stuff\fall24_b\starter_code\Assignment4.2given> gradle generateProto
Starting a Gradle Daemon (subsequent builds will be faster)

Deprecated Gradle features were used in this build, making it incompatible with Gradle 9.0.

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

For more on this, please refer to https://docs.gradle.org/8.10/userguide/command_line_interface.html#sec:command_line_warnings in the Gradle documentation.

BUILD SUCCESSFUL in 33s
3 actionable tasks: 3 executed
PS C:\ASU\SI_Stuff\fall24_b\starter_code\Assignment4.2given>
```

Assignment4.2given > src > main > java > server > SockBaseServer 2:1 LF UTF-8 4 spaces

SER 321
Protobufs

Options for Message Creation

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

SockBaseServer

Create the message in a single statement

Create the message in increments

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

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

SockBaseClient - main

SER 321
Protobufs

Options for Message Creation

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

SockBaseClient - chooseMenu

Need one more step…

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

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

SockBaseClient - main

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

SER 321
Protobufs

Parsing Messages

GETTERS!

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

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

Fetch a single value

Fetch a *repeated* value

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

Austin Walter's Traffic Comparison

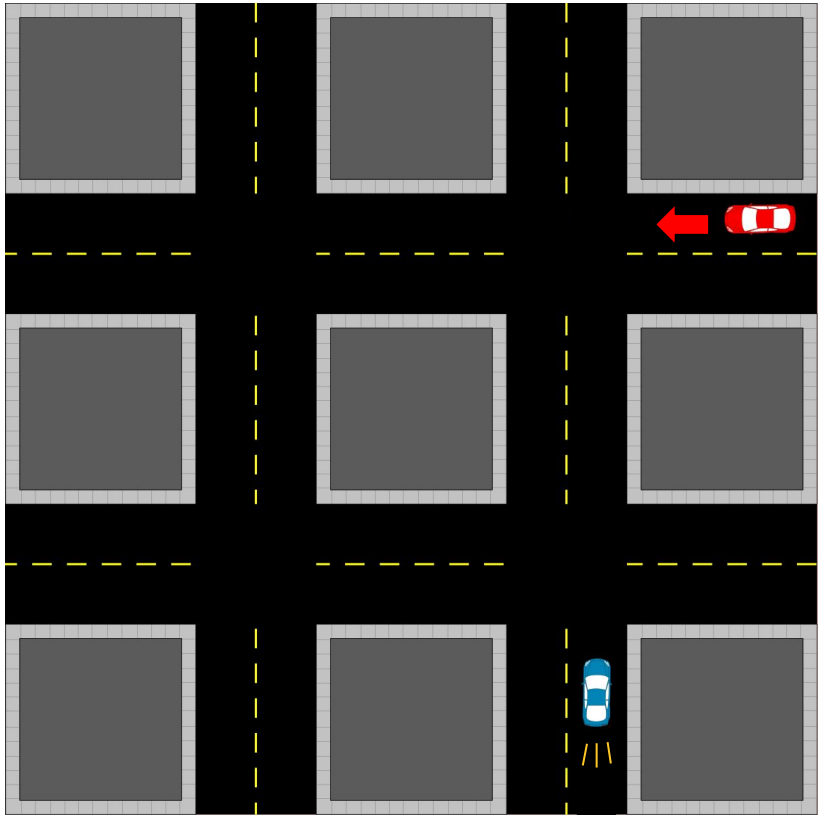
SER 321

Threading Pitfalls

Race Condition

Crash

More than one thread accesses a single resource at once



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

Race Condition

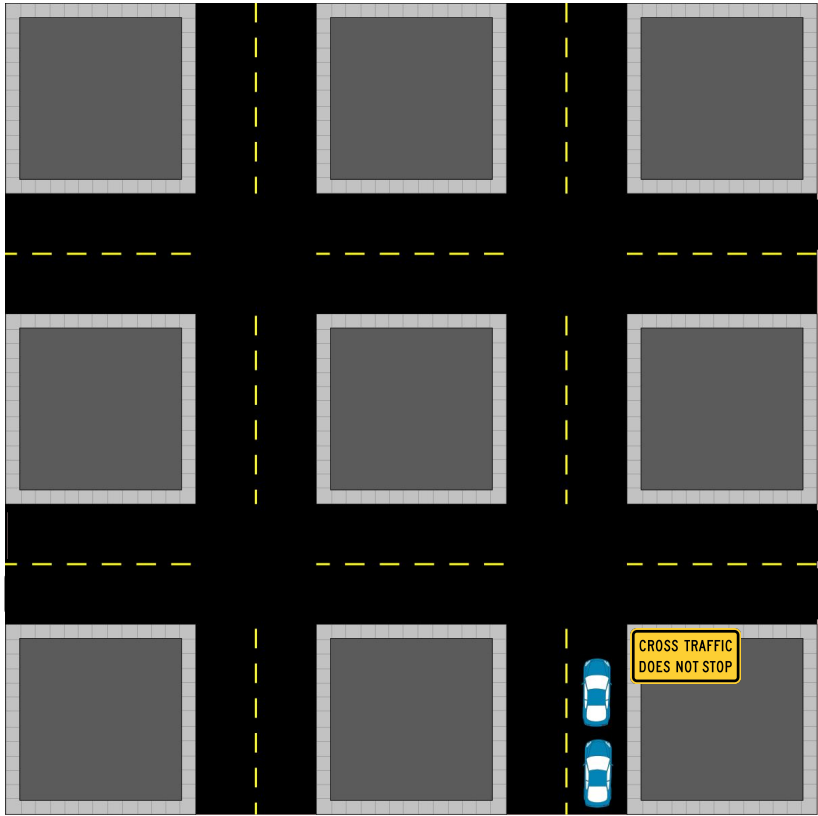
Crash

More than one thread accesses a single resource at once

Starvation

Cross Traffic

A thread never gains access to the resource it needs



Austin Walter's Traffic Comparison

SER 321

Threading Pitfalls

Race Condition

Crash

More than one thread accesses a single resource at once

Starvation

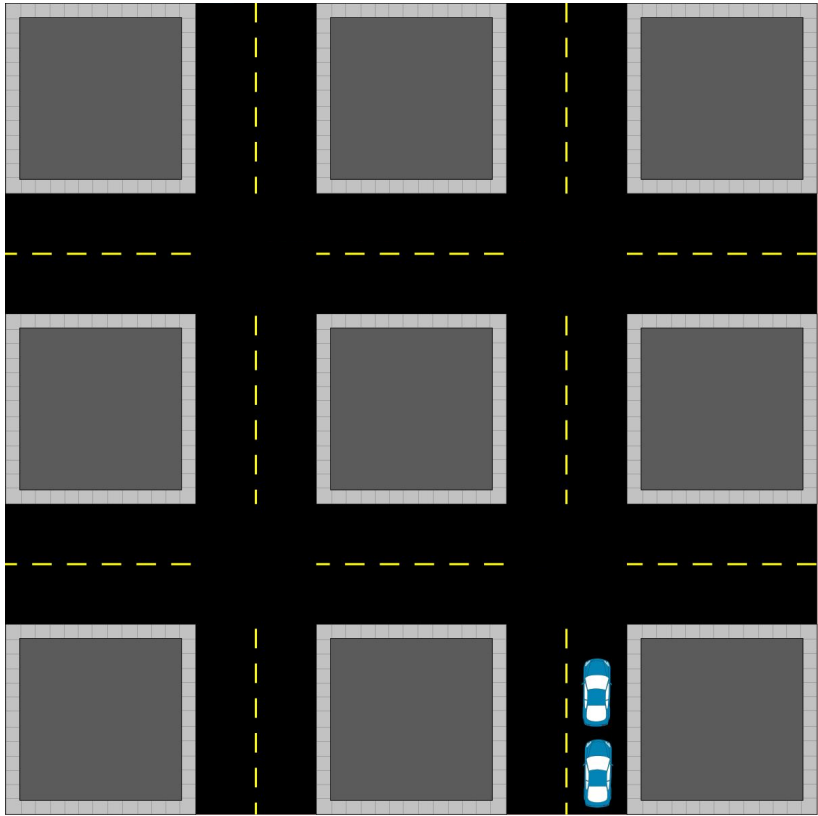
Cross Traffic

A thread never gains access to the resource it needs

Deadlock

Gridlock

A thread is only able to acquire some of the needed resources



Can we name some concurrency structures?

Atomic Operations & Variables

Locks

Semaphores

Monitors

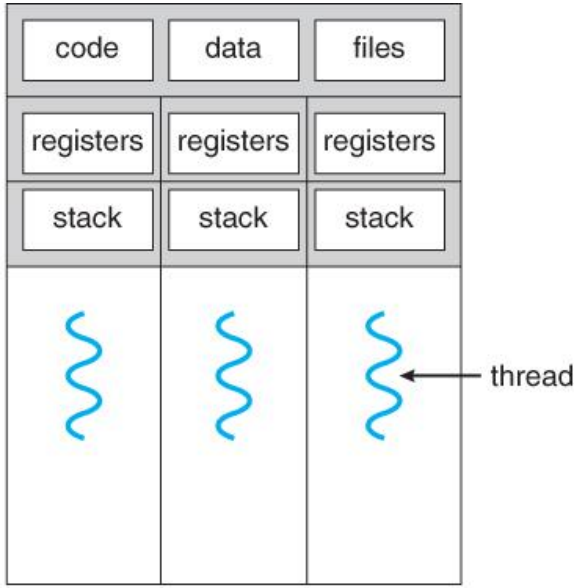
SER 321

Concurrency Structures

Atomic Operations & Variables

Recall *registers*…

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




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

SER 321
Concurrency Structures

Pros and Cons?

Locks

Acquire the Lock → Open & Enter / Close & Lock

Release the Lock → Unlock & Exit

SER 321

Concurrency Structures

How am I different from a lock?

Semaphores

More than one stall!

Acquire Lock → Open & Enter / Close & Lock

Release Lock → Unlock & Exit

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

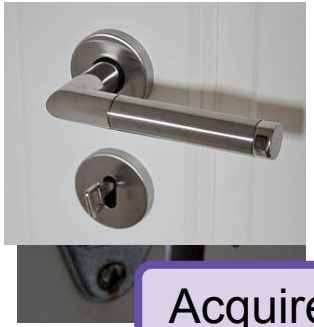
When would that be beneficial?

SER 321
Concurrency Structures

Pros and Cons?

Monitors

You lock the main door instead!



Acquire Lock → { Open & Enter; Close & Lock }

Release Lock → Unlock & Exit

Covers the ***entire object***

SER 321

Concurrency Structures

RECAP

Atomic Operations & Variables

YOU control the locks directly

Locks

YOU control the locks directly

Semaphores

YOU control the locks directly

Monitors

Locks managed for you

SER 321
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

JavaThreadSock

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

```java
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();
    }
}
}
```

JavaThreadSock

SER 321

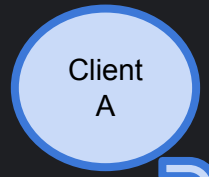
Threads

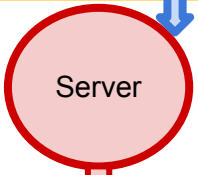
```java
public void run() {
  try {
    // setup read/write channels for connection
    ObjectInputStream in = new ObjectInputStream(conn.getInputStream());
    ObjectOutputStream out = new ObjectOutputStream(conn.getOutputStream());

    // 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");
      }
```

```java
    // 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();
}
}
```






```java
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();
  }
}
}
```

JavaThreadSock

SER 321

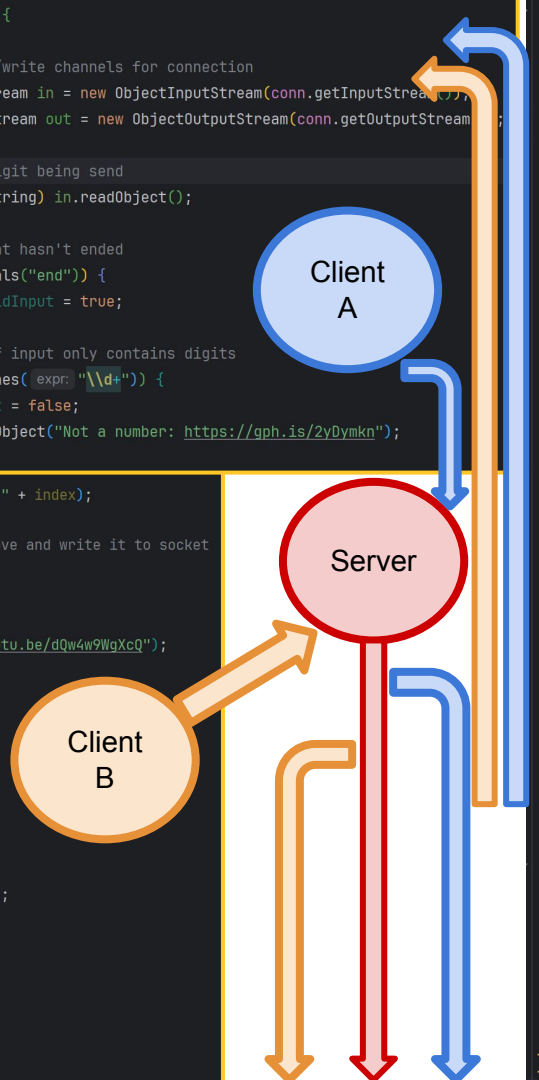
Threads

```java
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");
      }
```

```java
      // 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();
  }
}
```




```java
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();
  }
}
}
```

JavaThreadSock

SER 321

Threads

```java
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

```java
      // 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();
  }
}
```

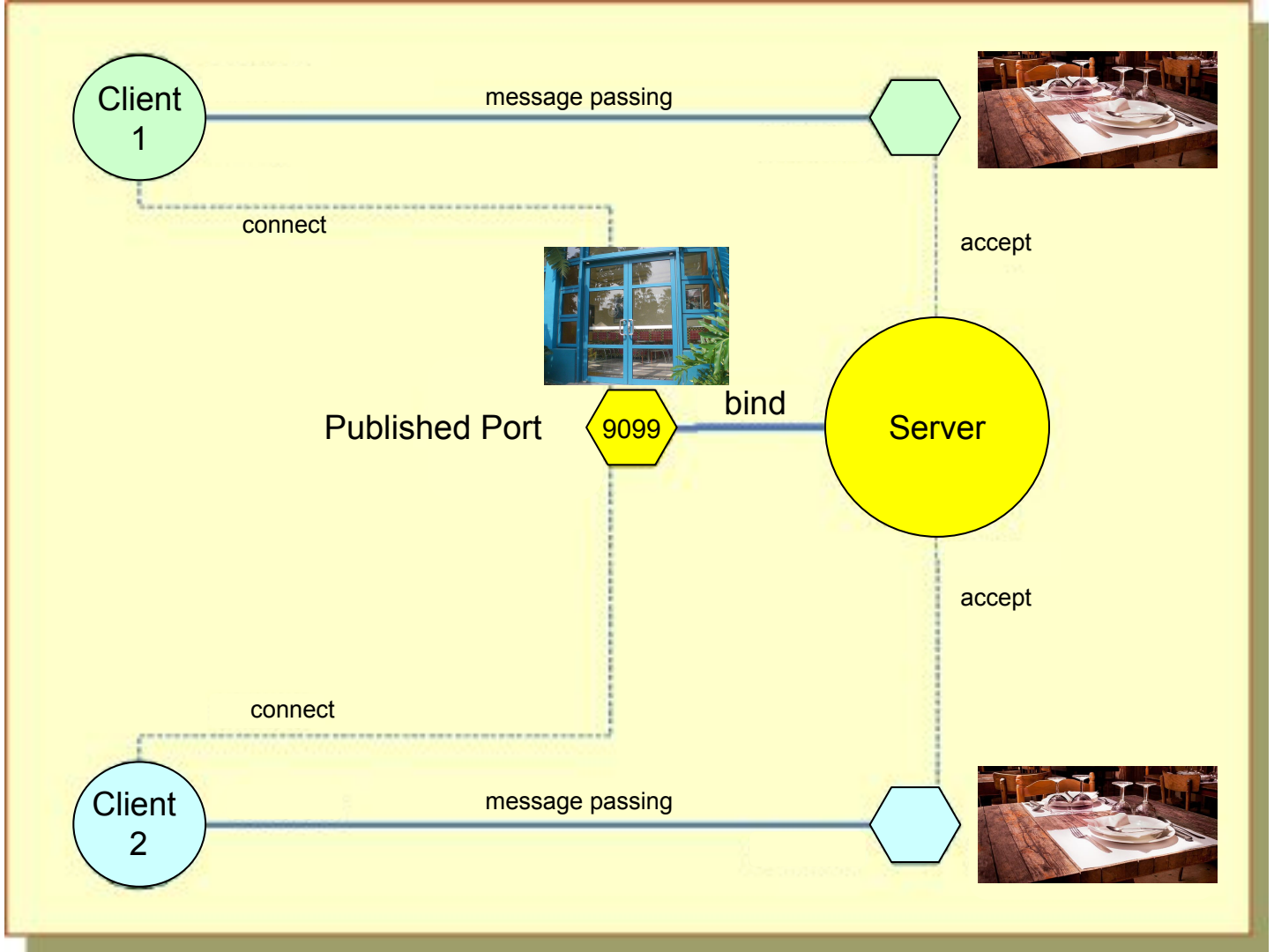
Server

Client B

```java
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
Sockets!



Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321

Scratch Space

Upcoming Events

SI Sessions:

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

Review Sessions:

- Sunday, December 1st at 7:00 pm MST - **2 hour Review Session**
- Tuesday, December 3rd at 10:00 am MST - **Q&A Session**

Questions?

Survey:

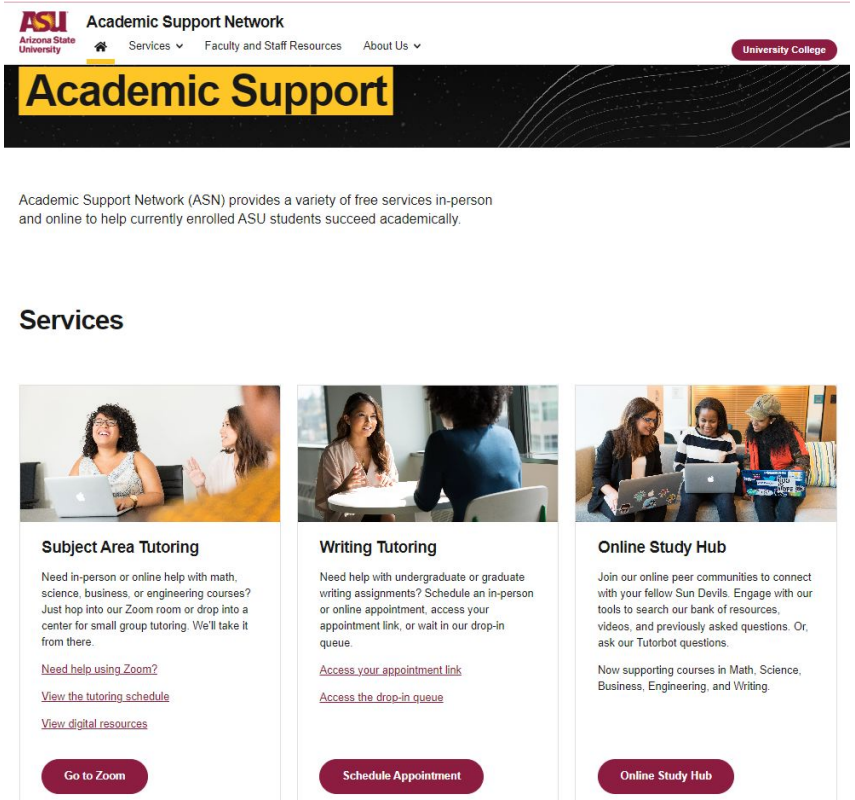
https://asuasn.info/ASNSurvey

29

More Questions?

Check out our other resources!

tutoring.asu.edu



1 – Go to Zoom

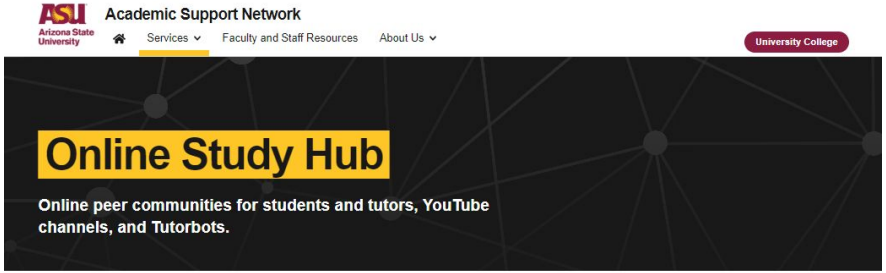
2- Need help using Zoom?
View the tutoring schedule
View digital resources

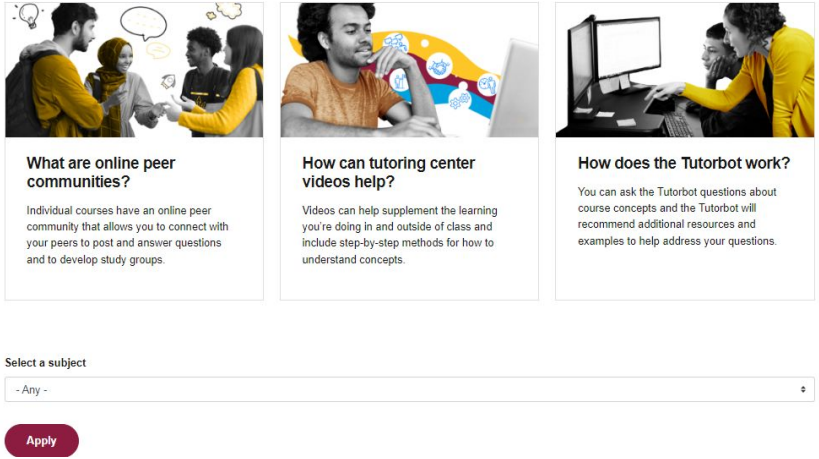
1. **Click on 'Go to Zoom' to log onto our Online Tutoring Center.**
2. **Click on 'View the tutoring schedule' to see when tutors are available for specific courses.**

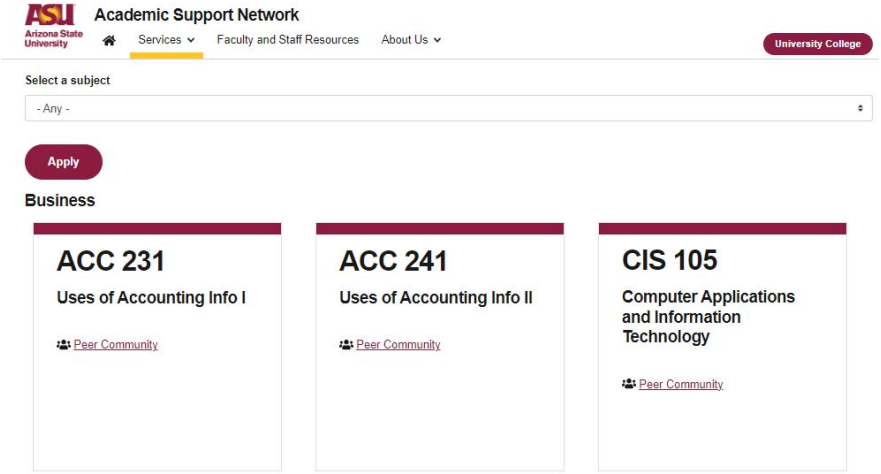
More Questions?

Check out our other resources!

tutoring.asu.edu/online-study-hub



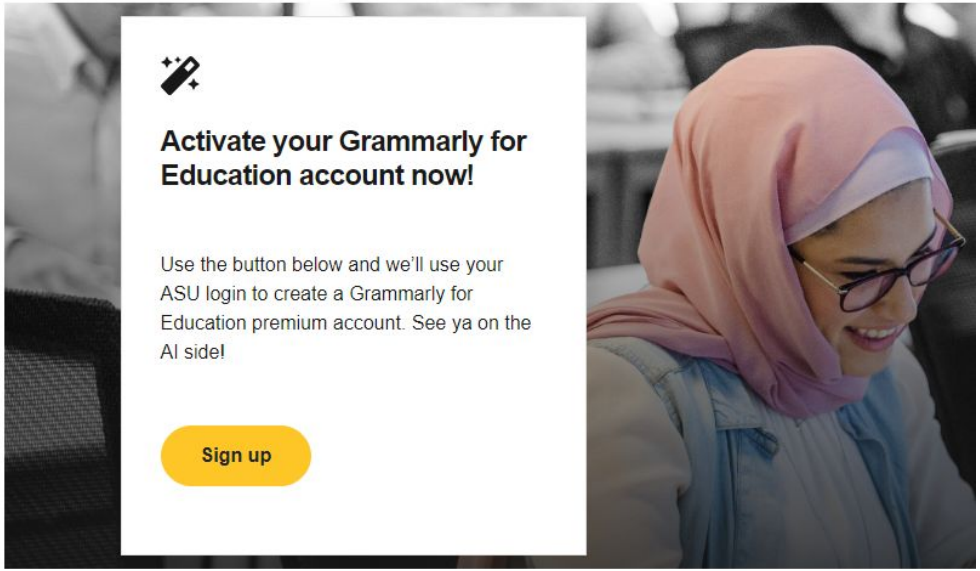




Don't forget to check out the Online Study Hub for additional resources!

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