SER 321 B Session

SI Session

Sunday, April 7th 2024

7:00 pm - 8:00 pm MST

Agenda

- Threads!
- Threading your Server
- Threading Pitfalls

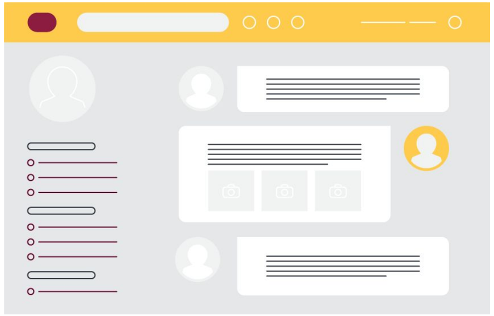
SI Session Expectations

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

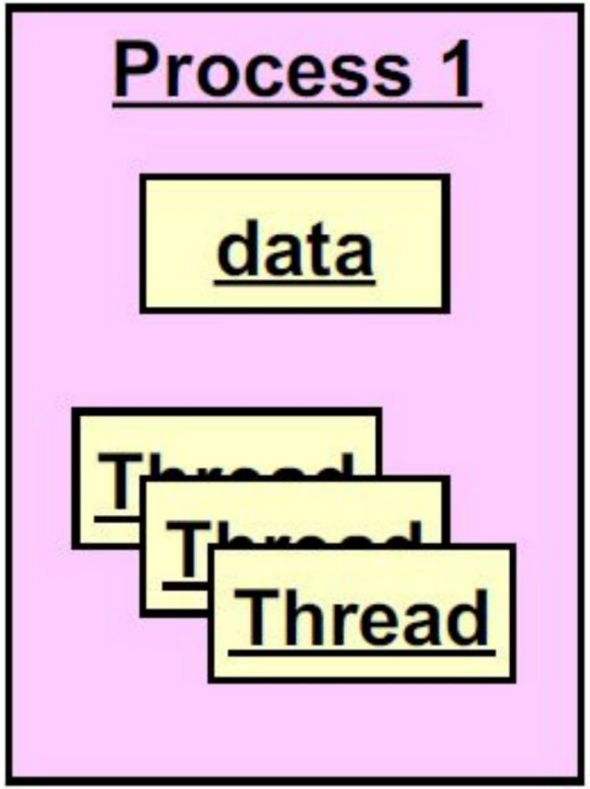
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321

Threads

Remember that they exist *within* the parent process

What does that imply?

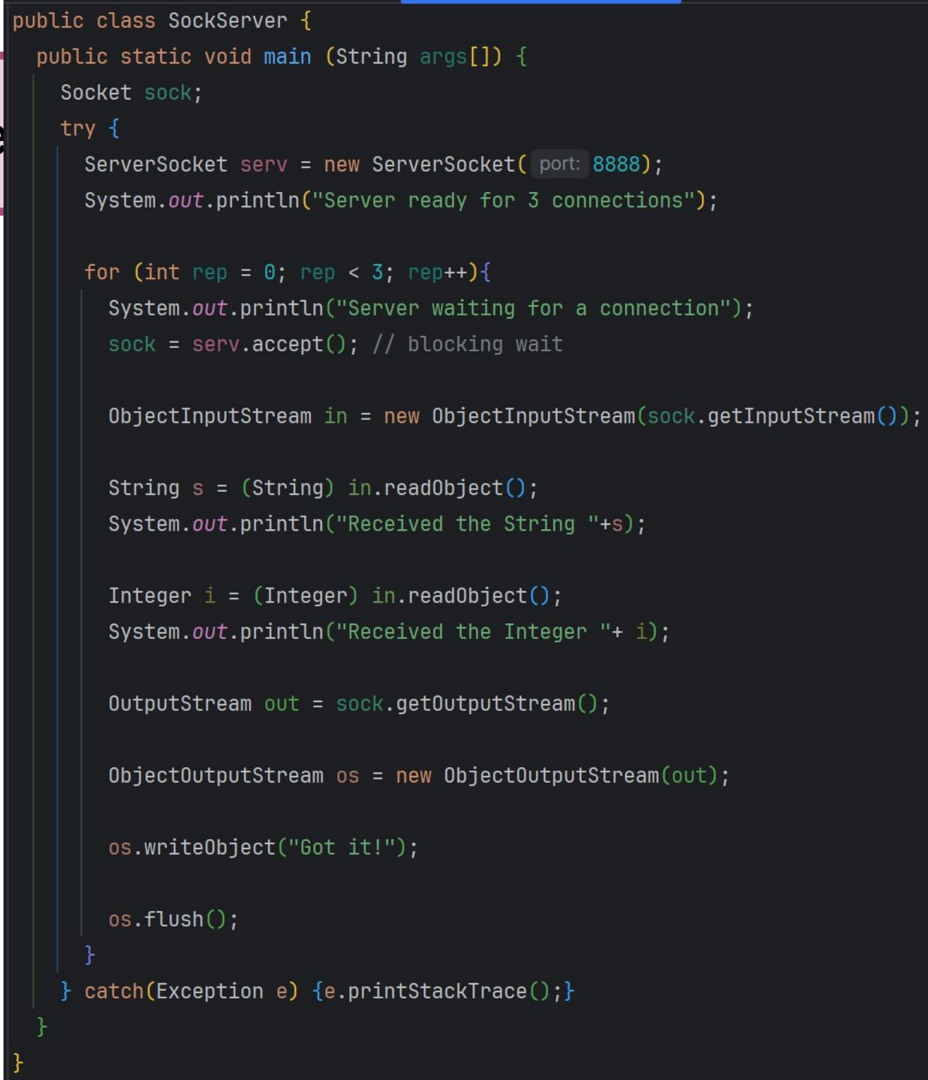


JavaSimpleSock2

SER 321
Threads

Do we remember the

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

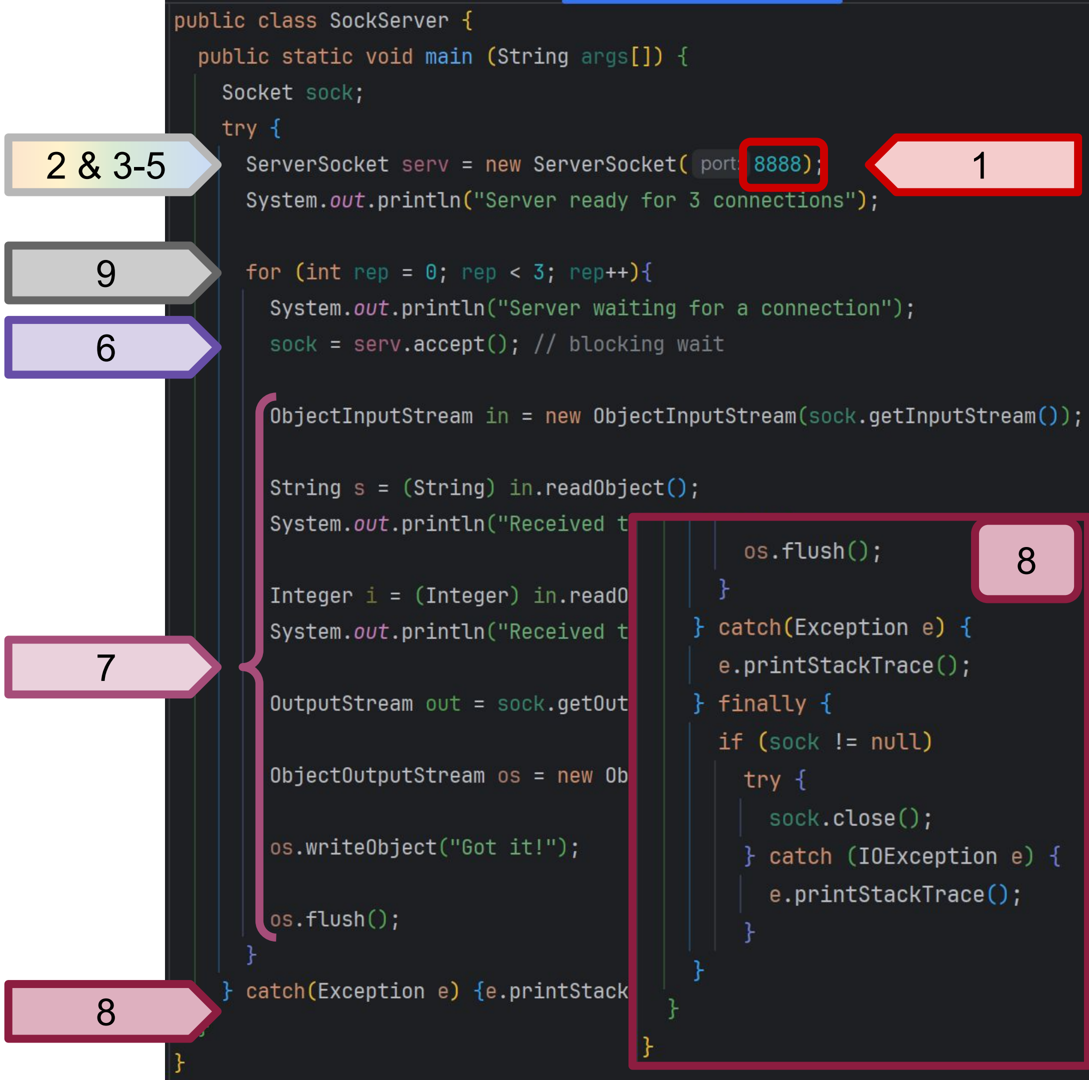


```
public class SockServer {
  public static void main (String args[]) {
    Socket sock;
    try {
      ServerSocket serv = new ServerSocket( port: 8888);
      System.out.println("Server ready for 3 connections");

      for (int rep = 0; rep < 3; rep++){
        System.out.println("Server waiting for a connection");
        sock = serv.accept(); // blocking wait

        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();}
  }
}
```

JavaSimpleSock2

SER 321
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

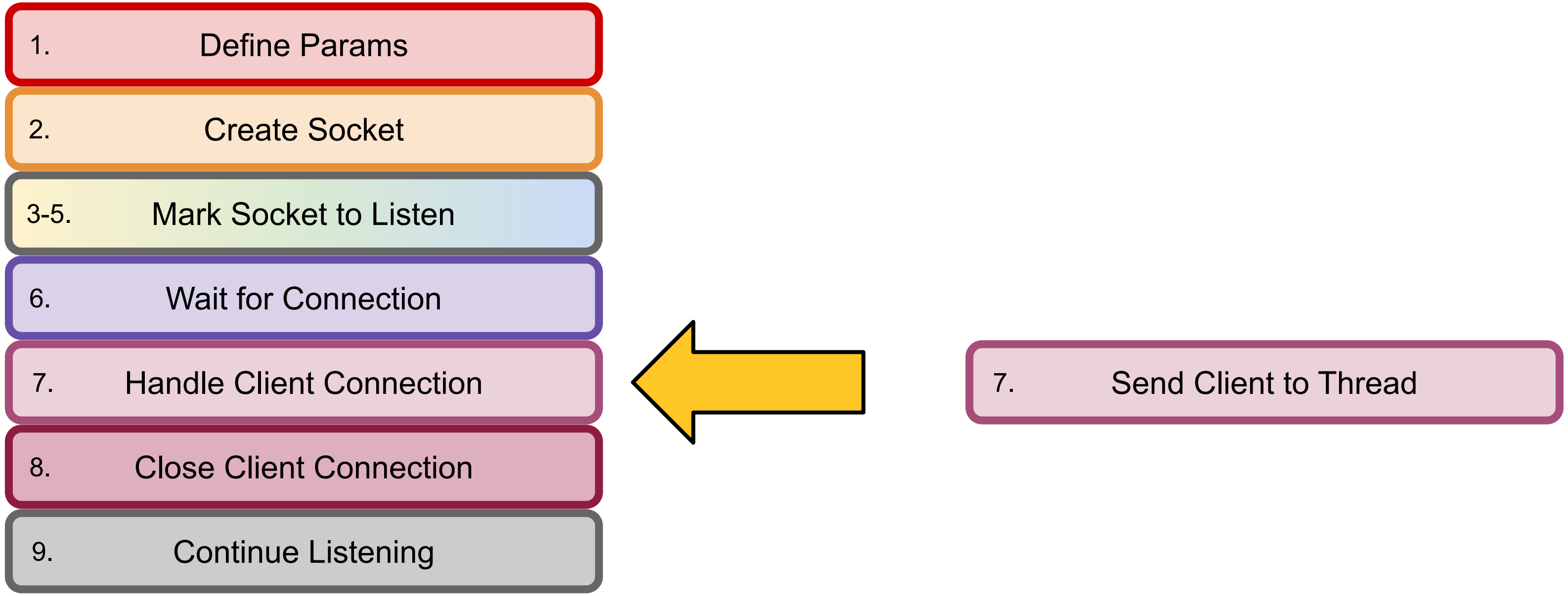


```
public class SockServer {
  public static void main (String args[]) {
    Socket sock;
    try {
      ServerSocket serv = new ServerSocket( port: 8888);
      System.out.println("Server ready for 3 connections");

      for (int rep = 0; rep < 3; rep++){
        System.out.println("Server waiting for a connection");
        sock = serv.accept(); // blocking wait

        ObjectInputStream in = new ObjectInputStream(sock.getInputStream());

        String s = (String) in.readObject();
        System.out.println("Received t

        Integer i = (Integer) in.readO
        System.out.println("Received t

        OutputStream out = sock.getOut

        ObjectOutputStream os = new Ob

        os.writeObject("Got it!");

        os.flush();
      }
    } catch(Exception e) {e.printStack
}
```

```
          os.flush();
        }
      } catch(Exception e) {
        e.printStackTrace();
      } finally {
        if (sock != null)
          try {
            sock.close();
          } catch (IOException e) {
            e.printStackTrace();
          }
      }
    }
}
```

SER 321
Threads

Ideas on how we would go about threading this?



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

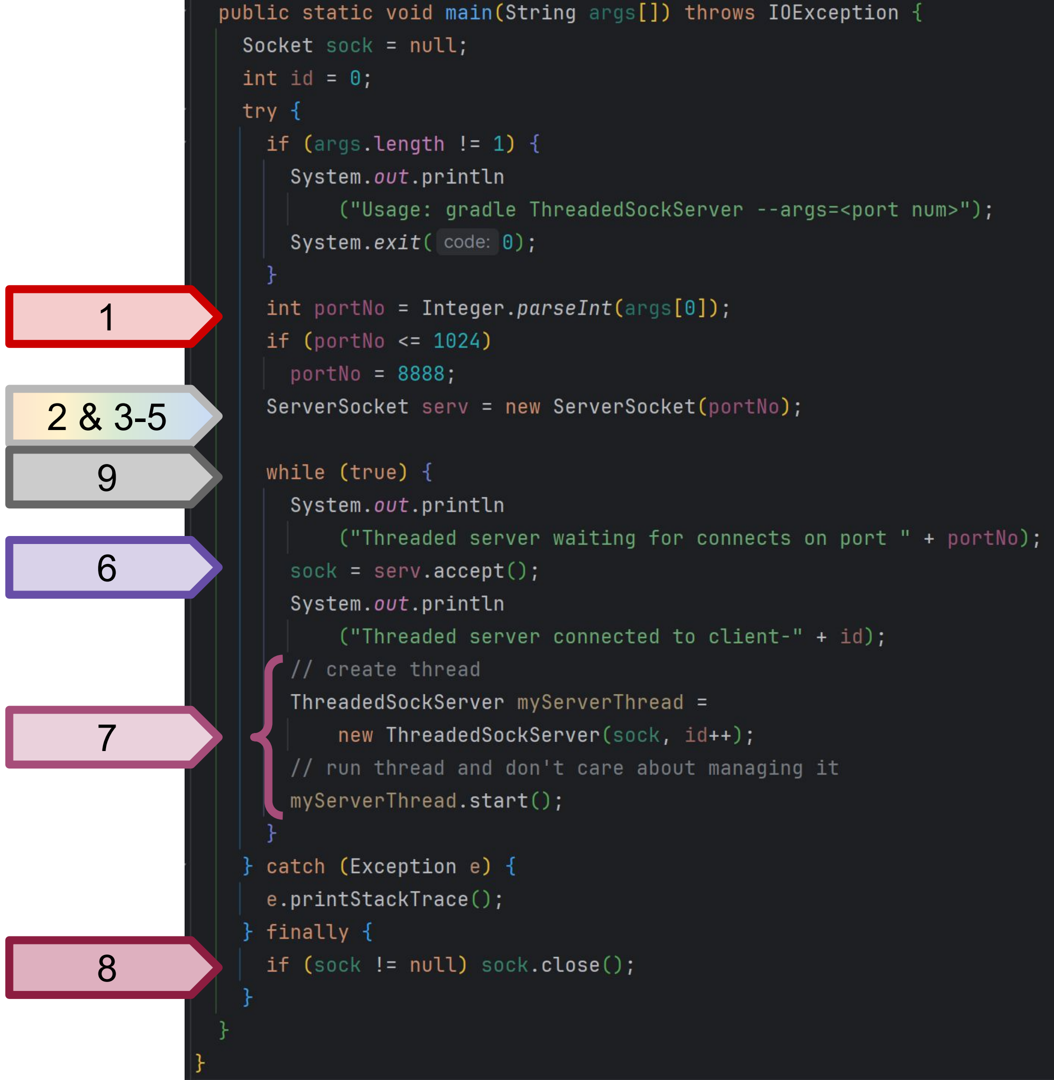
7. Send Client to Thread

JavaThreadSock

SER 321

Threads

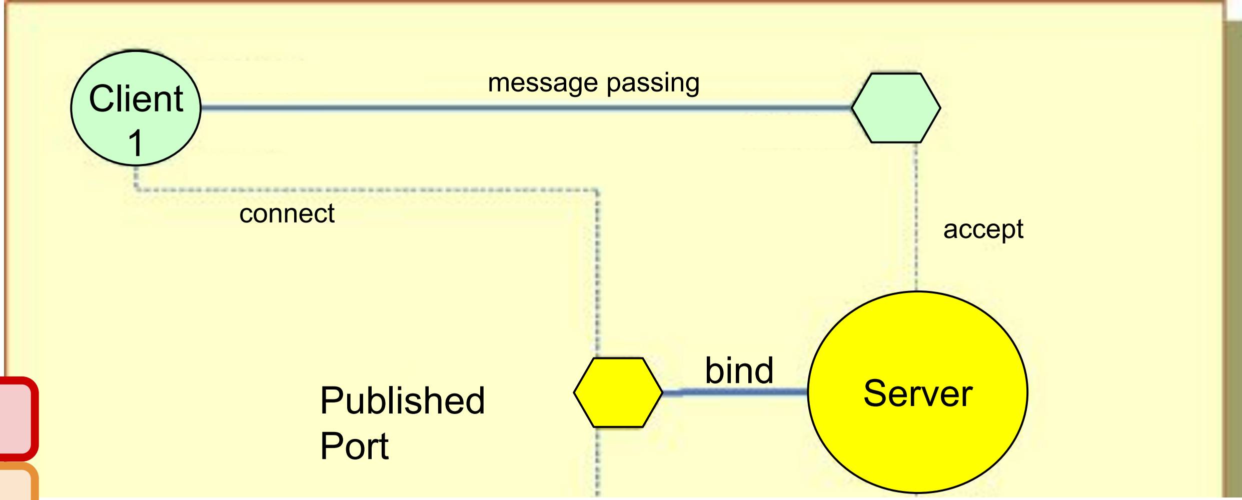
1. Define Params
2. Create Socket
3-5. Mark Socket to Listen
6. Wait for Connection
7. Send Client to Thread
8. Close Client Connection
9. Continue Listening



```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
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

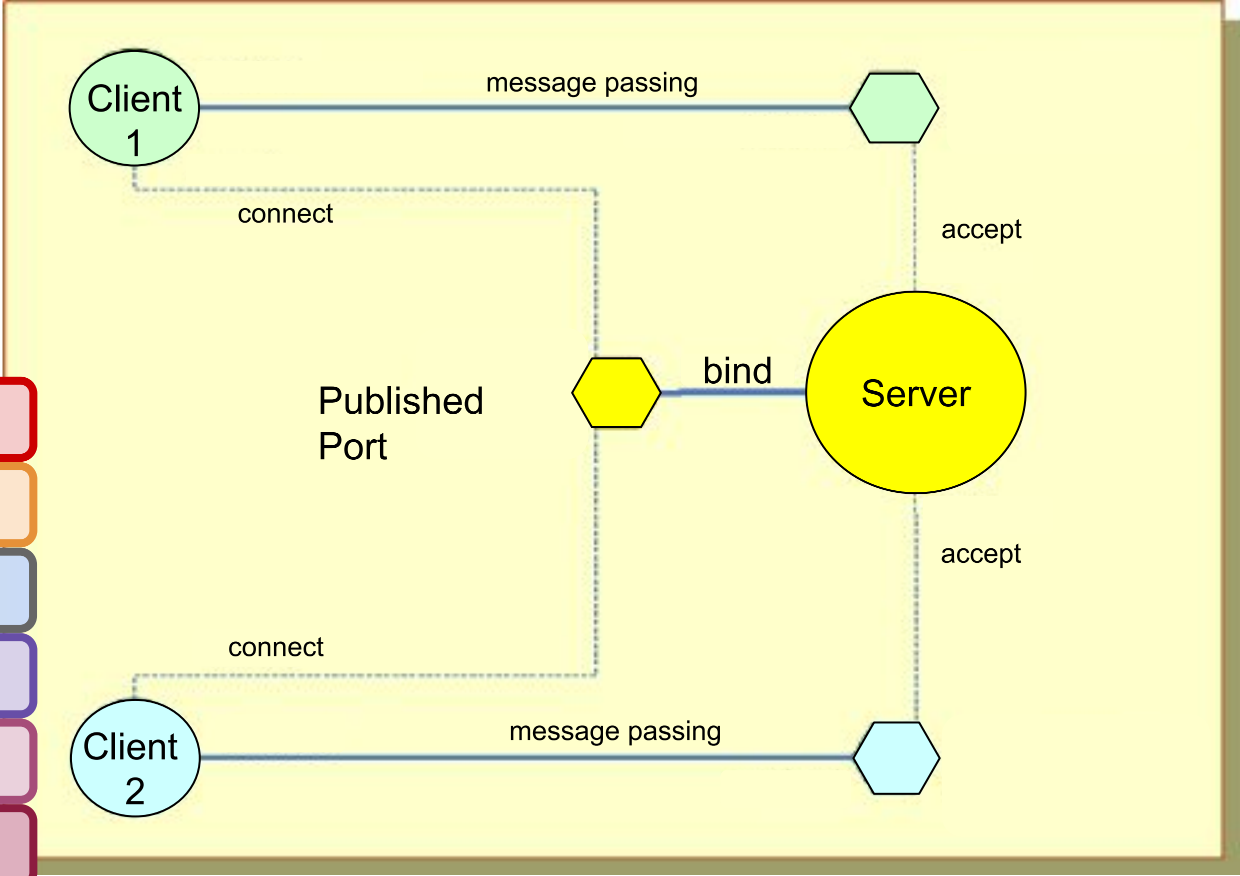


Why do we send the client **socket** to the thread?

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
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



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

Threading Pitfalls

Race Condition

A thread never gains access to the resource it needs

Starvation

A thread is only able to acquire some of the resources it needs

Deadlock

More than one thread accesses a single resource at the same time

SER 321

Threading Pitfalls

Race Condition

Starvation

Deadlock

A thread never gains access to the resource it needs

A thread is only able to acquire some of the resources it needs

More than one thread accesses a single resource at the same time

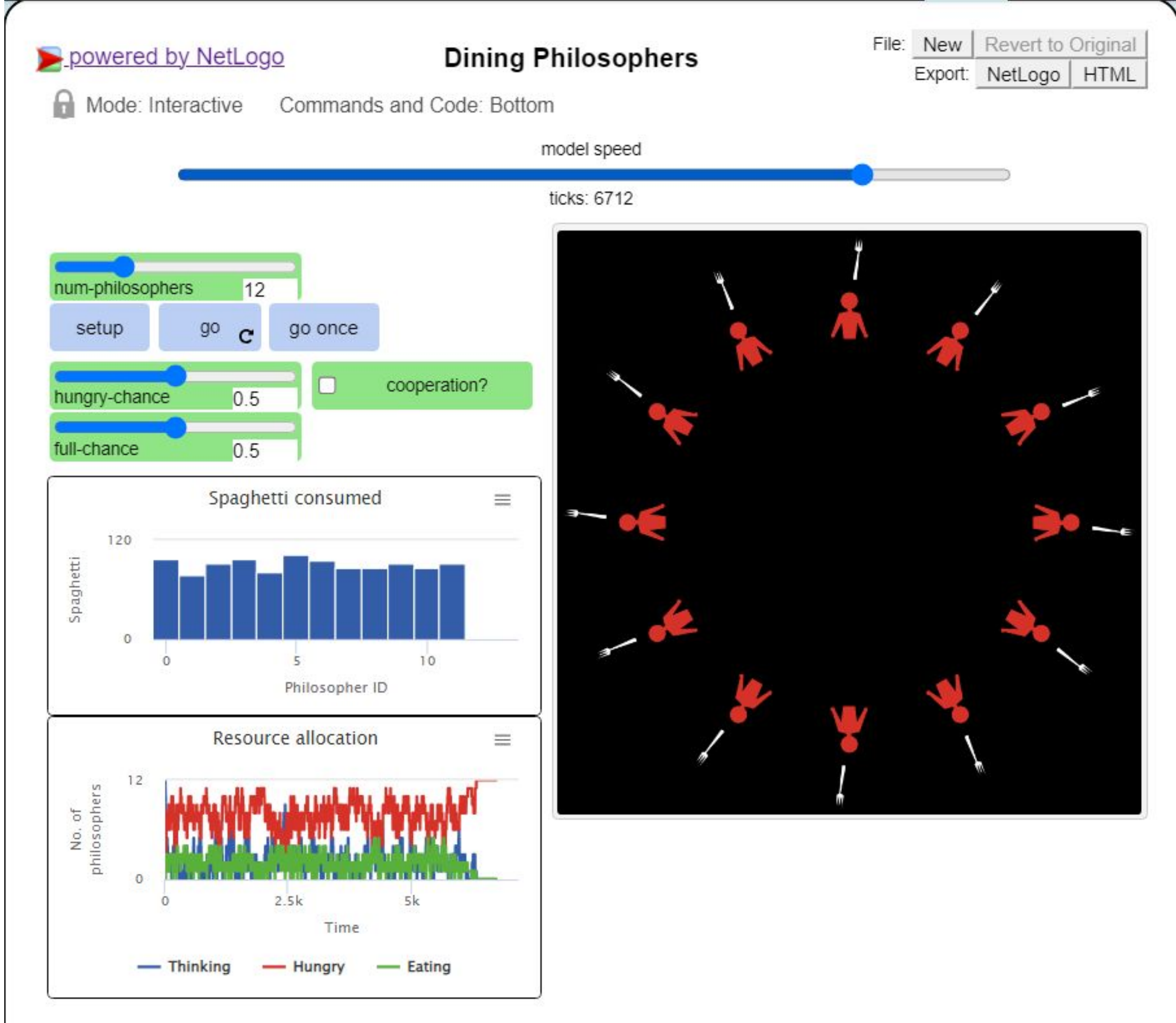
Dining Philosophers

SER 321
Threading Pitfalls

What does *Spaghetti Consumed* represent?

What does *Thinking* represent?

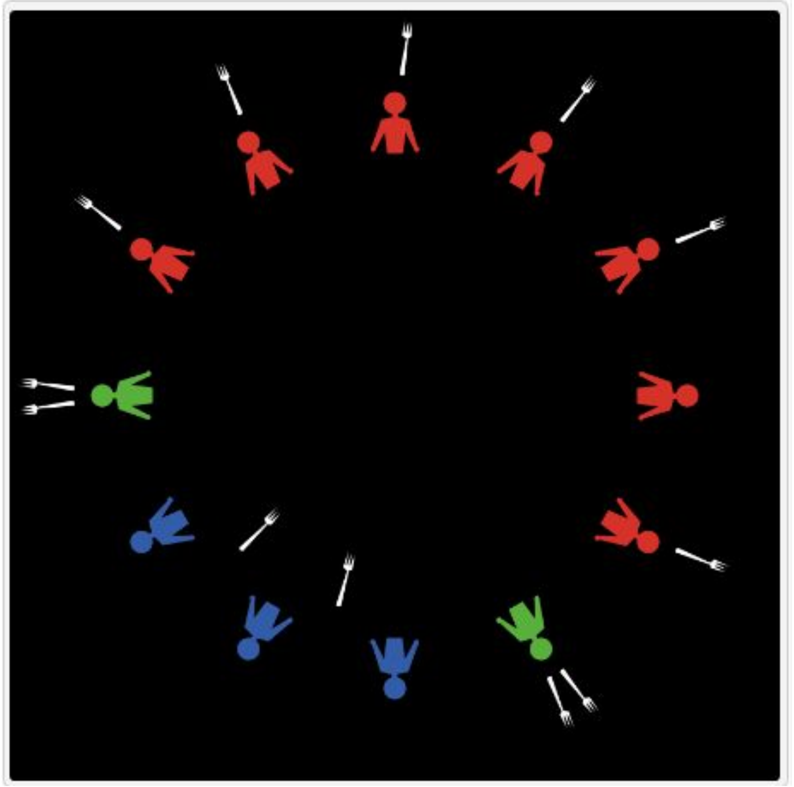
What does *Hungry* represent?



SER 321

Threading Pitfalls

Can we identify any concurrency issues here?



NetworkDeadlock

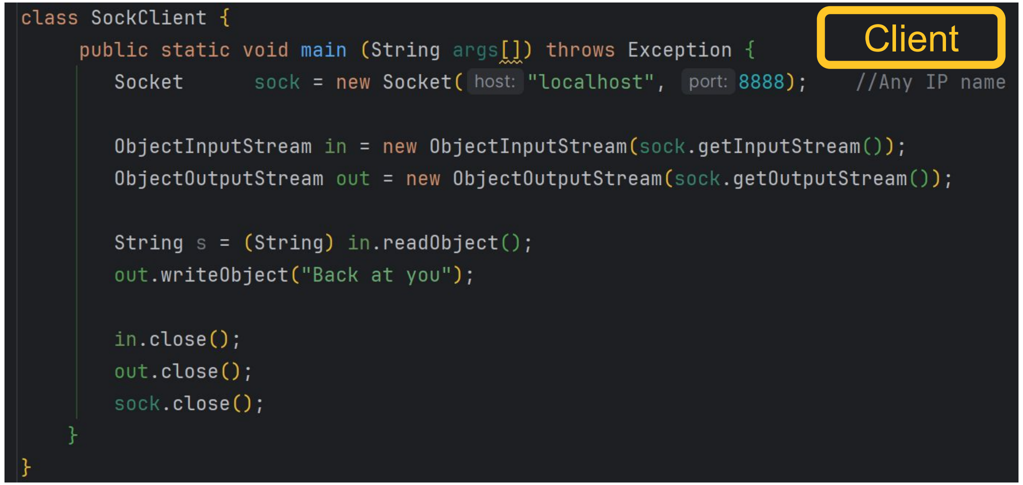
SER 321

Threading Pitfalls

As the project name implies, we encounter a ***deadlock***.

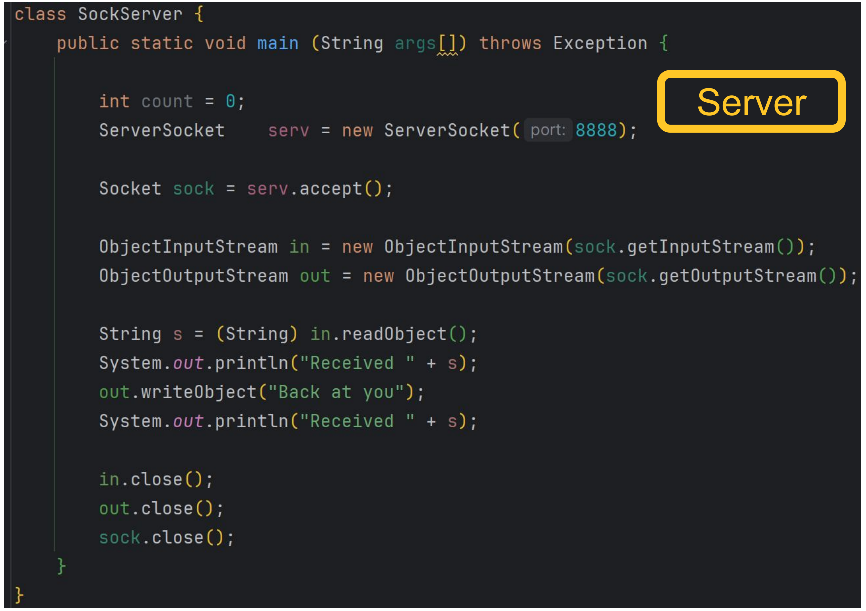
But what happened?

Client

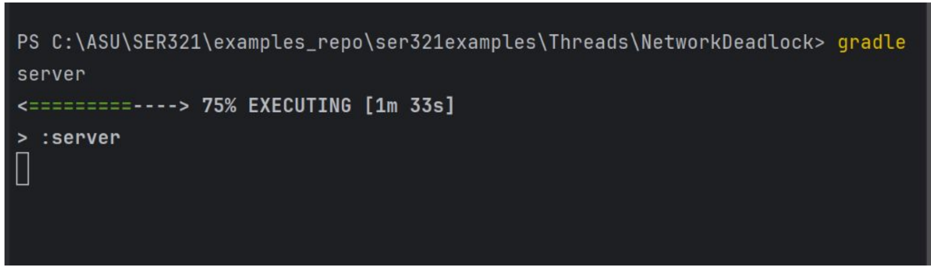


```
class SockClient {
    public static void main (String args[]) throws Exception {
        Socket      sock = new Socket( host: "localhost",  port: 8888);    //Any IP name

        ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
        ObjectOutputStream out = new ObjectOutputStream(sock.getOutputStream());

        String s = (String) in.readObject();
        out.writeObject("Back at you");

        in.close();
        out.close();
        sock.close();
    }
}
```

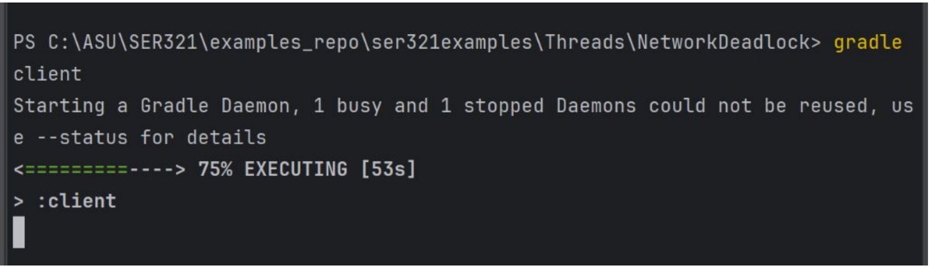
Server



```
class SockServer {
    public static void main (String args[]) throws Exception {

        int count = 0;
        ServerSocket    serv = new ServerSocket( port: 8888);

        Socket sock = serv.accept();

        ObjectInputStream in = new ObjectInputStream(sock.getInputStream());
        ObjectOutputStream out = new ObjectOutputStream(sock.getOutputStream());

        String s = (String) in.readObject();
        System.out.println("Received " + s);
        out.writeObject("Back at you");
        System.out.println("Received " + s);

        in.close();
        out.close();
        sock.close();
    }
}
```



```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
server
<=========----> 75% EXECUTING [1m 33s]
> :server
```



```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
client
Starting a Gradle Daemon, 1 busy and 1 stopped Daemons could not be reused, us
e --status for details
<=========----> 75% EXECUTING [53s]
> :client
```

Austin Walter's Traffic Comparison

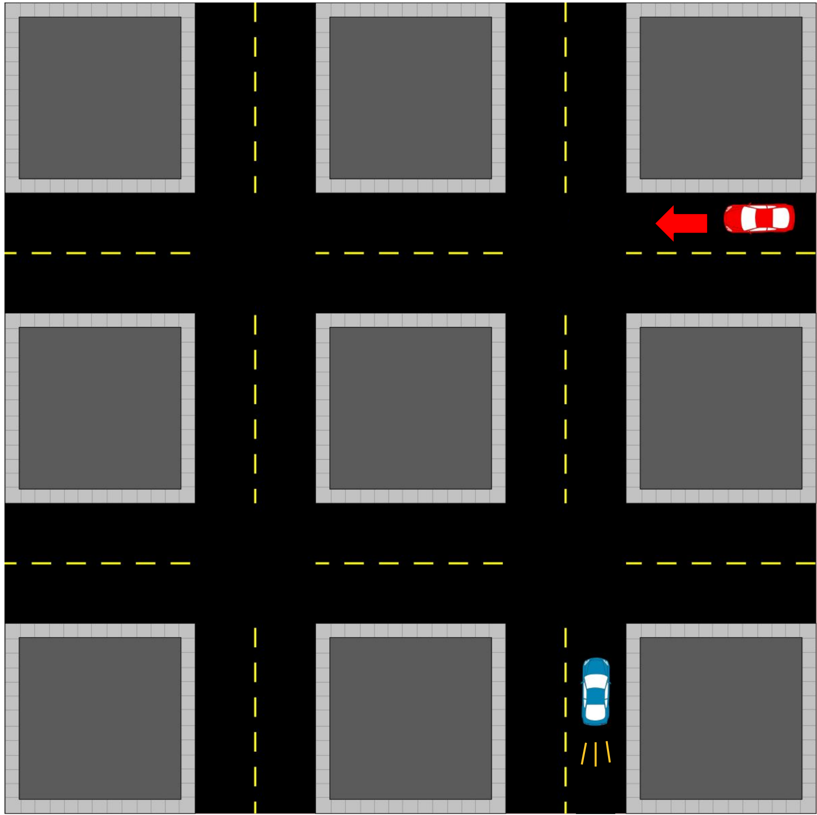
SER 321

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

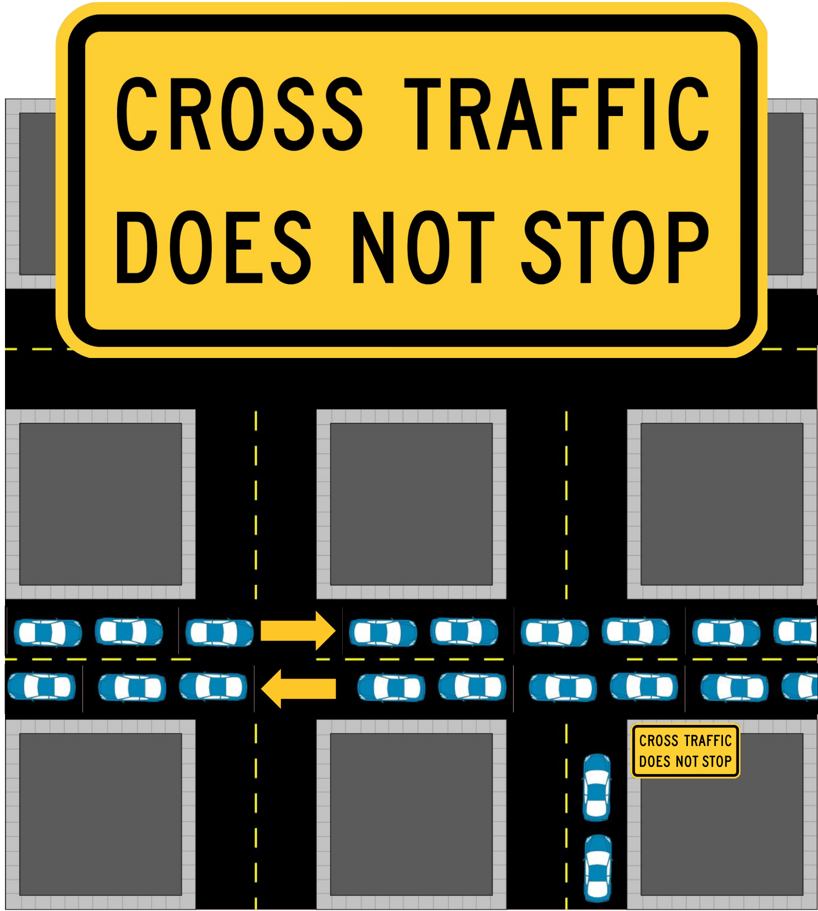
Crash

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

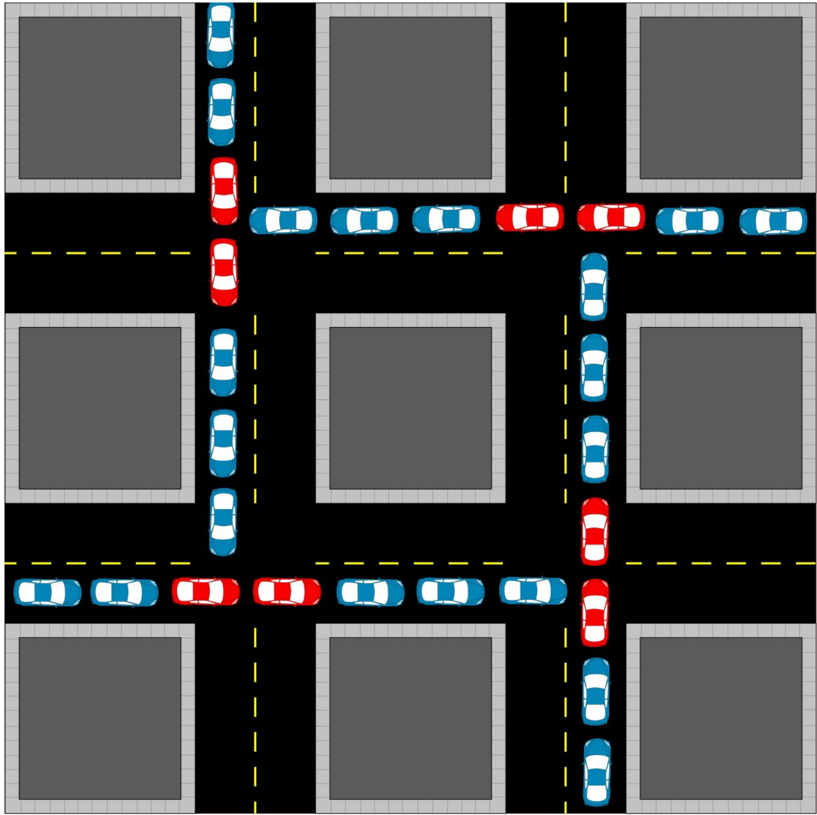
Cross Traffic

A thread never gains access to the resource it needs

Deadlock

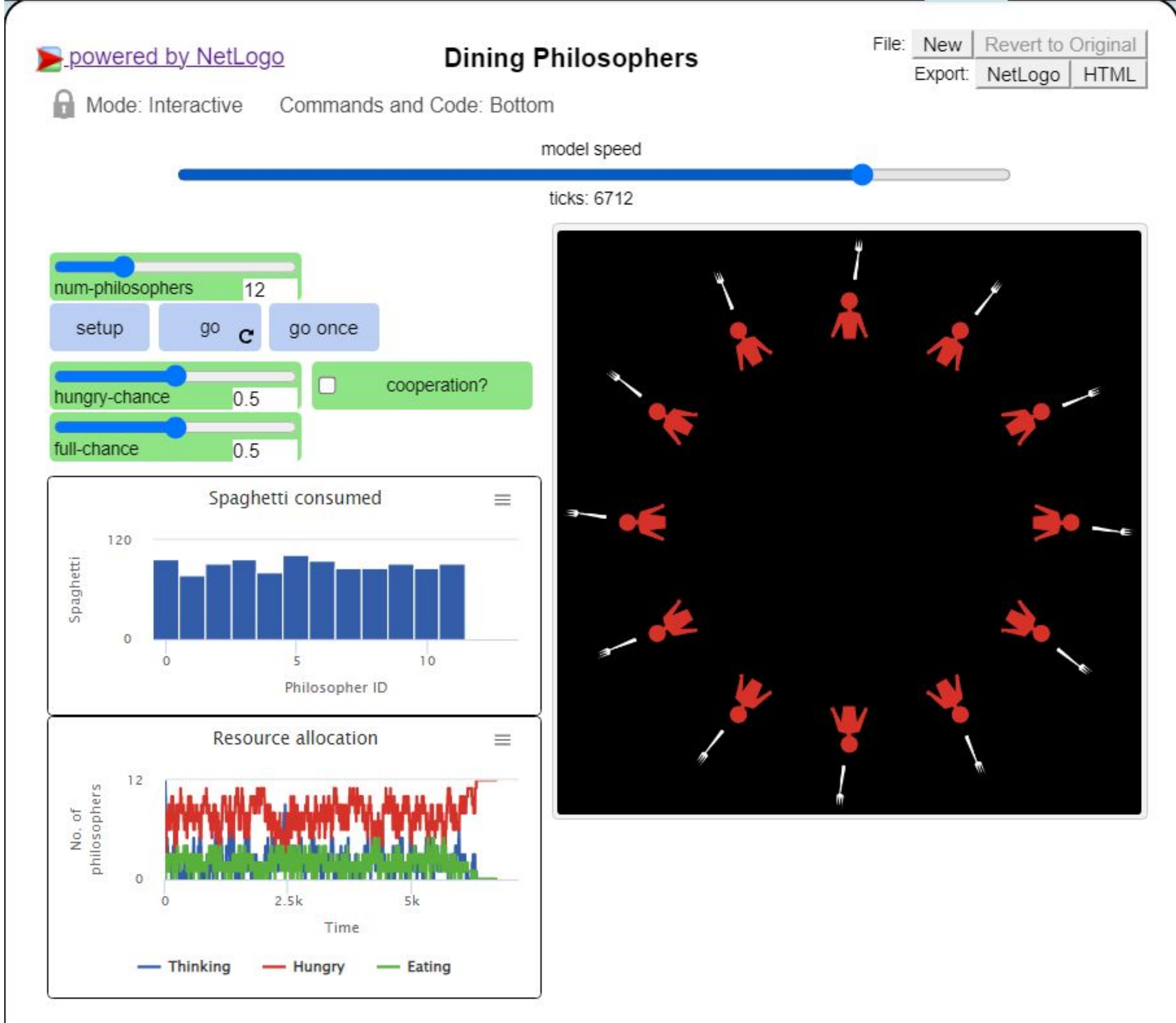
Gridlock

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



Dining Philosophers

SER 321
Threading Pitfalls



ticks: 6712

num-philosophers 12

hungry-chance 0.5

full-chance 0.5

SER 321

Scratch Space

Questions?

Survey:

http://bit.ly/ASN2324

Upcoming Events

SI Sessions:

- Monday, April 8th at 7:00 pm MST
- Thursday, April 11th at 7:00 pm MST
- Sunday, April 14th at 7:00 pm MST

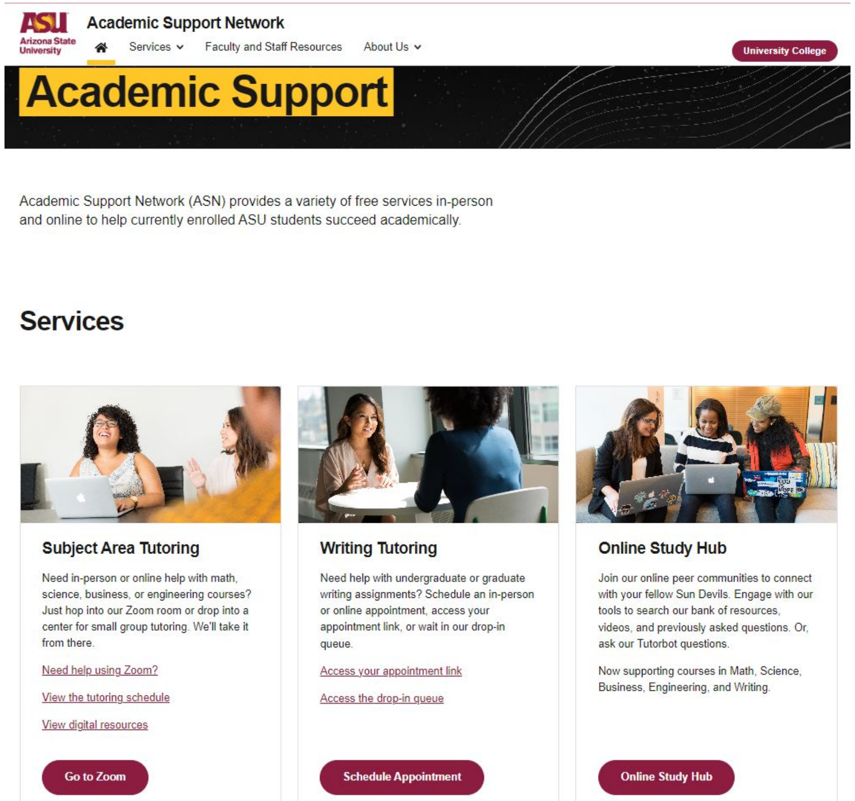
Review Sessions:

- Sunday, April 21st at 7:00 pm MST
- **Thursday, April 25th Session is *cancelled***

More Questions?

Check out our other resources!

tutoring.asu.edu


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

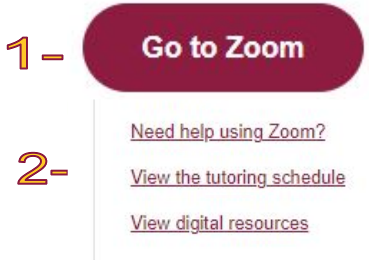
Schedule Appointment

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

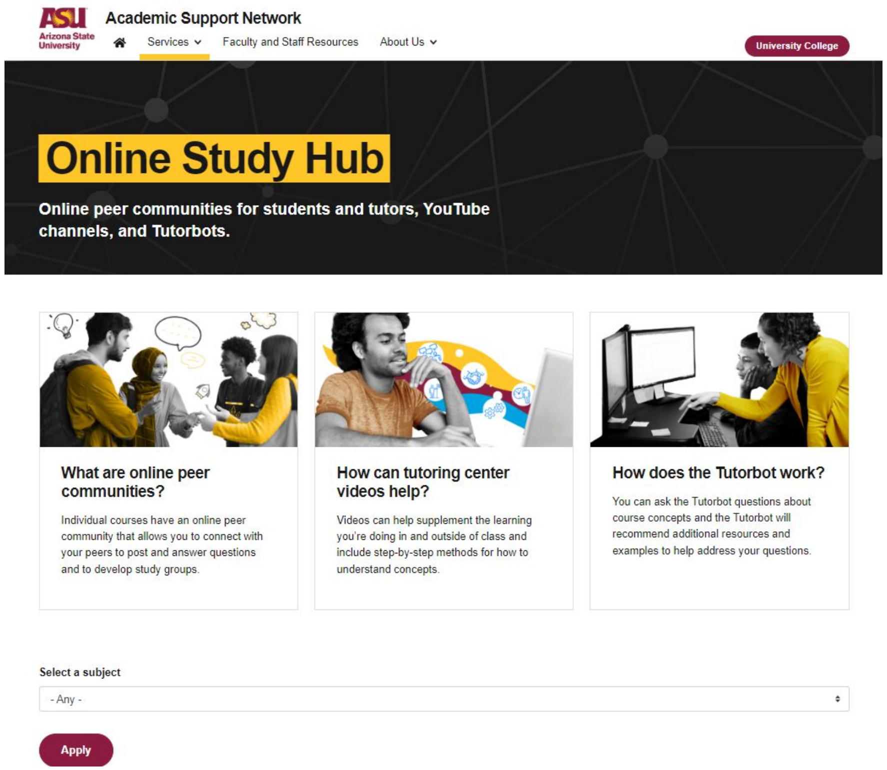


1. Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.
2. Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.

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.

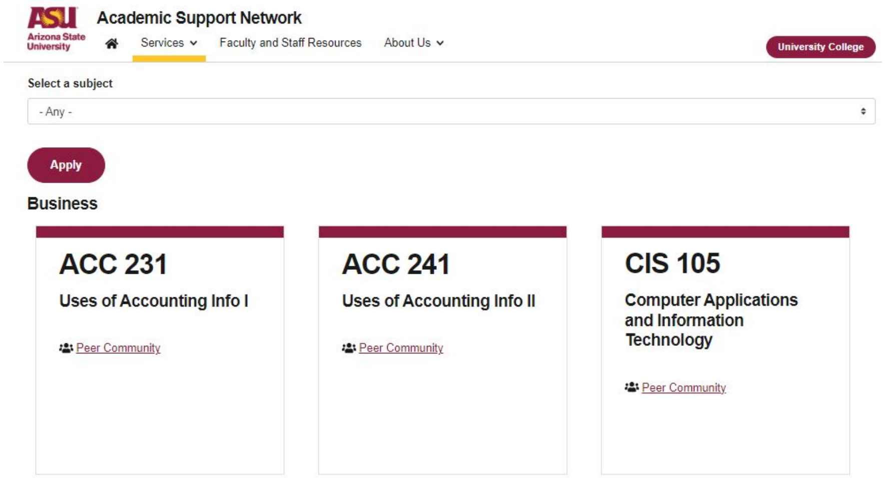
How does the Tutorbot work?

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 -

Apply



Select a subject

- Any -

Apply

Business

ACC 231
Uses of Accounting Info I
Peer Community

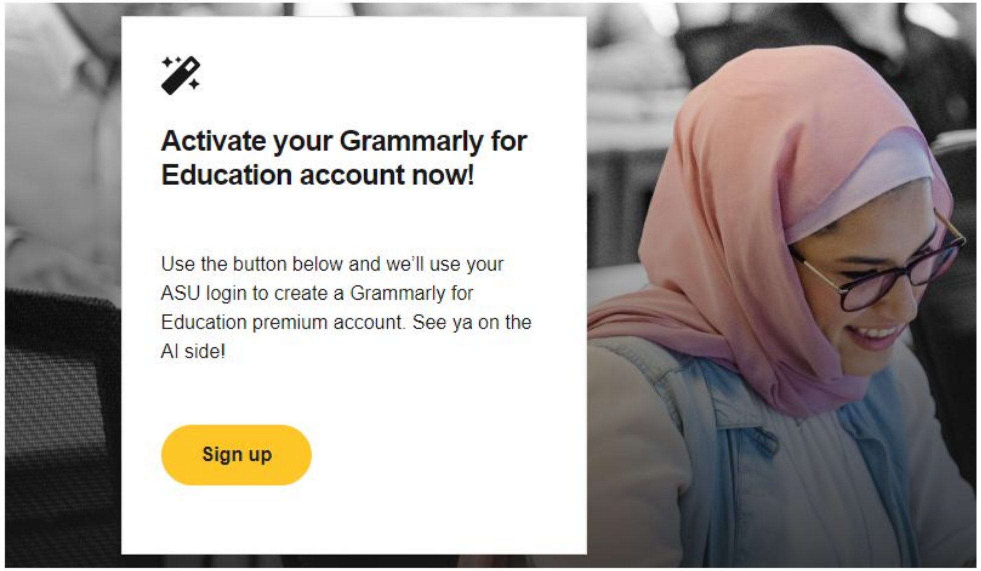
ACC 241
Uses of Accounting Info II
Peer Community

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!

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