SER 321 B Session

SI Session

Monday, April 8th 2024

7:00 pm - 8:00 pm MST

Agenda

- Protocol Buffer Speed-Run
- Threads!
  - Review Pitfalls
  - Threading Examples
  - Concurrency Constructs

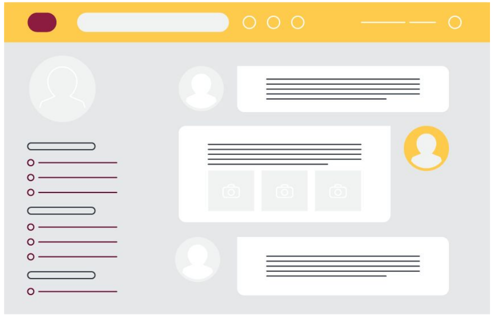
SI Session Expectations

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

How do we feel about Protocol Buffers?

SER 321

Threading Pitfalls

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
Race Condition	A thread never gains access to the resource it needs

SER 321

Threading Pitfalls

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
Race Condition	A thread never gains access to the resource it needs

What's the difference?

Starvation	vs.	Deadlock
A thread never gains access to the resource it needs		A thread is only able to acquire some of the resources it needs
Waiting to access the ***CPU***		Waiting to access the ***resource***

NetworkDeadlock

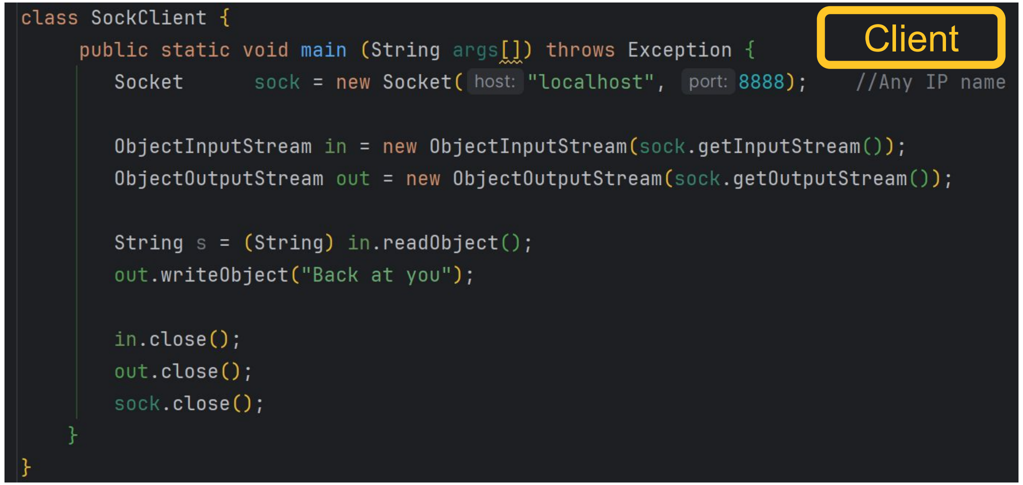
SER 321

Threading Pitfalls

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

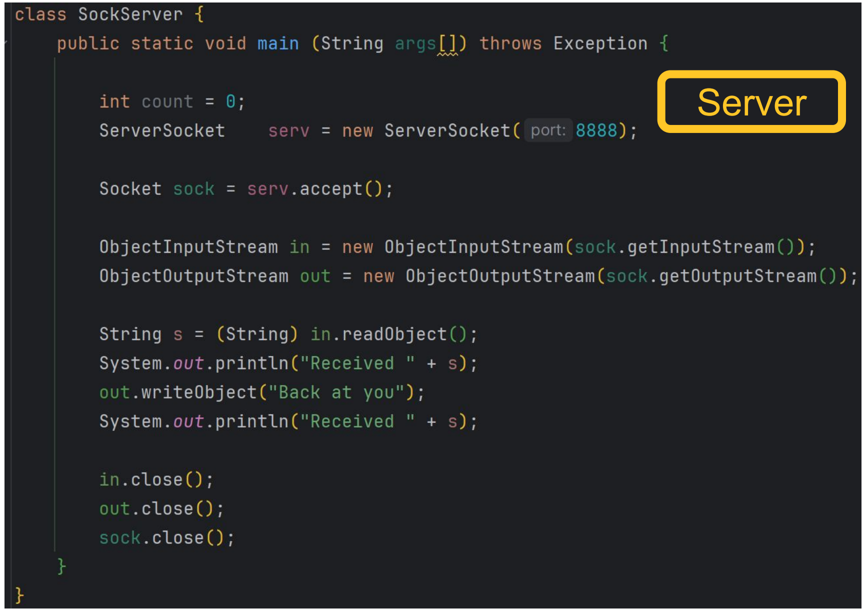
But what happened?

Client

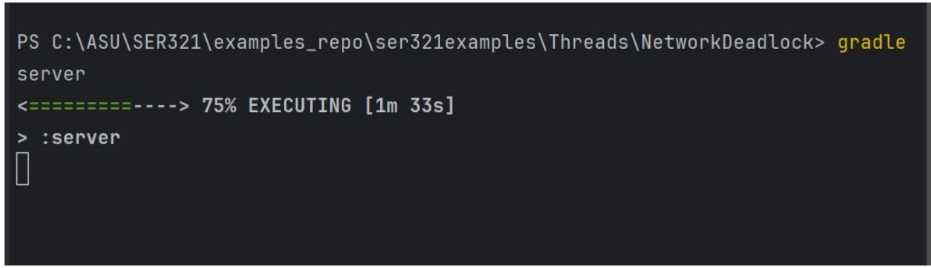


```
class SockClient {
    public static void main (String args[]) throws Exception {
        Socket      sock = new Socket( host: "localhost",  port: 8888);     //Any IP name

        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();
    }
}
```

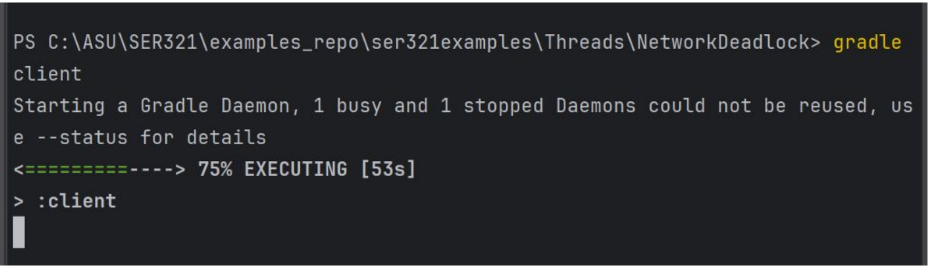
Server



```
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();
    }
}
```



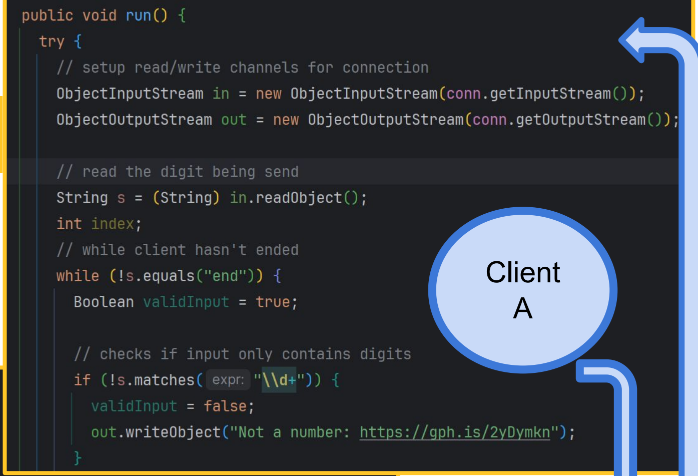
```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
server
<=========----> 75% EXECUTING [1m 33s]
> :server
```



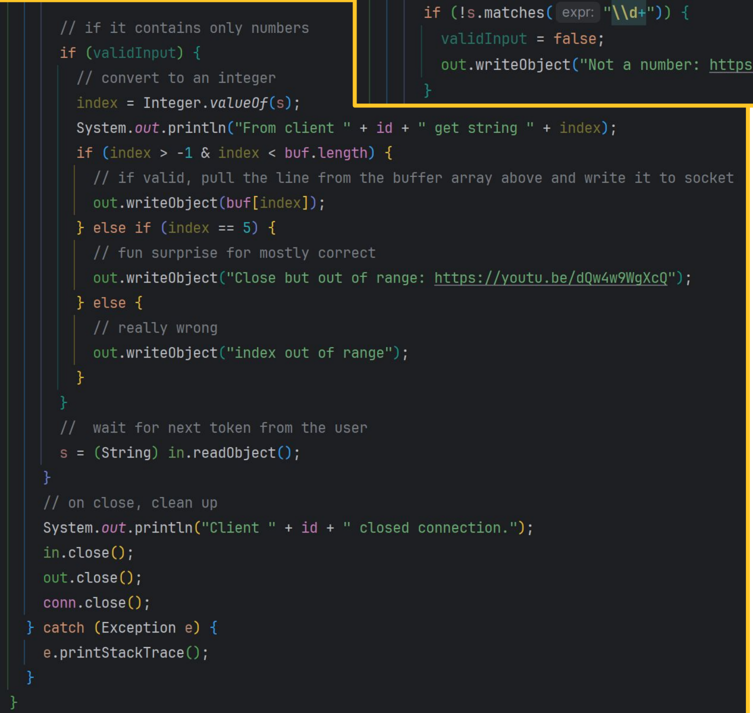
```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\NetworkDeadlock> gradle
client
Starting a Gradle Daemon, 1 busy and 1 stopped Daemons could not be reused, us
e --status for details
<=========----> 75% EXECUTING [53s]
> :client
```

JavaThreadSock

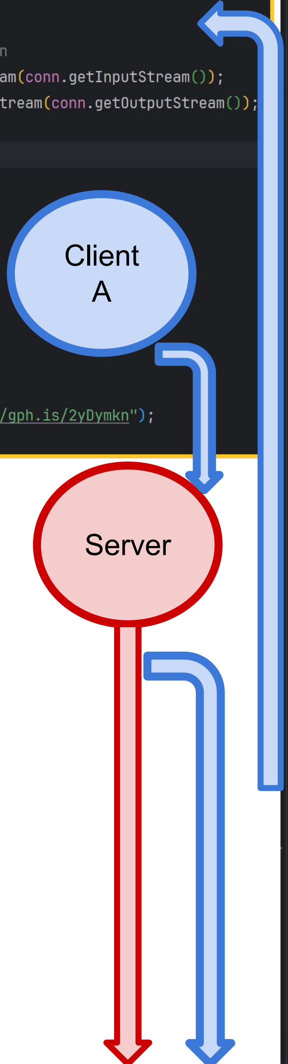
SER 321
Threads

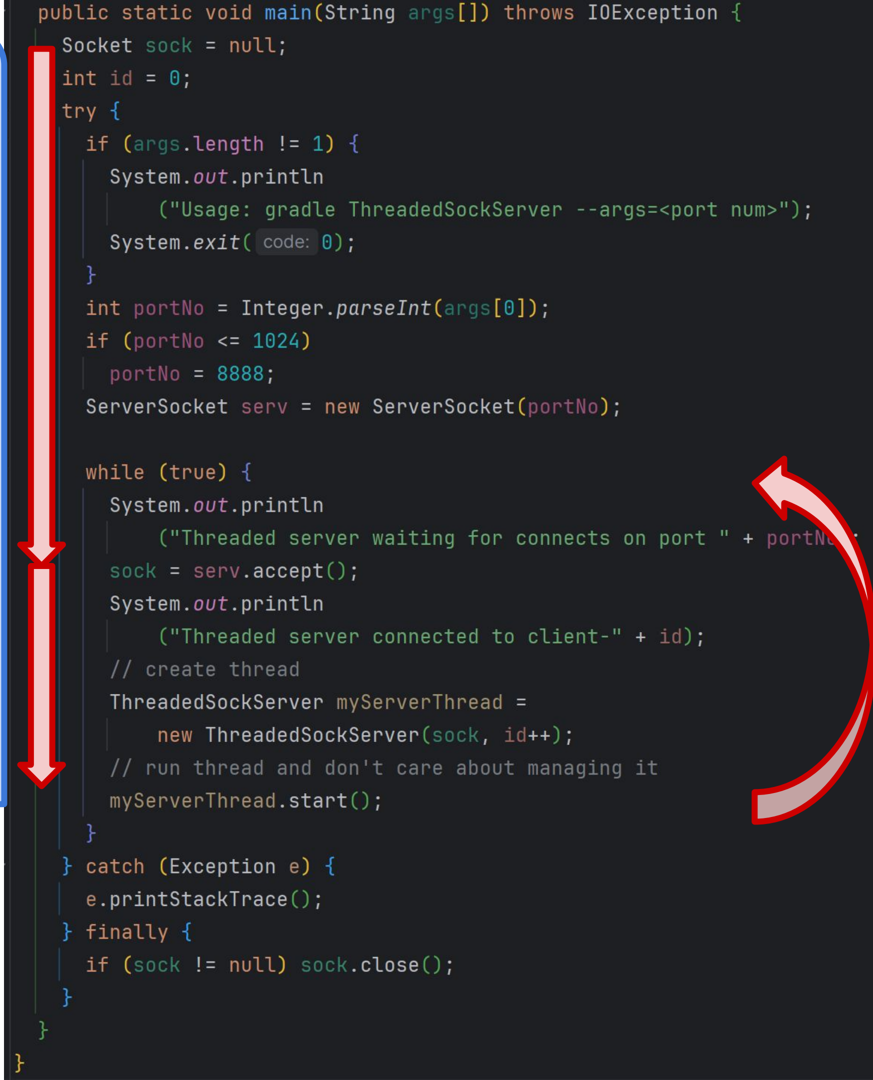


```java
public void run() {
  try {
    // setup read/write channels for connection
    ObjectInputStream in = new ObjectInputStream(conn.getInputStream());
    ObjectOutputStream out = new ObjectOutputStream(conn.getOutputStream());

    // 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();
  }
}
```

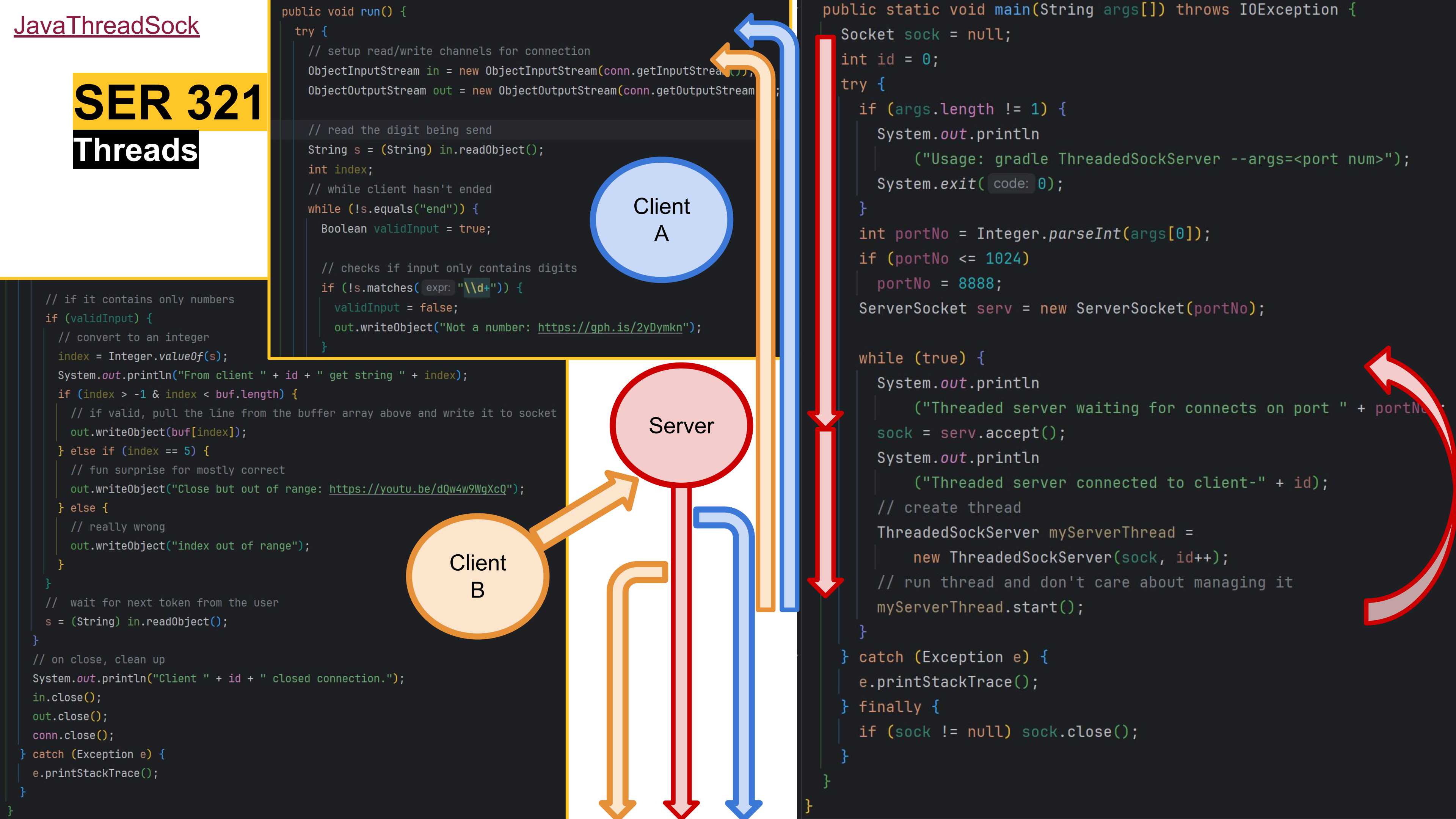




```java
public static void main(String args[]) throws IOException {
  Socket sock = null;
  int id = 0;
  try {
    if (args.length != 1) {
      System.out.println
          ("Usage: gradle ThreadedSockServer --args=<port num>");
      System.exit( code: 0);
    }
    int portNo = Integer.parseInt(args[0]);
    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



```
public void run() {
  try {
    // setup read/write channels for connection
    ObjectInputStream in = new ObjectInputStream(conn.getInputStrea
    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");
      }
```

```
      // 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 " + portN
      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

Concurrency Structures

Can we name some concurrency structures?

Atomic Operations & Variables

Locks

Semaphores

Monitors

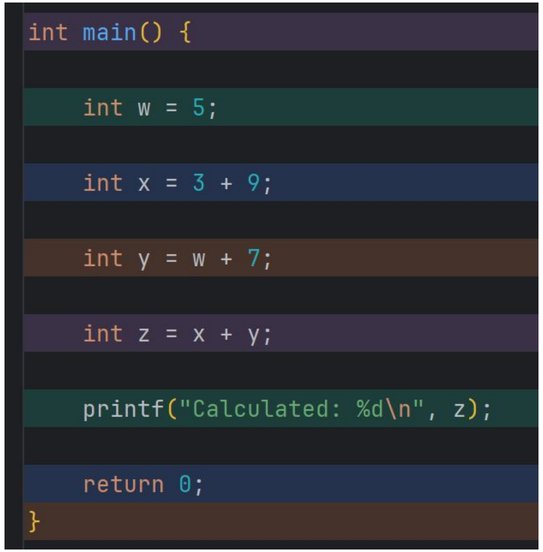
SER 321

Concurrency Structures

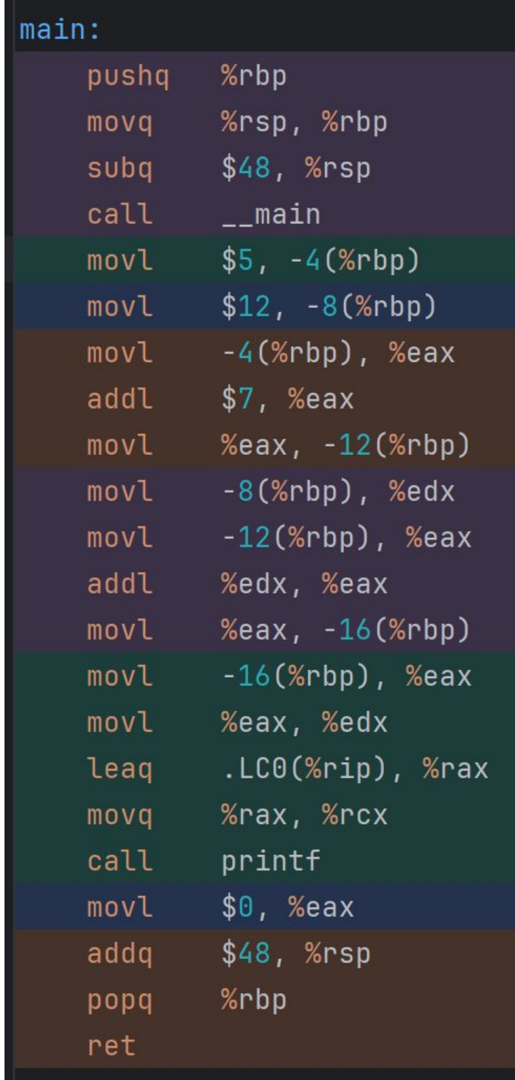
Atomic Operations & Variables

Recall *registers*…

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



```c
int main() {

    int w = 5;

    int x = 3 + 9;

    int y = w + 7;

    int z = x + y;

    printf("Calculated: %d\n", z);

    return 0;
}
```

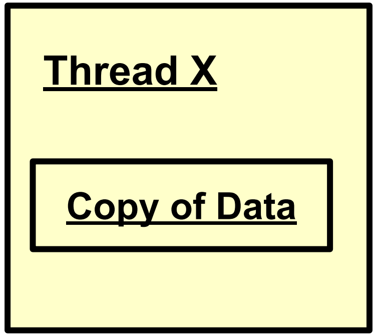


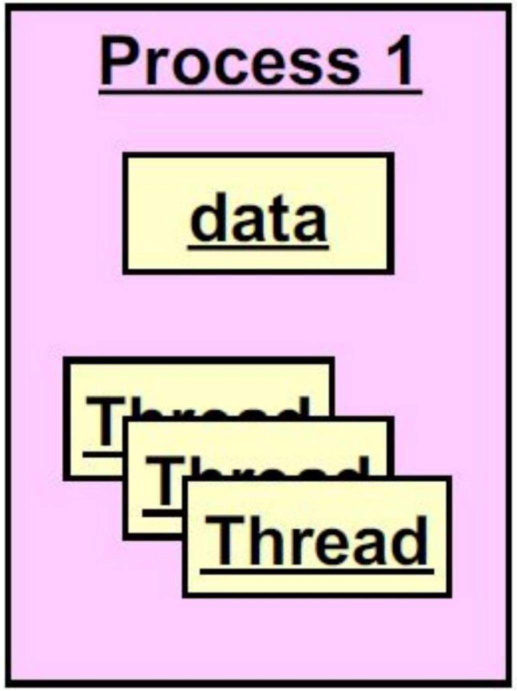
```
main:
        pushq   %rbp
        movq    %rsp, %rbp
        subq    $48, %rsp
        call    __main
        movl    $5, -4(%rbp)
        movl    $12, -8(%rbp)
        movl    -4(%rbp), %eax
        addl    $7, %eax
        movl    %eax, -12(%rbp)
        movl    -8(%rbp), %edx
        movl    -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
```

Atomic Operations & Variables

Recall *registers*…

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





The *shared* data variable is only updated if needed

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

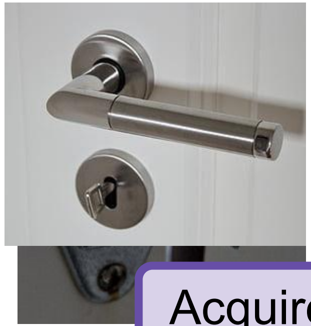
Semaphores support ***more than one*** acquirer

When would that be beneficial?

Pros and Cons?

Monitors

You lock the main door instead!



Covers the ***entire object***

Acquire Lock ⇨ { Open & Enter, Close & Lock }

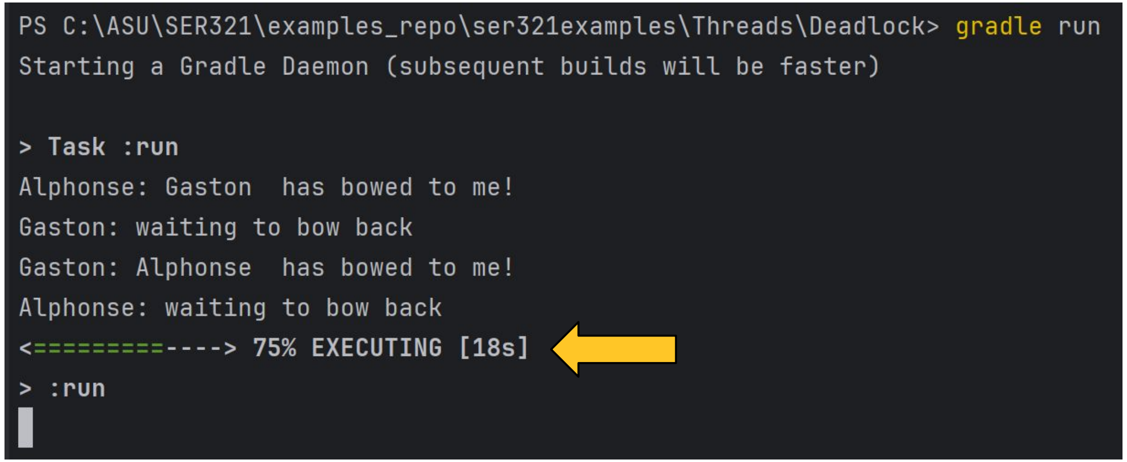
Release Lock ⇨ Unlock & Exit

Deadlock

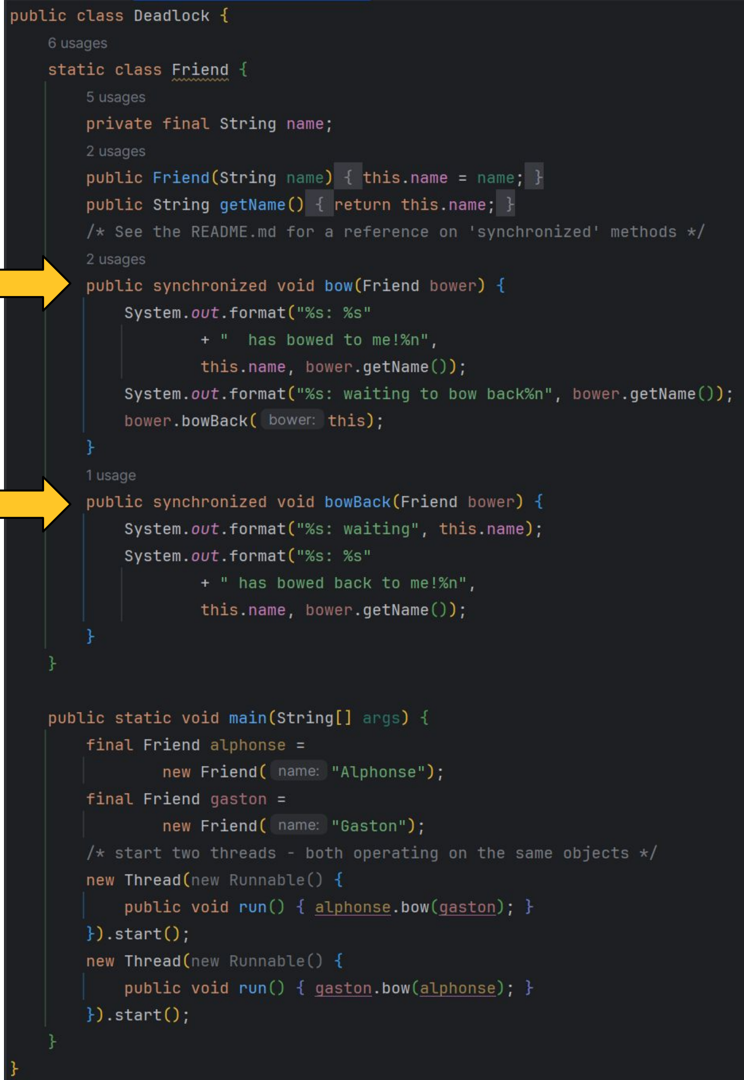
SER 321
Concurrency Structures

How can we fix this?

What happened??



```
PS C:\ASU\SER321\examples_repo\ser321examples\Threads\Deadlock> gradle run
Starting a Gradle Daemon (subsequent builds will be faster)

> Task :run
Alphonse: Gaston  has bowed to me!
Gaston: waiting to bow back
Gaston: Alphonse  has bowed to me!
Alphonse: waiting to bow back
<=========----> 75% EXECUTING [18s]
> :run
```



```java
public class Deadlock {
    6 usages
    static class Friend {
        5 usages
        private final String name;
        2 usages
        public Friend(String name) { this.name = name; }
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s"
                    + "  has bowed to me!%n",
                    this.name, bower.getName());
            System.out.format("%s: waiting to bow back%n", bower.getName());
            bower.bowBack( bower: this);
        }
        1 usage
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting", this.name);
            System.out.format("%s: %s"
                    + " has bowed back to me!%n",
                    this.name, bower.getName());
        }
    }

    public static void main(String[] args) {
        final Friend alphonse =
                new Friend( name: "Alphonse");
        final Friend gaston =
                new Friend( name: "Gaston");
        /* start two threads - both operating on the same objects */
        new Thread(new Runnable() {
            public void run() { alphonse.bow(gaston); }
        }).start();
        new Thread(new Runnable() {
            public void run() { gaston.bow(alphonse); }
        }).start();
    }
}
```

Deadlock

SER 321

Concurrency Structures

How can we fix this?

Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

Synchronize the bowBack call

```
synchronized(bower.bowBack(this));
```

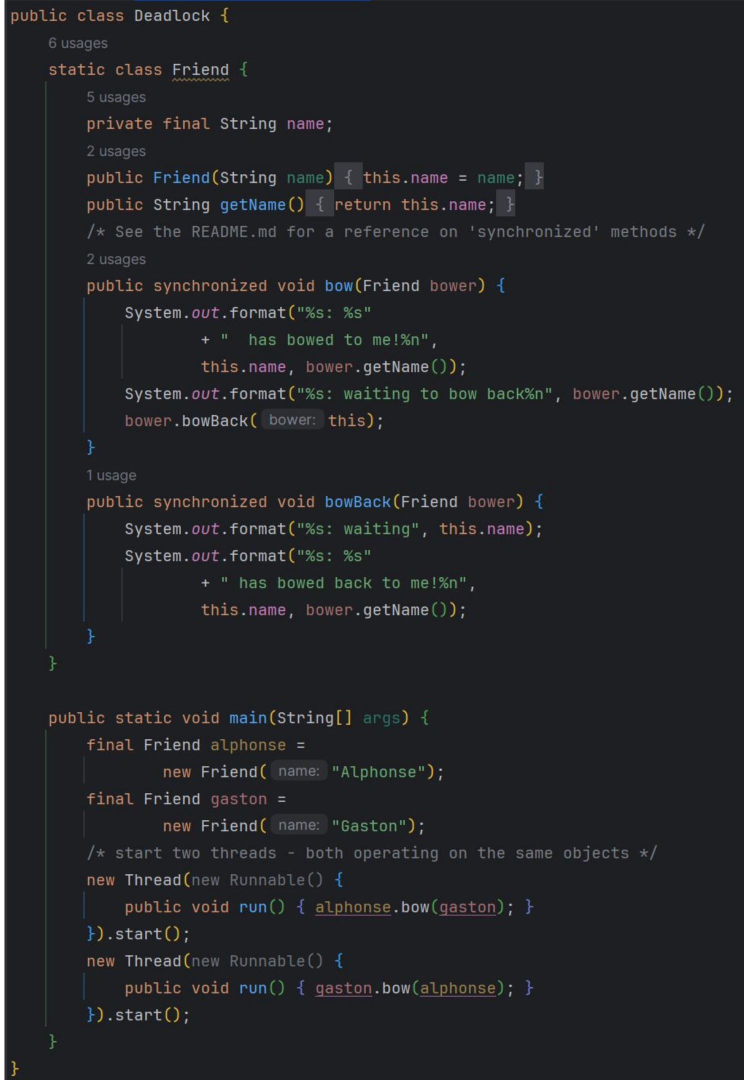
Synchronize the bowBack call with a synchronized statement

```
synchronized (this) { bower.bowBack( bower: this); }
```

Synchronize the run method calls

```
public synchronized void run() { alphonse.bow(gaston); }
```

```
public synchronized void run() { gaston.bow(alphonse); }
```



```
public class Deadlock {
    6 usages
    static class Friend {
        5 usages
        private final String name;
        2 usages
        public Friend(String name) { this.name = name; }
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s"
                    + "  has bowed to me!%n",
                    this.name, bower.getName());
            System.out.format("%s: waiting to bow back%n", bower.getName());
            bower.bowBack( bower: this);
        }
        1 usage
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting", this.name);
            System.out.format("%s: %s"
                    + " has bowed back to me!%n",
                    this.name, bower.getName());
        }
    }

    public static void main(String[] args) {
        final Friend alphonse =
                new Friend( name: "Alphonse");
        final Friend gaston =
                new Friend( name: "Gaston");
        /* start two threads - both operating on the same objects */
        new Thread(new Runnable() {
            public void run() { alphonse.bow(gaston); }
        }).start();
        new Thread(new Runnable() {
            public void run() { gaston.bow(alphonse); }
        }).start();
    }
}
```

Deadlock

SER 321

Concurrency Structures

How can we fix this?

Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

Synchronize the bowBack call

```
synchronized(bower.bowBack(this));
```

RACE

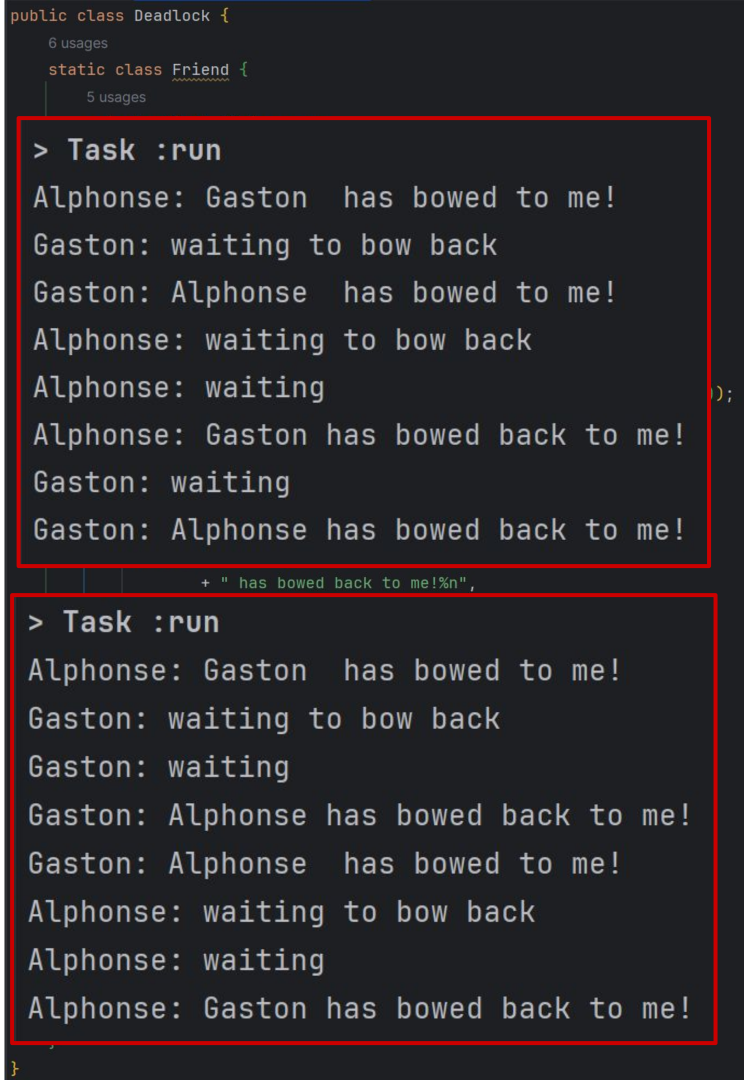
Synchronize the bowBack call with a synchronized statement

```
synchronized (this) { bower.bowBack( bower: this); }
```

Synchronize the run method calls

```
public synchronized void run() { alphonse.bow(gaston); }
```

```
public synchronized void run() { gaston.bow(alphonse); }
```



```
public class Deadlock {
    6 usages
    static class Friend {
        5 usages
```

```
> Task :run
Alphonse: Gaston  has bowed to me!
Gaston: waiting to bow back
Gaston: Alphonse  has bowed to me!
Alphonse: waiting to bow back
Alphonse: waiting
Alphonse: Gaston has bowed back to me!
Gaston: waiting
Gaston: Alphonse has bowed back to me!
```

```
                    + " has bowed back to me!%n",
```

```
> Task :run
Alphonse: Gaston  has bowed to me!
Gaston: waiting to bow back
Gaston: waiting
Gaston: Alphonse has bowed back to me!
Gaston: Alphonse  has bowed to me!
Alphonse: waiting to bow back
Alphonse: waiting
Alphonse: Gaston has bowed back to me!
```

```
}
```

Deadlock

SER 321
Concurrency Structures

How can we fix this?

Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

Synchronize the bowBack call

```
synchronized(bower.bowBack(this));
```

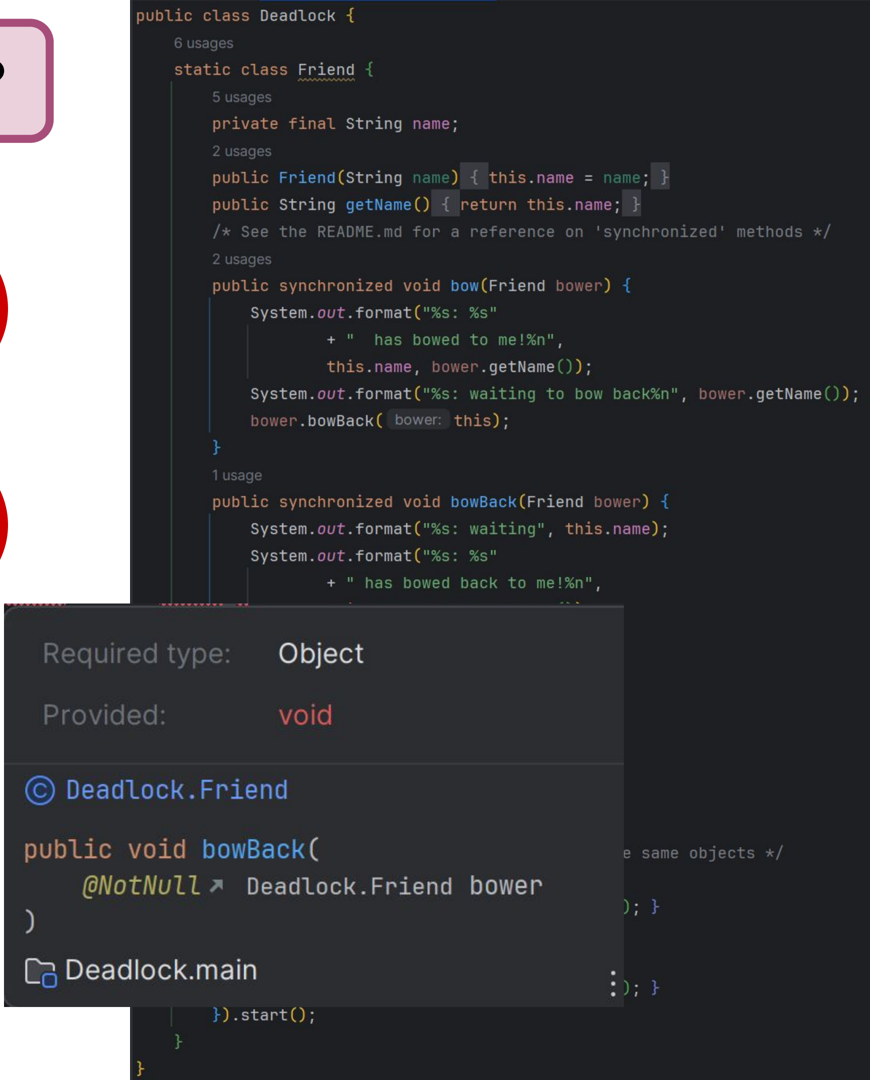
Synchronize the bowBack call with a synchronized statement

```
synchronized (this) { bower.bowBack( bower: this); }
```

Synchronize the run method calls

```
public synchronized void run() { alphonse.bow(gaston); }
```

```
public synchronized void run() { gaston.bow(alphonse); }
```



```
public class Deadlock {
    6 usages
    static class Friend {
        5 usages
        private final String name;
        2 usages
        public Friend(String name) { this.name = name; }
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages
        public synchronized void bow(Friend bower) {
            System.out.format("%s: %s"
                    + "  has bowed to me!%n",
                    this.name, bower.getName());
            System.out.format("%s: waiting to bow back%n", bower.getName());
            bower.bowBack( bower: this);
        }
        1 usage
        public synchronized void bowBack(Friend bower) {
            System.out.format("%s: waiting", this.name);
            System.out.format("%s: %s"
                    + " has bowed back to me!%n",
```

```
Required type:   Object
Provided:        void

Deadlock.Friend

public void bowBack(
    @NotNull Deadlock.Friend bower
)

Deadlock.main
```

```
e same objects */
); }
); }
            }).start();
        }
}
```

Deadlock

SER 321

Concurrency Structures

How can we fix this?

Remove the synchronized methods 🚫

```
public void bow
```

```
public void bowBack
```

Synchronize the bowBack call 🚫

```
synchronized(bower.bowBack(this));
```

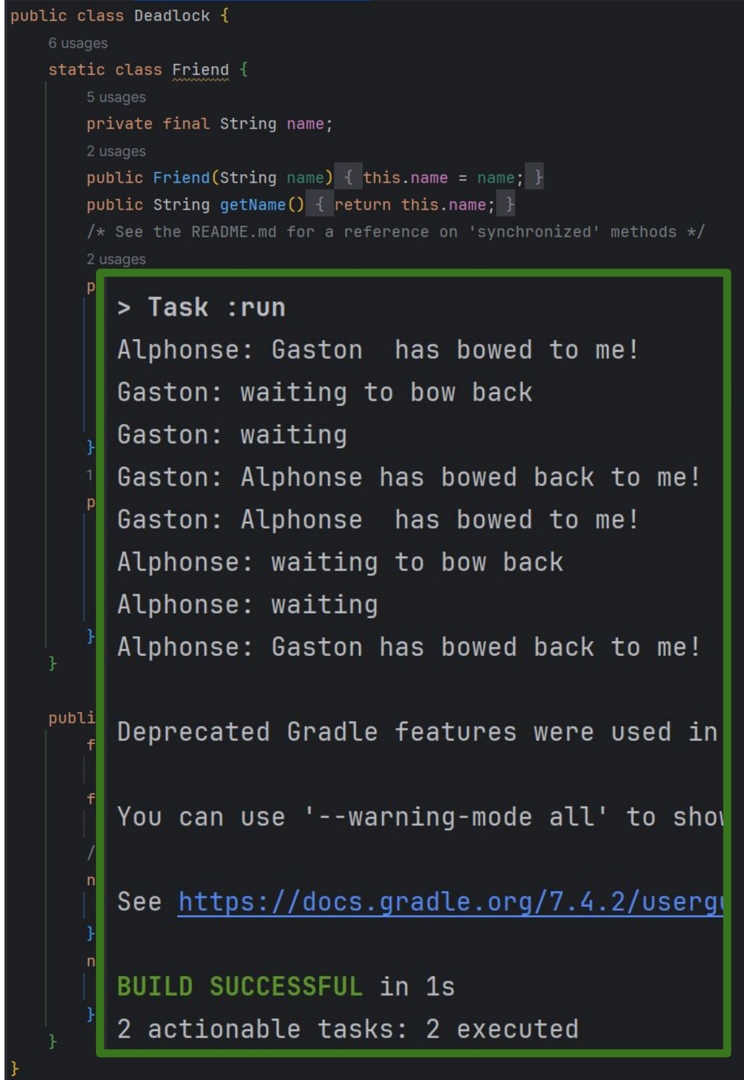
Synchronize the bowBack call with a synchronized statement ✓

```
synchronized (this) { bower.bowBack( bower: this); }
```

Synchronize the run method calls ✓

```
public synchronized void run() { alphonse.bow(gaston); }
```

```
public synchronized void run() { gaston.bow(alphonse); }
```



```
public class Deadlock {
    6 usages
    static class Friend {
        5 usages
        private final String name;
        2 usages
        public Friend(String name) { this.name = name; }
        public String getName() { return this.name; }
        /* See the README.md for a reference on 'synchronized' methods */
        2 usages
```

```
> Task :run
Alphonse: Gaston  has bowed to me!
Gaston: waiting to bow back
Gaston: waiting
Gaston: Alphonse has bowed back to me!
Gaston: Alphonse  has bowed to me!
Alphonse: waiting to bow back
Alphonse: waiting
Alphonse: Gaston has bowed back to me!

Deprecated Gradle features were used in

You can use '--warning-mode all' to show

See https://docs.gradle.org/7.4.2/usergu

BUILD SUCCESSFUL in 1s
2 actionable tasks: 2 executed
```

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

Scratch Space

Questions?

Survey:

http://bit.ly/ASN2324

Upcoming Events

SI Sessions:

- Thursday, April 11th at 7:00 pm MST
- Sunday, April 14th at 7:00 pm MST
- Monday, April 15th at 7:00 pm MST

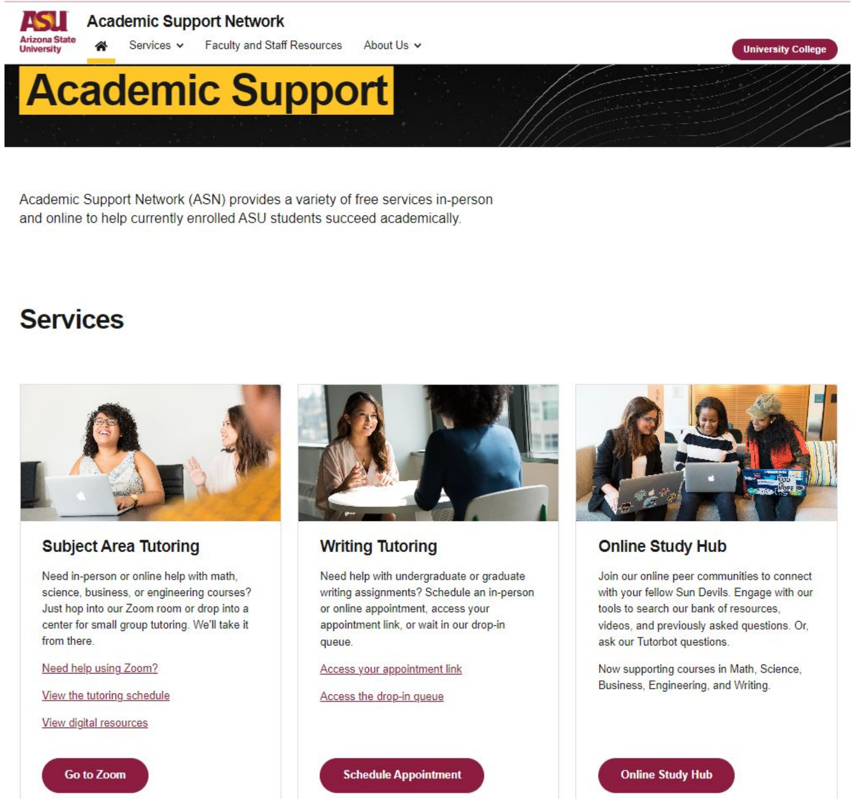
Review Sessions:

- 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

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

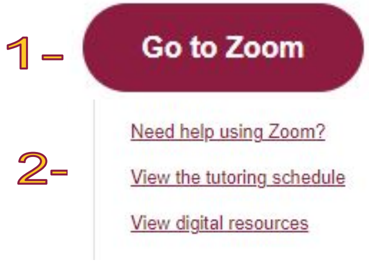
Schedule Appointment

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

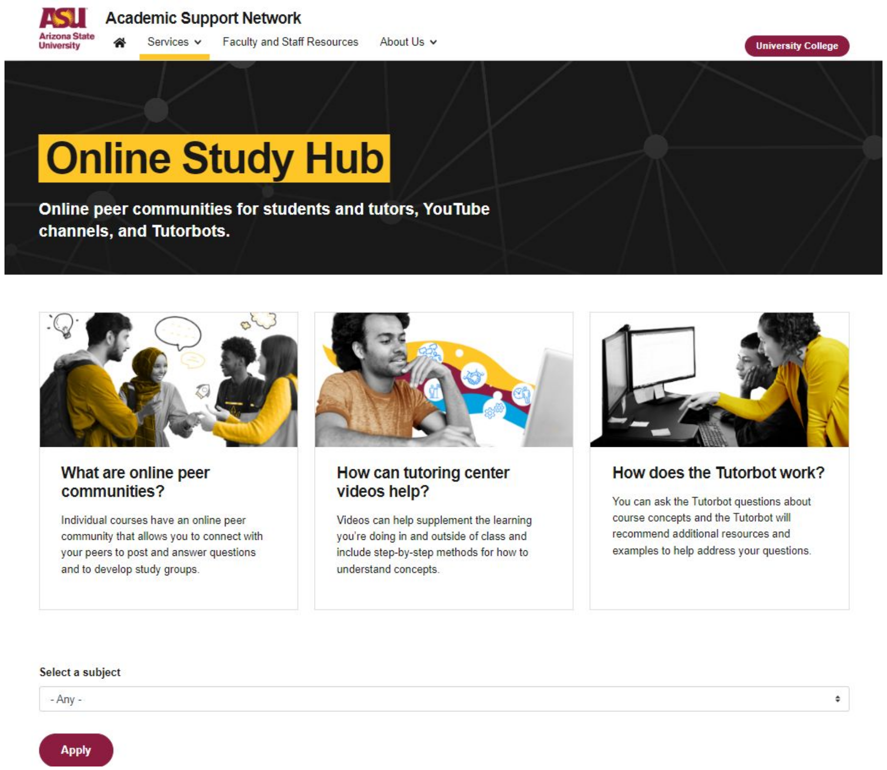


1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

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.

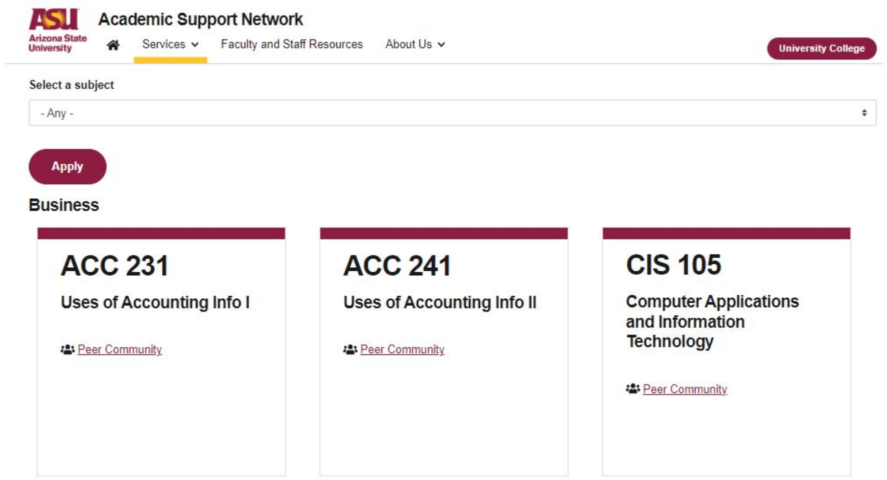
How can tutoring center videos help?

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

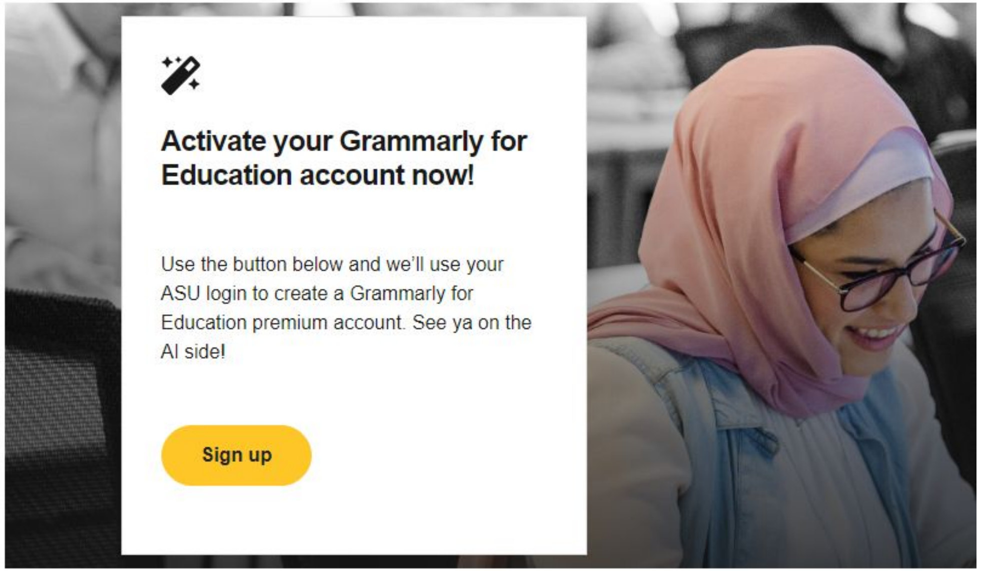
ACC 241 Uses of Accounting Info II — Peer Community

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
- Dining Philosophers Interactive
- Austin G Walters Traffic Comparison