SER 321 C Session

SI Session

Sunday, June 16th 2024

6:00 pm - 7:00 pm MST

Agenda

- Review Threading Pitfalls
- Concurrency Structures
    - Structure Analogy
    - Sample Problem: Deadlock
- Threading your Code

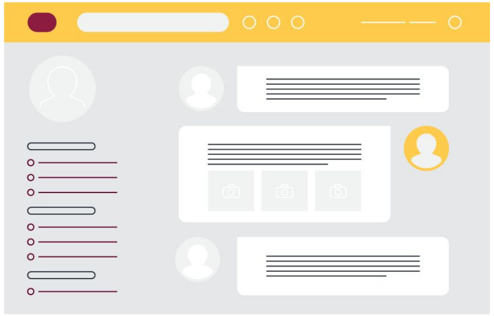
SI Session Expectations

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



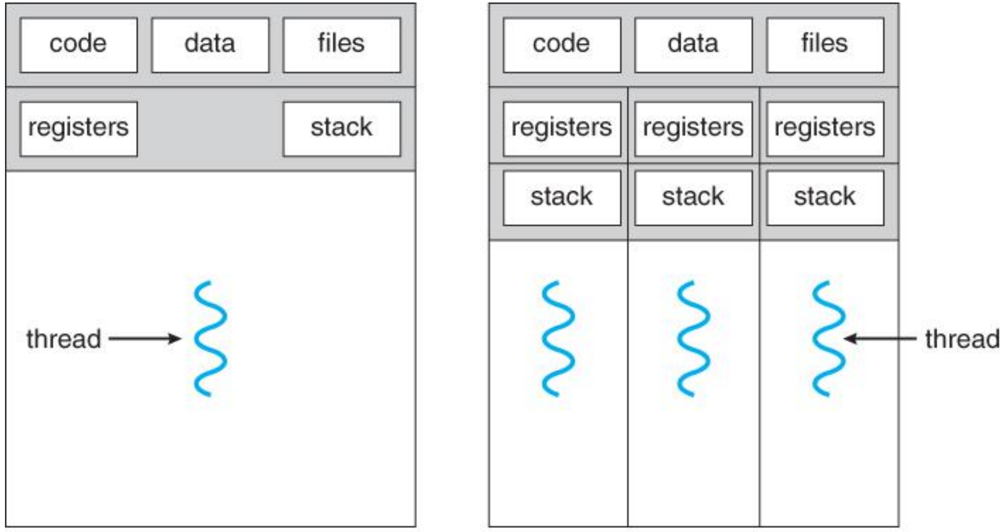
Zoom Chat

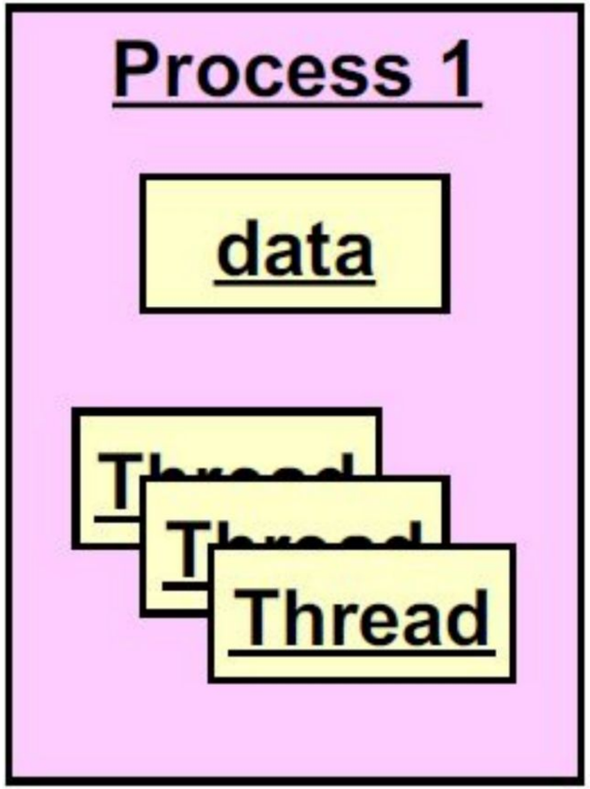
- Use the chat feature to interact with the presenter and respond to presenter's questions.
- Annotations are encouraged

SER 321
Threads

What does that imply?

Remember that they exist *within* the parent process





SER 321

Threading Pitfalls

Race Condition	A thread is only able to acquire some of the resources it needs
Starvation	More than one thread accesses a single resource at the same time
Deadlock	A thread never gains access to the resource it needs

SER 321

Threading Pitfalls

Race Condition

Starvation

Deadlock

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

A thread never gains access to the resource it needs

SER 321

Concurrency Structures

Can we name some concurrency structures?

Atomic Operations & Variables

Locks

Semaphores

Monitors

SER 321

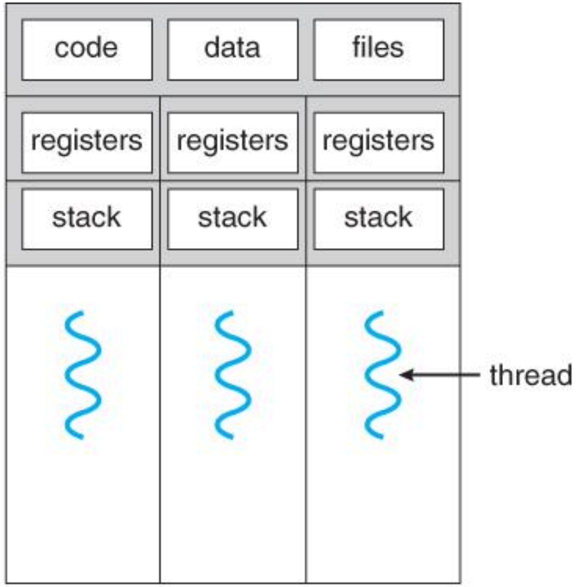
Concurrency Structures

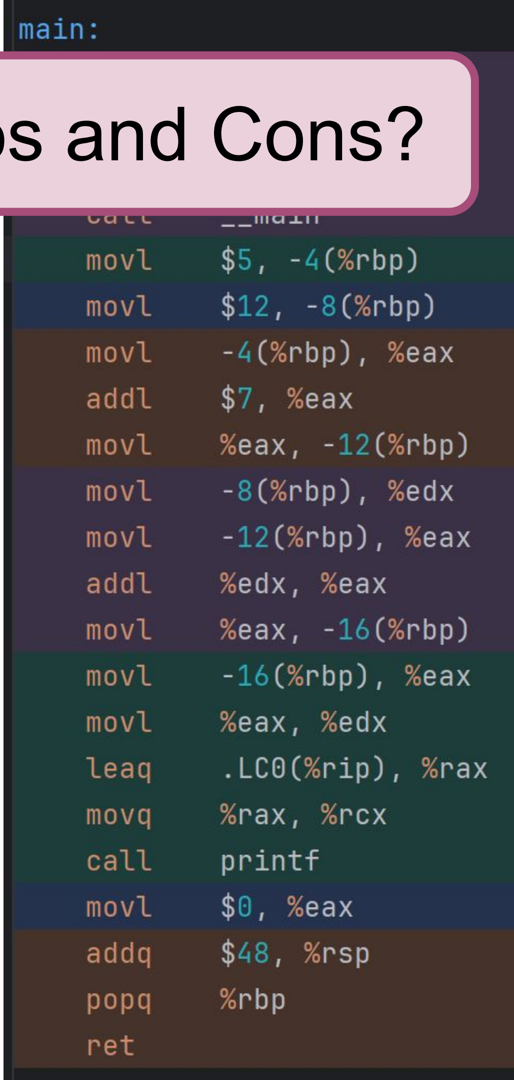
Pros and Cons?

Atomic Operations & Variables

Recall *registers*…

Volatile keyword ensures updates are immediately visible for the local copy in *each and every thread*





```
main:
        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
```

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

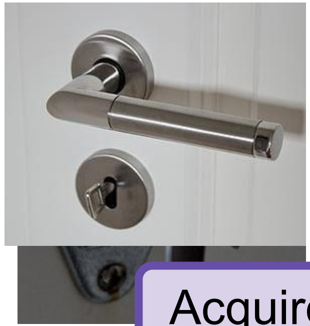
When would that be beneficial?

SER 321
Concurrency Structures

Pros and Cons?

Monitors

You lock the main door instead!



Covers the ***entire object***

Acquire Lock → Open & Enter, Close & Lock

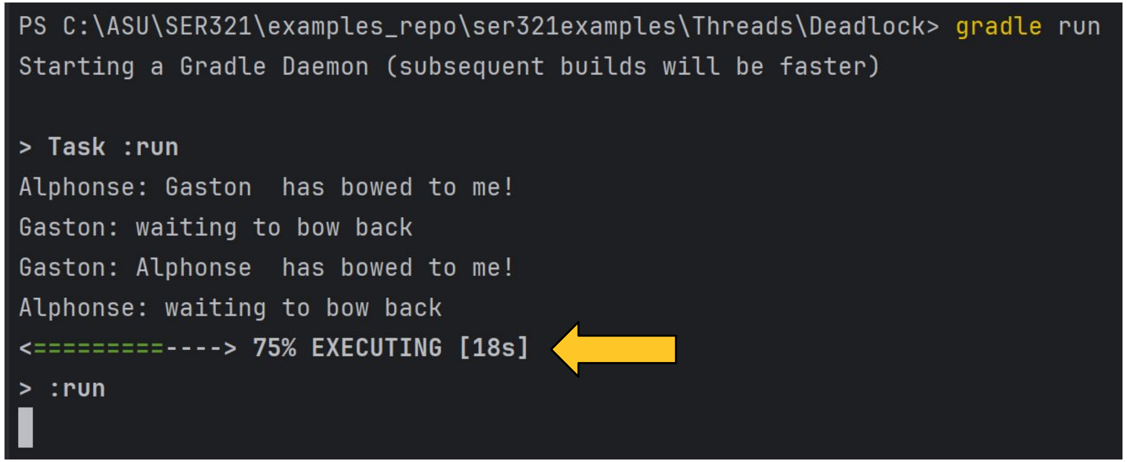
Release Lock → Unlock & Exit

Deadlock

SER 321
Concurrency Structures

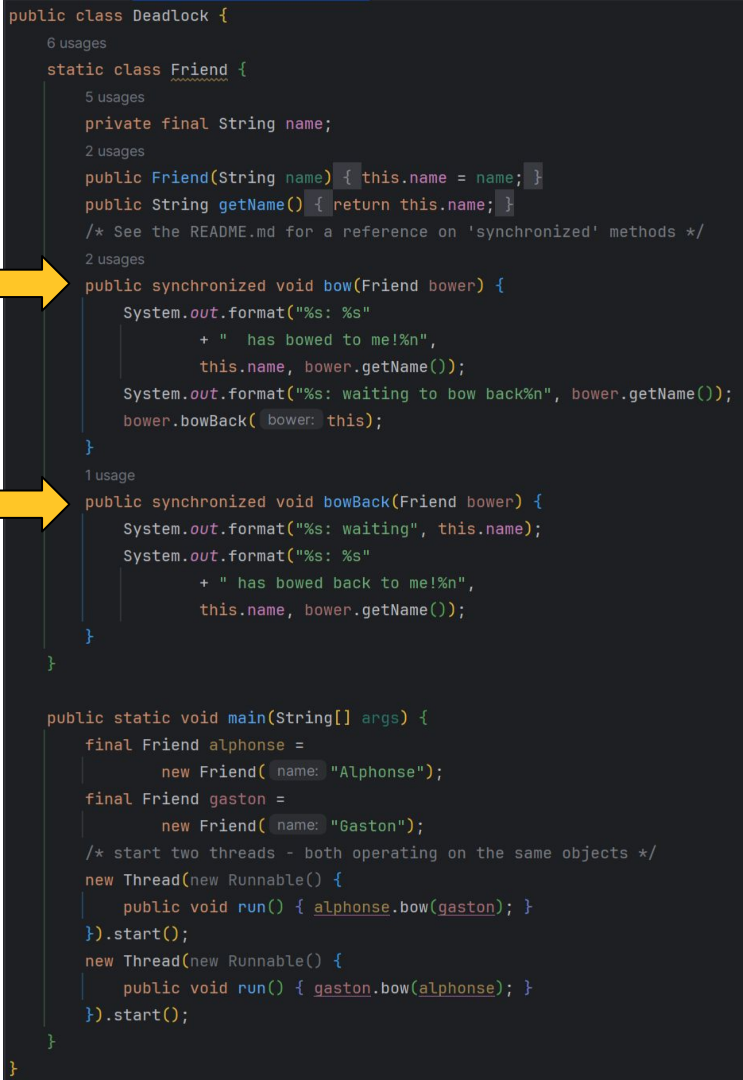
How can we fix this?

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();
    }
}
```

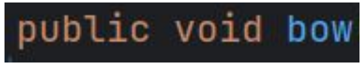
Deadlock

SER 321

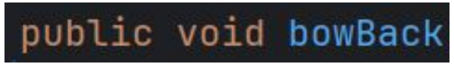
Concurrency Structures

Options to fix this?

1. Remove the synchronized methods

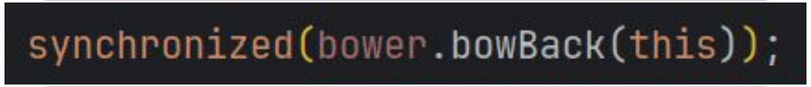


```
public void bow
```



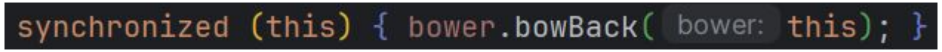
```
public void bowBack
```

2. Synchronize the bowBack call



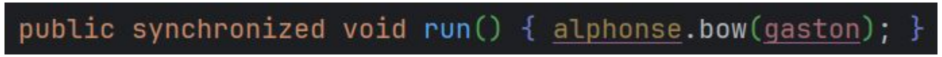
```
synchronized(bower.bowBack(this));
```

3. Synchronize the bowBack call with a synchronized statement

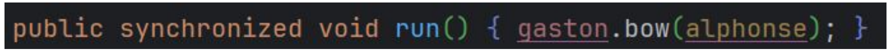


```
synchronized (this) { bower.bowBack( bower: this); }
```

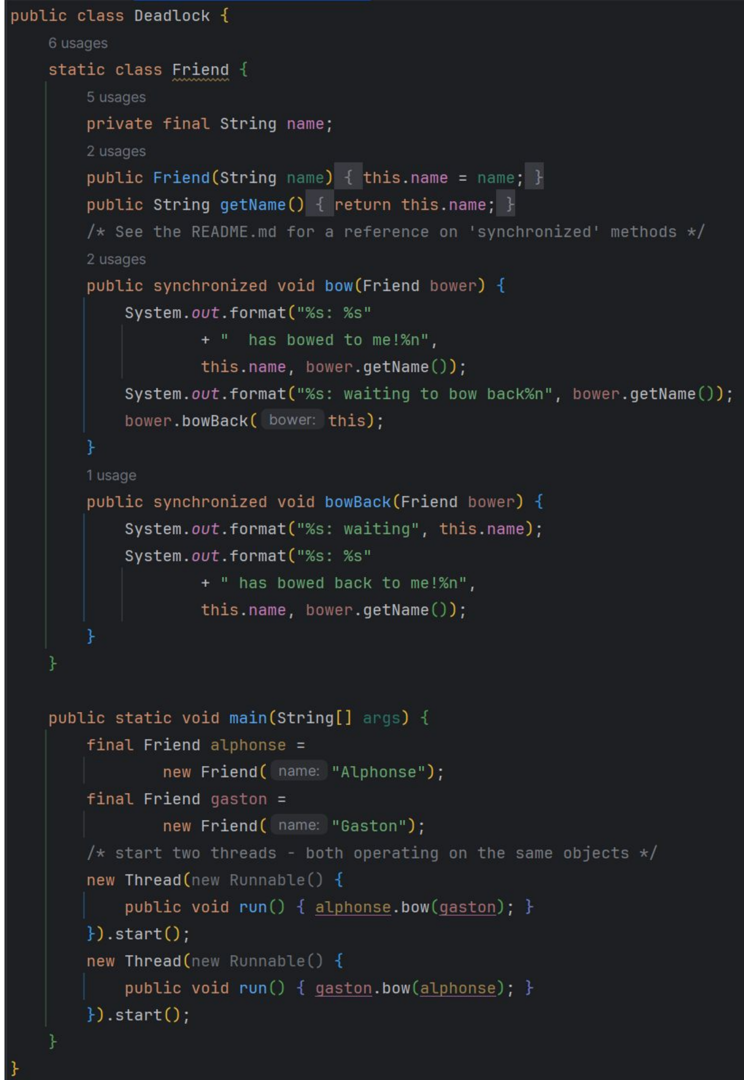
4. Synchronize the run method calls



```
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

How can we fix this?

SER 321

Concurrency Structures

1. Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

2. Synchronize the bowBack call

RACE

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

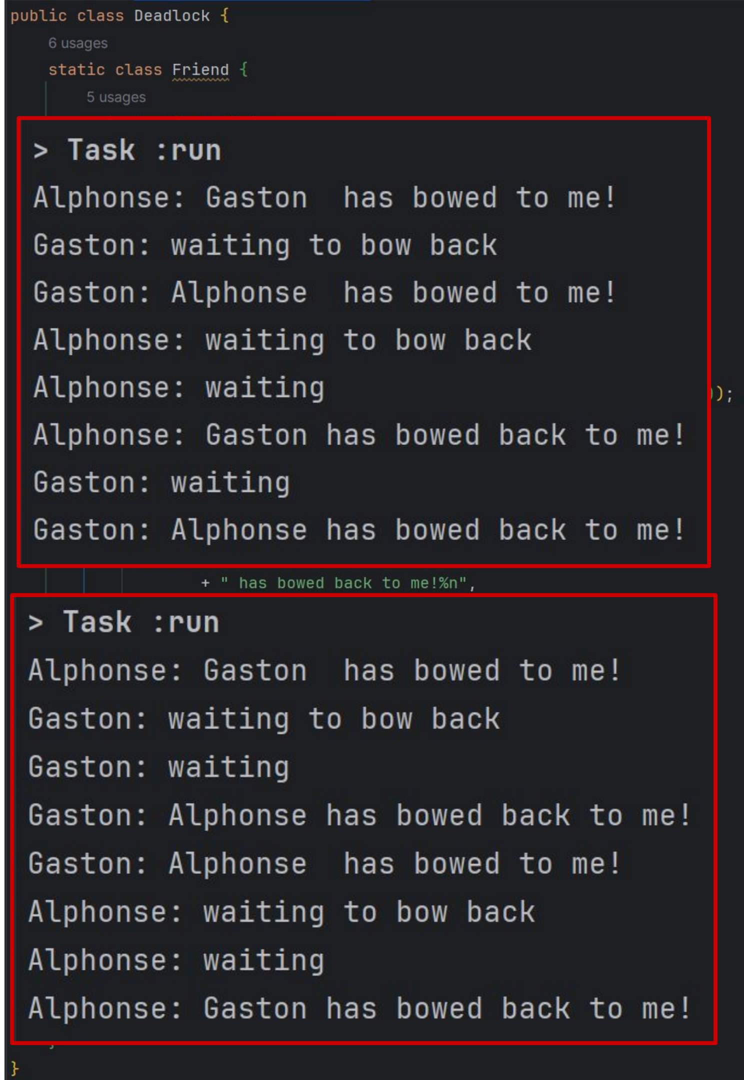
3. Synchronize the bowBack call with a synchronized statement

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

4. 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?

1. Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

2. Synchronize the bowBack call

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

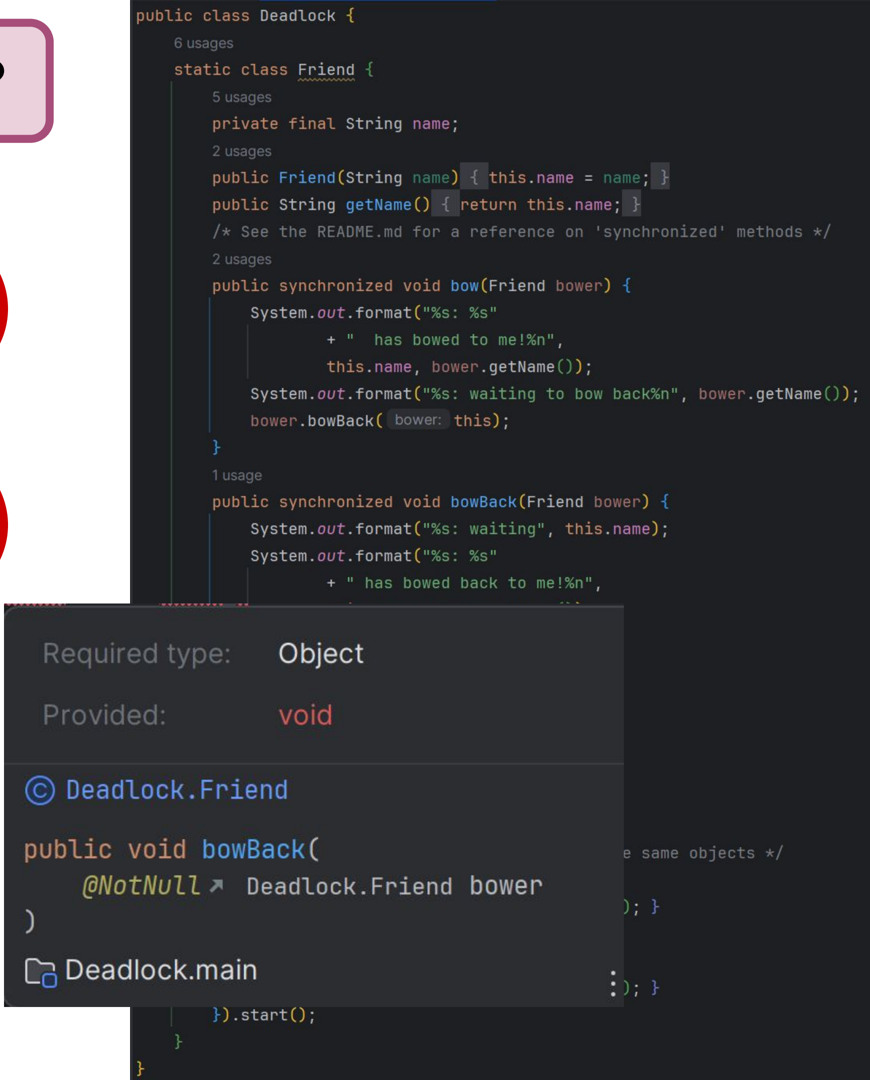
3. Synchronize the bowBack call with a synchronized statement

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

4. 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",
```

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

```
Required type:    Object
Provided:         void

Deadlock.Friend
public void bowBack(
    @NotNull Deadlock.Friend bower
)
Deadlock.main
```

Deadlock

SER 321

Concurrency Structures

How can we fix this?

1. Remove the synchronized methods

```
public void bow
```

```
public void bowBack
```

2. Synchronize the bowBack call

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

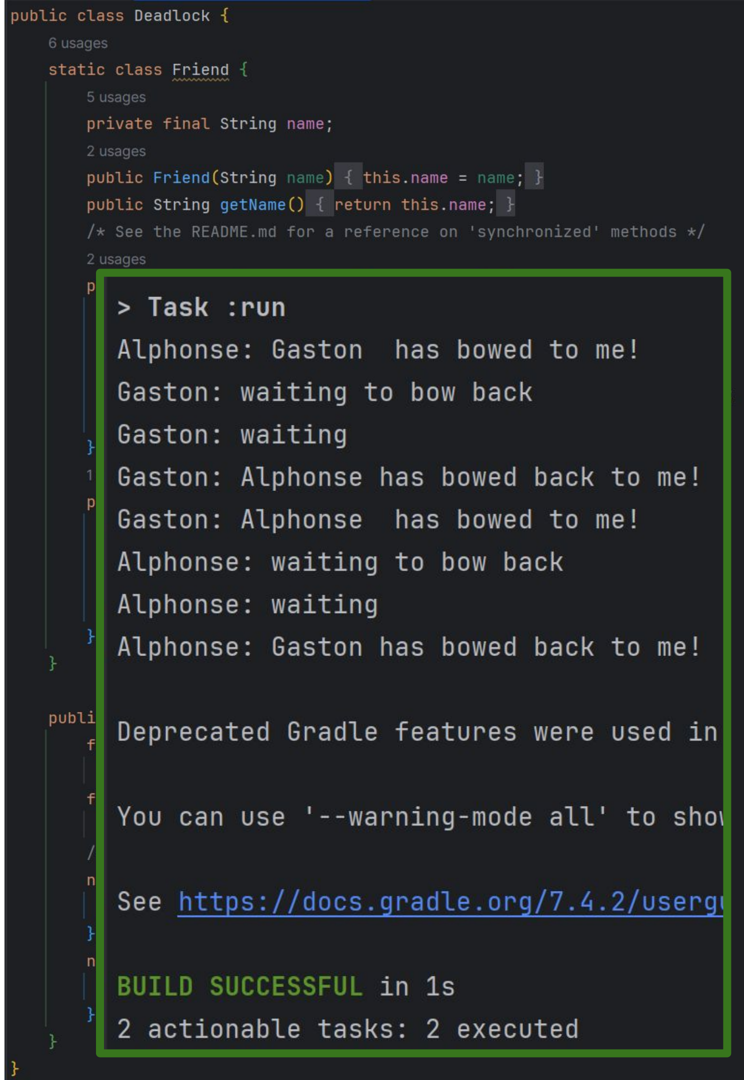
3. Synchronize the bowBack call with a synchronized statement

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

4. 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
        p
        }
        1
        p
        }
    }

    publi
        f
        f
        /
        n
        }
        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!

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
```

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?

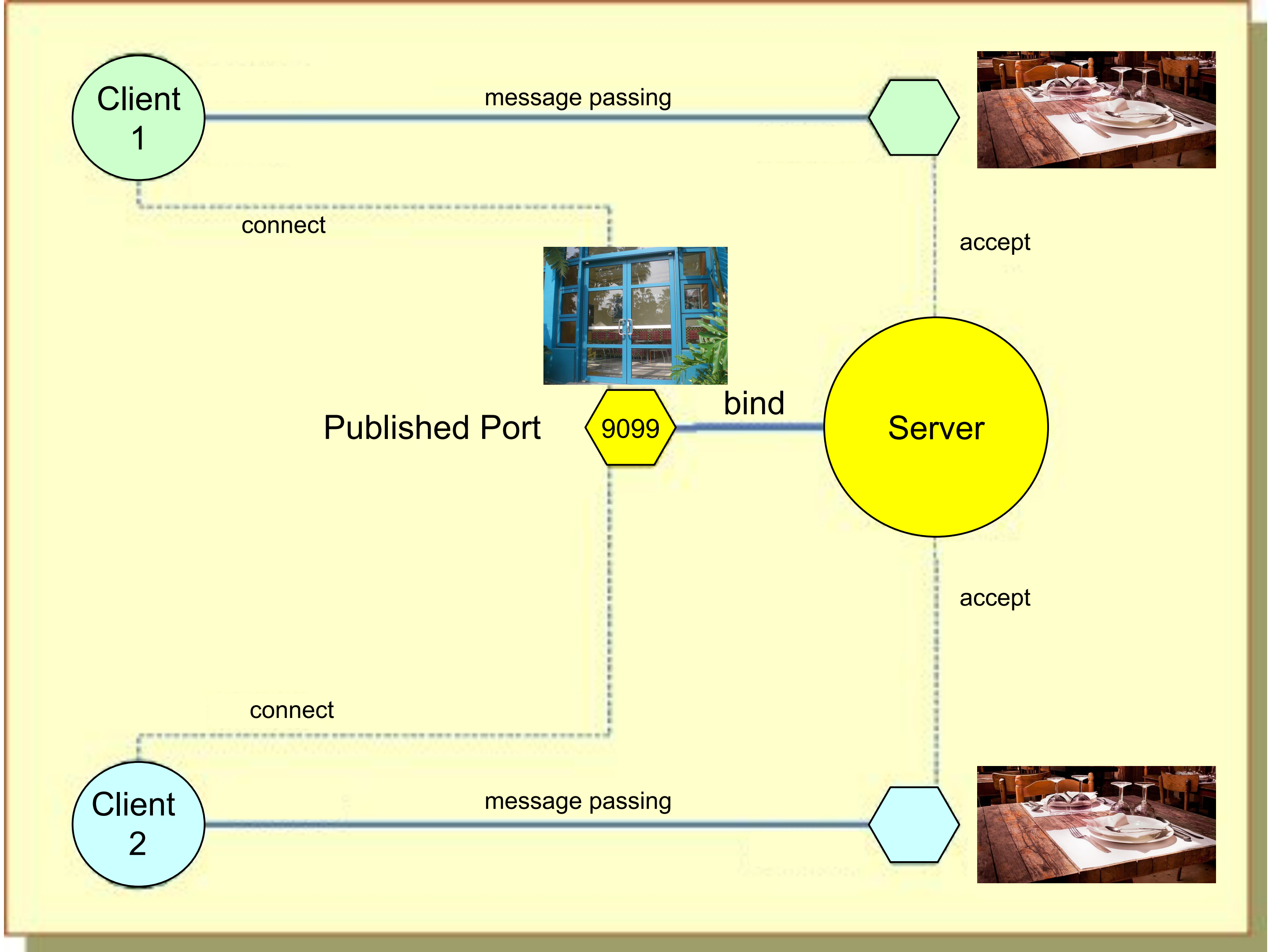
1. Define Params
2. Create Socket
3-5. Mark Socket to Listen
6. Wait for Connection
7. Handle Client Connection
8. Close Client Connection
9. Continue Listening

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

7. Send Client Socket to thread

SER 321

Sockets!

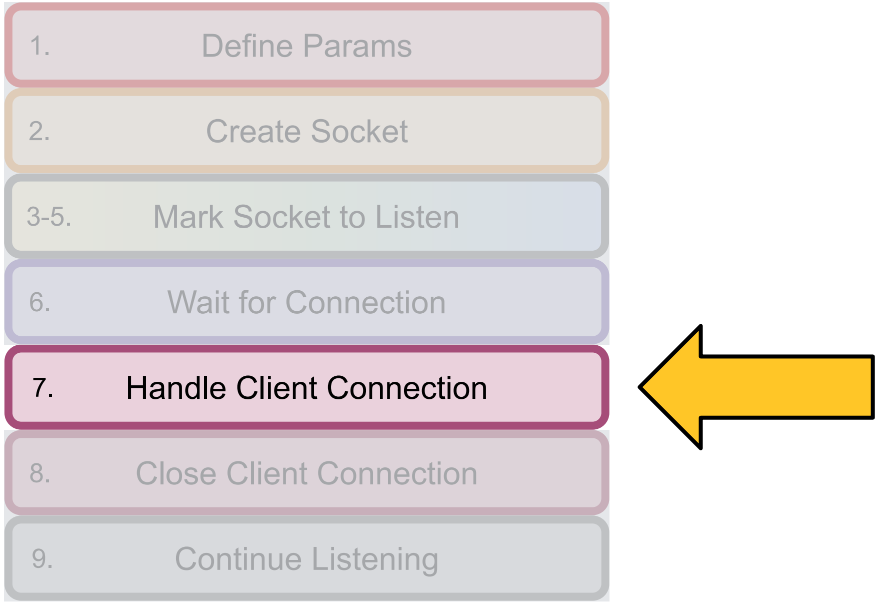


Design of an RFID Vehicle Authentication System: A Case Study for Al-Nahrain University Campus - Scientific Figure on ResearchGate. Available from: https://www.researchgate.net/figure/Client-and-Server-Socket-Ports_fig4_282671198

SER 321
Threaded Server

We send the Client Socket to the thread

Then within the thread we will…


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

1

2

3

4

5

SER 321

Scratch Space

Questions?

Survey:

http://bit.ly/ASN2324

Upcoming Events

SI Sessions:

- Monday, June 17th at 6:00 pm MST
- Thursday, June 20th at 6:00 pm MST
- Sunday, June 23rd at 6:00 pm MST

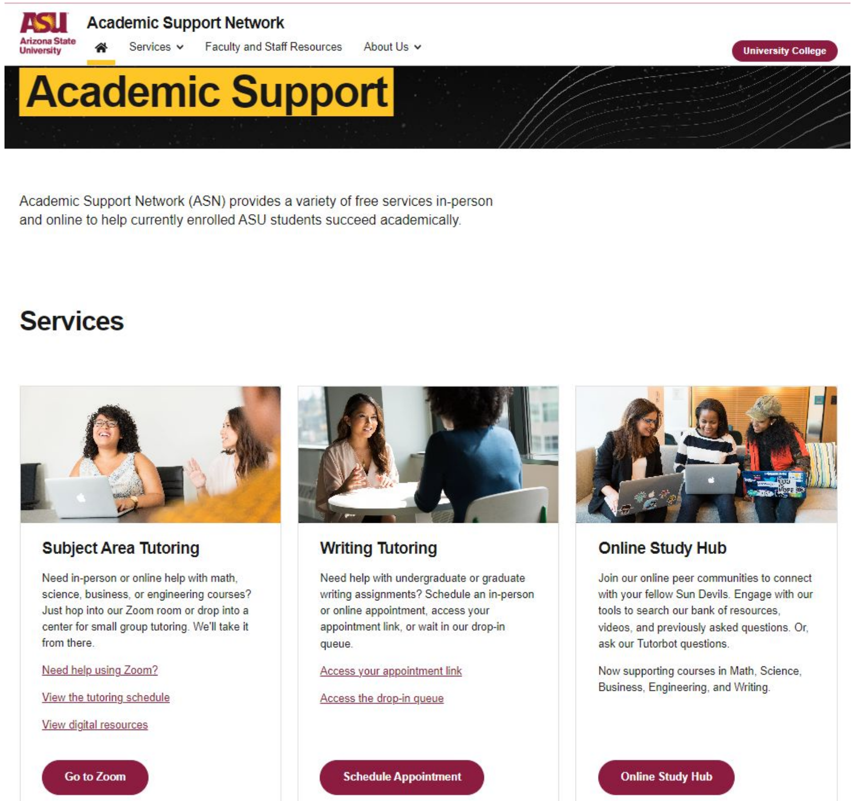
Review Sessions:

- 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 Network

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

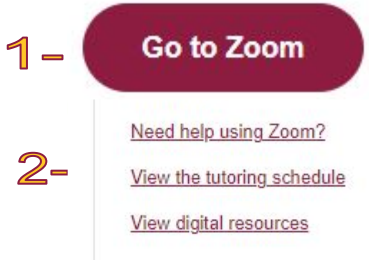
Schedule Appointment

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

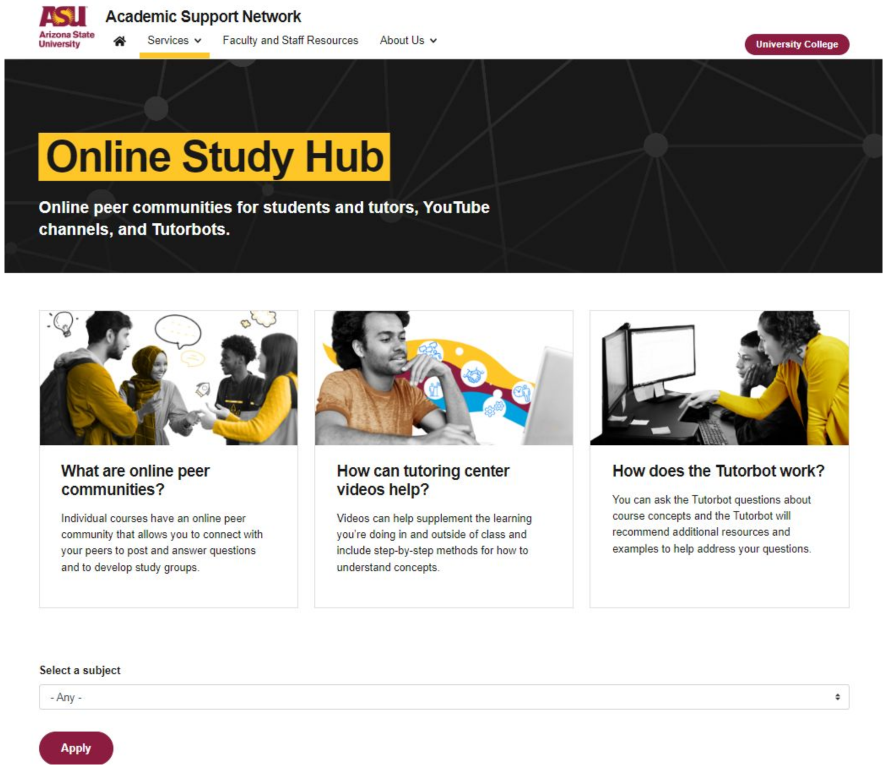


1. **Click on ‘Go to Zoom’ to log onto our Online Tutoring Center.**
2. **Click on ‘View the tutoring schedule’ to see when tutors are available for specific courses.**

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.

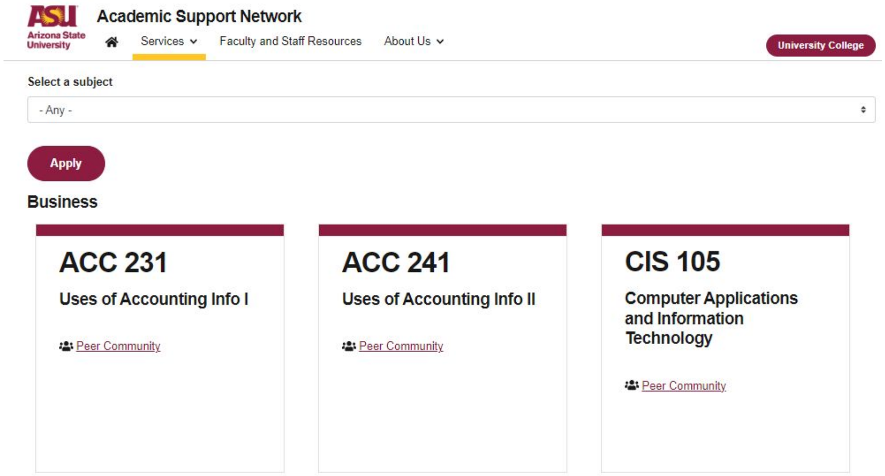
How can tutoring center videos help?

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

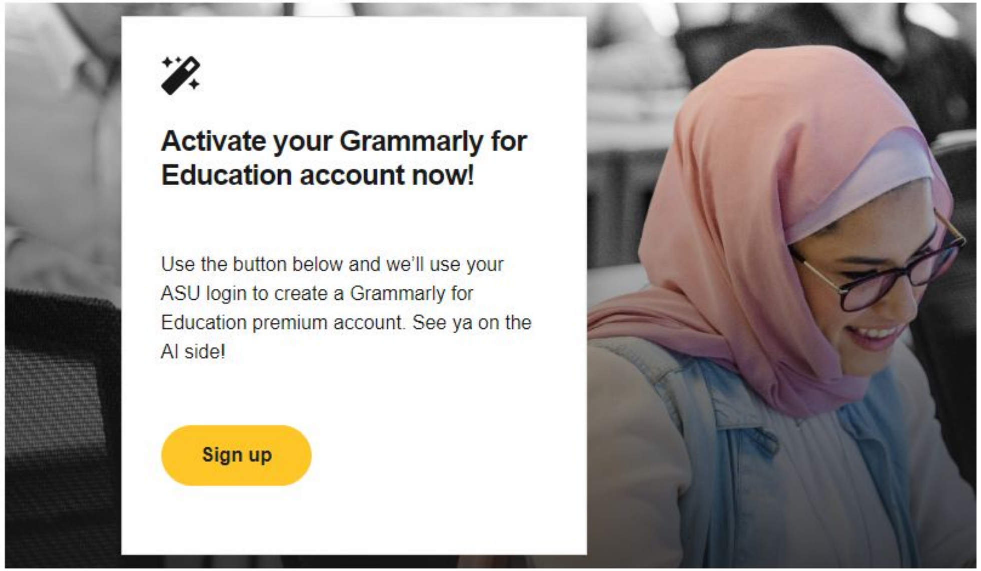
ACC 241 Uses of Accounting Info II — Peer Community

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