Java Semaphore (Part 3)

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Part of the Module

• Appreciate the concept of semaphores
• Recognize the two types of semaphores
• Know a human known use of semaphores
• Understand the structure & functionality of Java Semaphore & its methods
• Recognize how Java semaphores enable multiple threads to
• Mediate access to a limited # of shared resources
Learning Objectives in this Part of the Module

• Appreciate the concept of semaphores
• Recognize the two types of semaphores
• Know a human known use of semaphores
• Understand the structure & functionality of Java Semaphore & its methods
• Recognize how Java semaphores enable multiple threads to
  • Mediate access to a limited # of shared resources
• Coordinate the order in which operations occur
Applying a Java Semaphore to Mediate Access
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.

Each being is implemented to run in a separate thread.

See [en.wikipedia.org/wiki/Palantir](en.wikipedia.org/wiki/Palantir)
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.
- The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.

\[
\text{e.g., limit use to two palantiri on a quad-core device to ensure system responsiveness}
\]
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.

- The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.

- A permit must be acquired from a semaphore before a being can gaze.

Acquiring a permit atomically decrements the permit count.
Applying a Java Semaphore to Mediate Access

• This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.

• The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.

• A permit must be acquired from a semaphore before a being can gaze.

All available permits are now in use.
Applying a Java Semaphore to Mediate Access

• This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.

• The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.

• A permit must be acquired from a semaphore before a being can gaze.

• Other being threads must block until a permit is available.
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.
  - The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.
  - A permit must be acquired from a semaphore before a being can gaze.
  - Other being threads must block until a permit is available.
  - When a being thread is done gazing, it releases the semaphore.
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.
  - The app can be configured to restrict the number of being threads that concurrently gaze into Palantiri.
  - A permit must be acquired from a semaphore before a being can gaze.
  - Other being threads must block until a permit is available.
  - When a being thread is done gazing, it releases the semaphore.
  - Another being thread can then acquire it and proceed to gaze.
Applying a Java Semaphore to Mediate Access

- This Android app shows how a Java semaphore can be used to limit the number of Middle-Earth beings who can gaze into Palantiri concurrently.
  - The app can be configured to restrict the number of being threads that concurrently gaze into palantiri.
  - A permit must be acquired from a semaphore before a being can gaze.
  - Other being threads must block until a permit is available.
    - When a being thread is done gazing, it releases the semaphore.
    - Another being thread can then acquire it and proceed to gaze.

This example “fully brackets” the acquiring & releasing of permits, i.e., the thread that acquires a semaphore is the same as the one that releases it.
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
start()
run()
start()
start()
run()
start()
run()
r.gaze()
run()
r.gaze()
r.gaze()
r.gaze()
r = acquire()
r = acquire()
r = acquire()
r = acquire()
release(r)
release(r)
release(r)
release(r)
```
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

: Palantiri Presenter

start()
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
start()
start()
start()

run()
run()
run()
```

- : Palantiri Presenter
- : BeingRunnables

• Palantiri Presenter

- BeingRunnables
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
Applying a Java Semaphore to Mediate Access

: Palantiri Presenter -->

: BeingRunnables

mPalantiriManager : PalantiriManager

start()

start() run()
p = acquire()

start() run()
p = acquire()

start() run()
p = acquire()
```
Applying a Java Semaphore to Mediate Access

• UML sequence diagram for this app

![UML sequence diagram]

```
start()
start()
start()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
mPalantiriManager
: PalantiriManager
```

```
: PalantiriPresenter
```

```
: BeingRunnables
```

```
Applying a Java Semaphore to Mediate Access
```

```
start()
start()
start()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
mPalantiriManager
: PalantiriManager
```

```
: PalantiriPresenter
```

```
: BeingRunnables
```

```
Applying a Java Semaphore to Mediate Access
```

```
start()
start()
start()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
mPalantiriManager
: PalantiriManager
```

```
: PalantiriPresenter
```

```
: BeingRunnables
```

```
Applying a Java Semaphore to Mediate Access
```

```
start()
start()
start()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
run()
p = acquire()
```

```
mPalantiriManager
: PalantiriManager
```

```
: PalantiriPresenter
```

```
: BeingRunnables
```
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
start()
start()
start()
start()
```

```
run()
p.gaze()
p = acquire()
```

```
mPalantiriManager : PalantiriManager
 PalantiriPresenter : Palantiri
 BeingRunnables
 : Palantiri
```
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
: Palantiri
  Presenter

: BeingRunnables

p : Palantir

mPalantiriManager : PalantiriManager
```

```
start() → run()
start() → run()
p = acquire()
p.gaze()
release(p)
p = acquire()
p.gaze()
release(p)
p = acquire()
p.gaze()
release(p)
```
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```java
start()
start()
start()
run()
run()
p = acquire()
p.gaze()
p.gaze()  
run()  
p = acquire()
p.gaze()  
run()  
p = acquire()
p.gaze()  
run()  
p = acquire()
p.gaze()  
run()  
p = acquire()
p.gaze()  
release(p)
release(p)
```

```
PalantiriPresenter

: Palantiri

BeingRunnables

p : Palantir

mPalantiriManager : PalantiriManager
```
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

start()

start()

start()

start()

run()

run()

run()

run()

p = acquire()

p = acquire()

p = acquire()

release(p)

release(p)

release(p)

release(p)
Applying a Java Semaphore to Mediate Access

- UML sequence diagram for this app

```
start()
start()
start()
start()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
run()
p = acquire()
p.gaze()
```
Applying Java Semaphores to Coordinate Threads
The Android ping-pong app coordinates thread interactions via various Java synchronizers, including Java semaphores. i.e., these two threads alternate printing “ping” & “pong” on the display.

```
// ping thread
PingPongThread ping = new PingPongThread();
ping.run();
ping.print("ping");

// pong thread
PingPongThread pong = new PingPongThread();
pong.run();
pong.print("pong");
```

See [github.com/douglascraigschmidt/POSA/tree/master/ex/M3/PingPong](https://github.com/douglascraigschmidt/POSA/tree/master/ex/M3/PingPong)
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

This app can be configured to use a pair of semaphores that coordinate the order in which the “ping” & “pong” threads are called to play ping-pong.
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

The PlayPingPongThread object starts two threads, ping & pong, that alternate printing "Ping" and "Pong", respectively, on the display.
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

The PingPongThread class implements the core ping-pong algorithm, but defers synchronization aspects to subclasses via the Template Method pattern.
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

The pingSem & PongSem semaphores coordinate the order in which the “ping” & “pong” threads are called to play ping-pong.

```java
Semaphore pingSem = new Semaphore(1);
Semaphore pongSem = new Semaphore(0);
```
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

This example does *not* “fully bracket” acquiring & releasing permits, i.e., the thread acquiring a semaphore is different from the thread releasing it!
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

```java
private final Semaphore mMine;
private final Semaphore mOther;
...

protected void acquire() { mMine.acquire(); }

protected void release() { mOther.release(); }
```

This example does not "fully bracket" acquiring & releasing permits, i.e., the thread acquiring a semaphore is different from the thread releasing it!
Applying Java Semaphores to Coordinate Threads

- UML sequence diagram for the ping-pong app

PlayPingPongThread joins with the ping & pong threads once they finish
End of Java Semaphores (Part 3)