Coordinating Threads via Java Semaphore

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

• Understand the concept of semaphores
• Be aware of the two types of semaphores
• Note a human known use of semaphores
• Recognize the structure & functionality of Java Semaphore
• Know the key methods defined by the Java Semaphore class
• Learn how Java semaphores enable multiple threads to
  • Mediate access to a limited # of shared resources
• Coordinate the order in which operations occur
Applying Java Semaphores to Coordinate Threads
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

See 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

The PlayPingPongThread object starts two threads, ping & pong, that alternate printing "Ping" & "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

```java
Semaphore pingSem = new Semaphore(1);
Semaphore pongSem = new Semaphore(0);
```

The pingSem & PongSem semaphores 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

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(); }
```

Block until semaphore is acquired

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 Coordinating Threads via Java Semaphore