

Java Barrier Synchronizers: Usage Considerations



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 Lesson

- Appreciate Java barrier synchronizer usage considerations



Java Barrier Usage Considerations

Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes



See stackoverflow.com/questions/6830904/java-tutorials-explanations-of-jsr166y-phaser/6831171#6831171

Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes
 - CountdownLatch focuses on actions



Java Barrier Usage Considerations

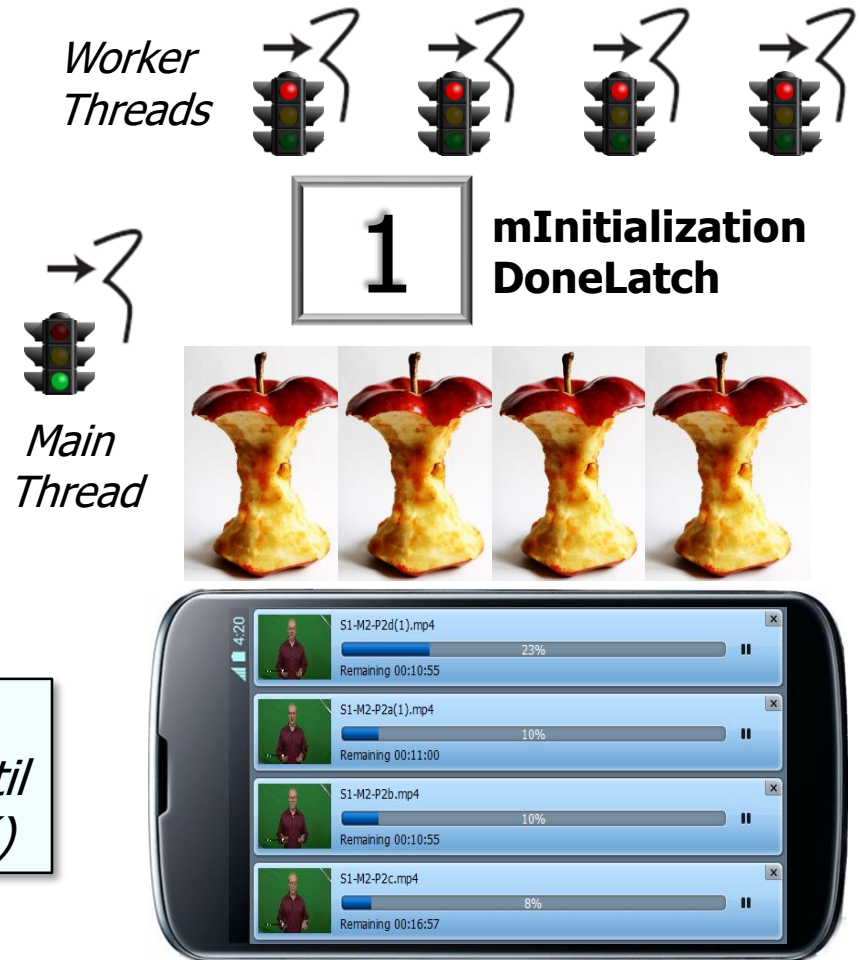
- Java's barrier synchronizers can be used for several purposes
 - CountdownLatch focuses on actions
 - It can be used as an on/off latch for an entry barrier



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
 - It can be used as an on/off latch for an entry barrier



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes
 - CountdownLatch focuses on actions
 - It can be used as an on/off latch for an entry barrier
 - It can also be used for more sophisticated exit barrier use cases



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions

- It can be used as an on/off latch for an entry barrier

- It can also be used for more sophisticated exit barrier use cases, e.g.

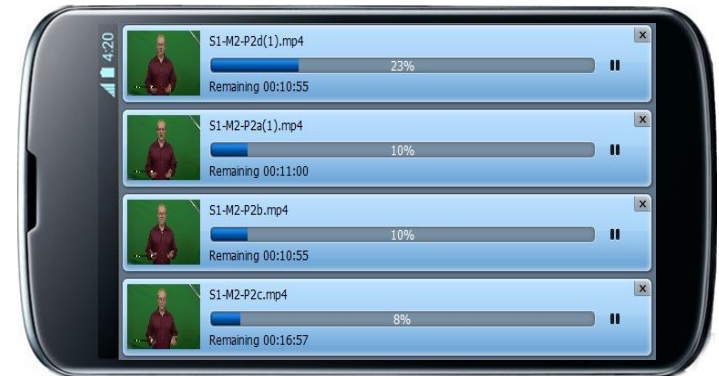
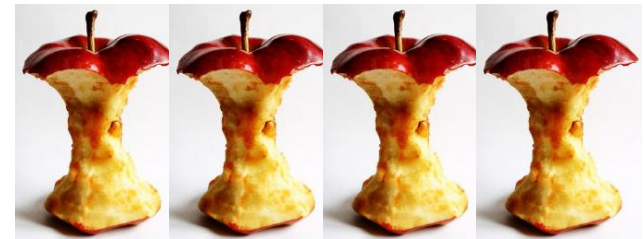
- 1 thread waits until N threads have completed an action



Main Thread

4

mConversion
DoneLatch



e.g., the main thread waits until the worker threads are finished rendering the video

Java Barrier Usage Considerations

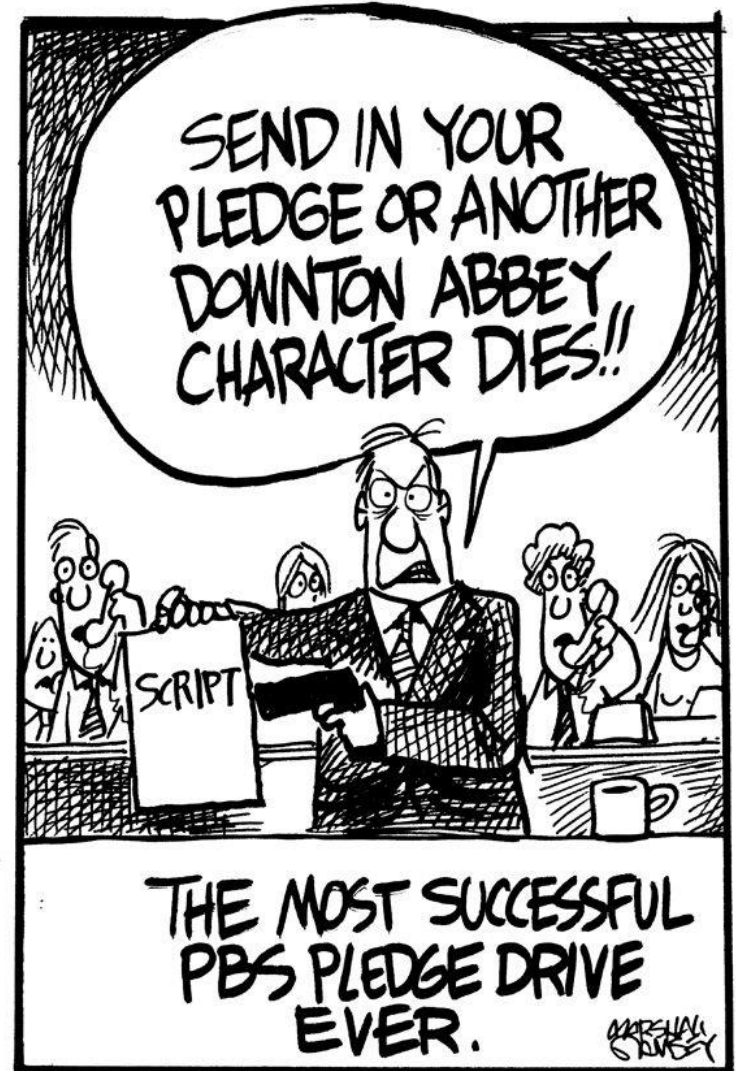
- Java's barrier synchronizers can be used for several purposes

- CountdownLatch focuses on actions

- It can be used as an on/off latch for an entry barrier

- It can also be used for more sophisticated exit barrier use cases, e.g.

- 1 thread waits until N threads have completed an action
- 1 thread waits until an action has completed N times, irrespective of which thread(s) were responsible



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes
 - CountdownLatch focuses on actions
 - It can be used as an on/off latch for an entry barrier
 - It can also be used for more sophisticated exit barrier use cases
 - Most appropriate/optimized for relatively simple use cases



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes
 - `CountDownLatch` focuses on actions
 - `CyclicBarrier` focuses on threads



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes
 - `CountDownLatch` focuses on actions
 - `CyclicBarrier` focuses on threads
 - It enables a set of threads to all wait for each other to reach a common barrier point



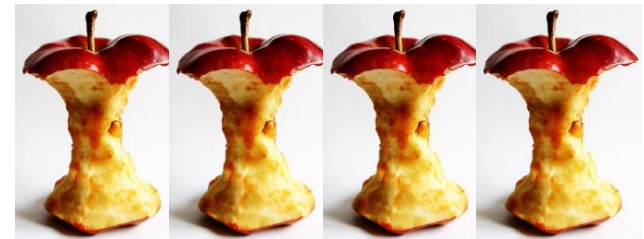
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

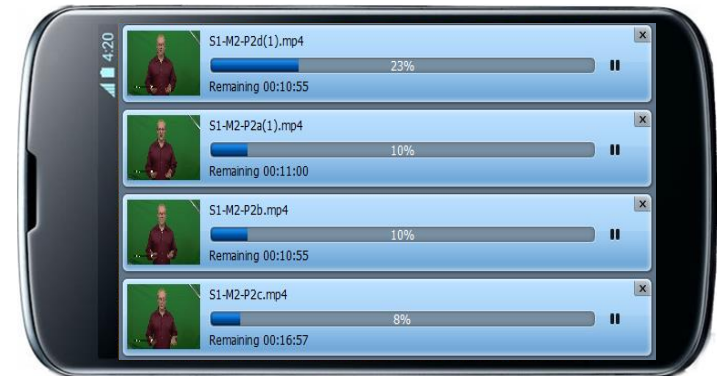
- `CountDownLatch` focuses on actions
- `CyclicBarrier` focuses on threads
 - It enables a set of threads to all wait for each other to reach a common barrier point



**mCyclic
Barrier**



e.g., a barrier can be used to wait for one or more algorithm iterations to finish before deciding to move on to the next cycle



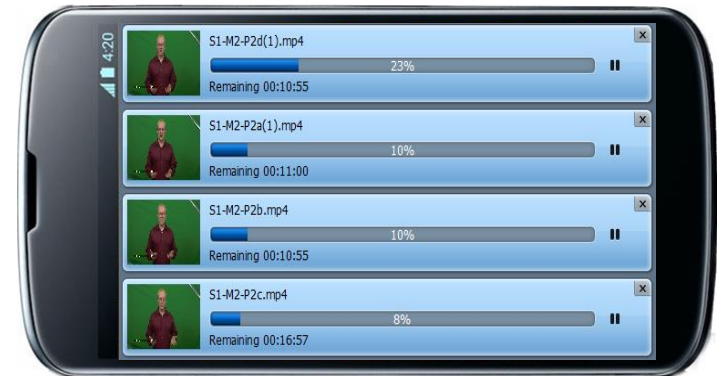
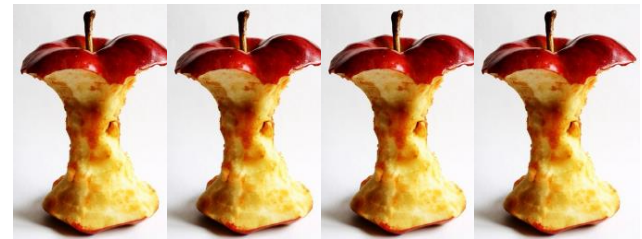
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
 - It enables a set of threads to all wait for each other to reach a common barrier point
 - It requires a fixed # of threads



**mCyclic
Barrier**



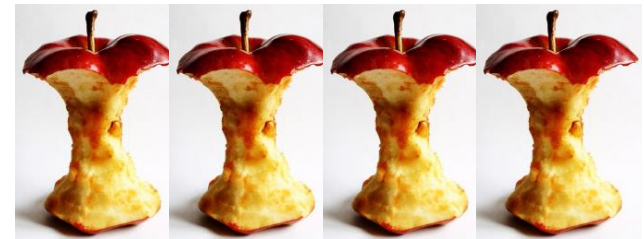
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

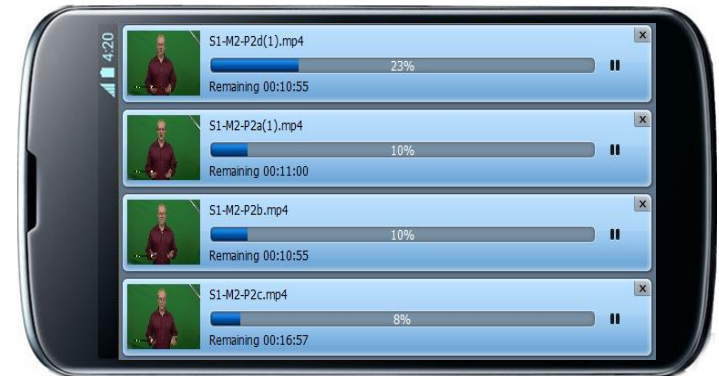
- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
 - It enables a set of threads to all wait for each other to reach a common barrier point
- It requires a fixed # of threads
 - This may be overly limited



**mCyclic
Barrier**



LIMITED



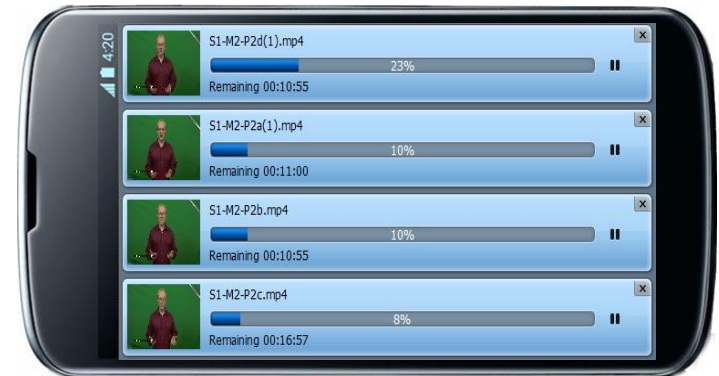
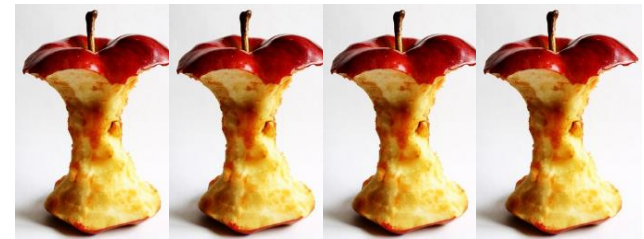
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
- Phaser focuses on a variable (or fixed) # of threads
 - It enables threads to wait for each other to complete processing in cycles



**mPhaser
Barrier**



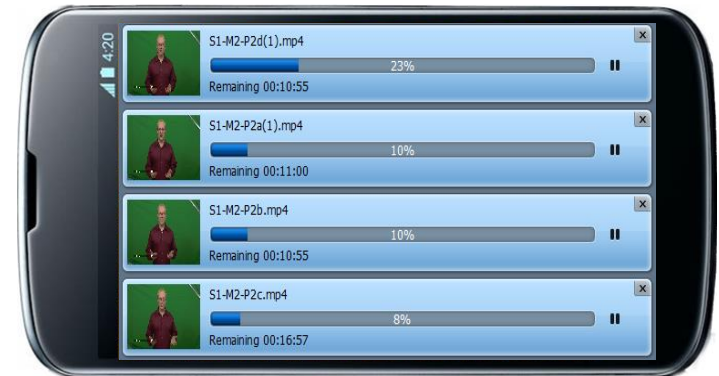
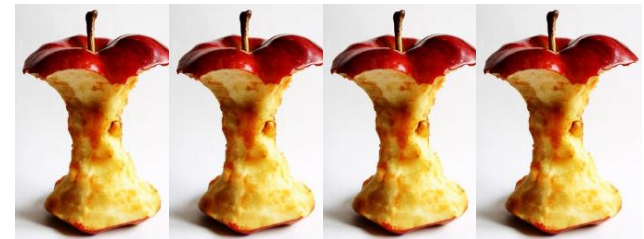
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
- Phaser focuses on a variable (or fixed) # of threads
 - It enables threads to wait for each other to complete processing in cycles



**mPhaser
Barrier**



Using Phasers for a fixed # of threads is typically overkill!

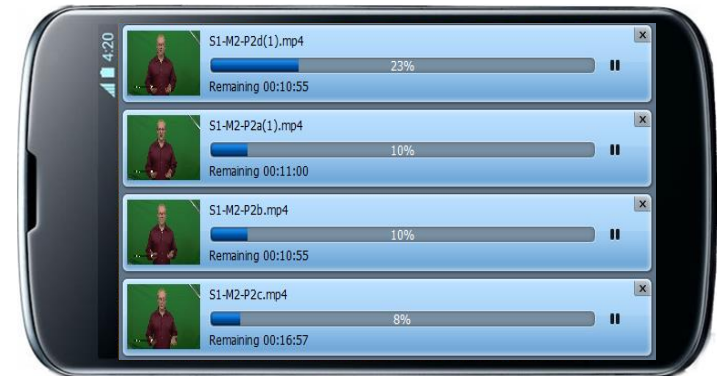
Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
- Phaser focuses on a variable (or fixed) # of threads
 - It enables threads to wait for each other to complete processing in cycles
 - It's more flexible than the two other types of Java barrier synchronizers



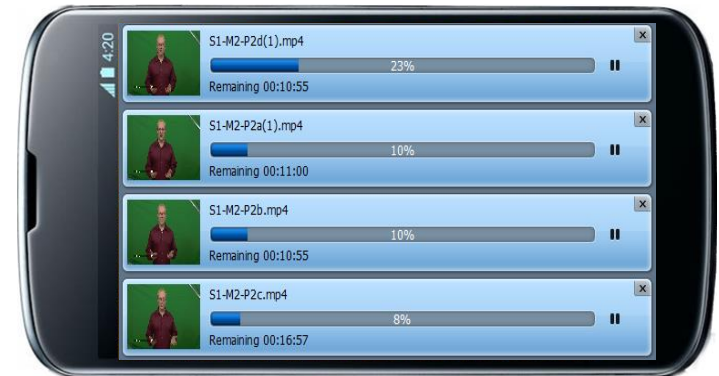
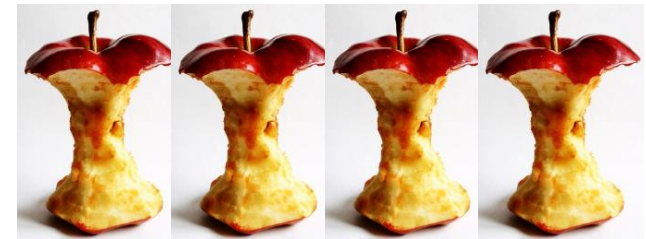
**mPhaser
Barrier**



Java Barrier Usage Considerations

- Java's barrier synchronizers can be used for several purposes

- CountDownLatch focuses on actions
- CyclicBarrier focuses on threads
- Phaser focuses on a variable (or fixed) # of threads
 - It enables threads to wait for each other to complete processing in cycles
 - It's more flexible than the two other types of Java barrier synchronizers
 - However, they are also more complex to program



COMPLEX

End of Java Barrier Synchronizers: Usage Considerations