Barrier Synchronization: Overview of Java Barrier Synchronizers

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

• Understand what barrier synchronization is & know three different ways of using barrier synchronizers
• Note a human known use of barrier synchronization
• Recognize the three types of Java barrier synchronizers
Learning Objectives in this Lesson

- Understand what barrier synchronization is & know three different ways of using barrier synchronizers
- Note a human known use of barrier synchronization
- Recognize the three types of Java barrier synchronizers
- Know how to categorize various type of Java barrier synchronizers

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDownLatch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>
Types of Java Barrier Synchronizers
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - **CountDownLatch**
    - Allows one or more threads to wait on the completion of operations in other threads

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/CountDownLatch.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/CountDownLatch.html)

**e.g., a race can’t begin until all horses are at the starting gate**
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - **CountDownLatch**
    - Allows one or more threads to wait on the completion of operations in other threads
    - Supports entry & exit barriers, but not cyclic barriers

```java
public class CountDownLatch {
  public CountDownLatch(int count) {
  }
  public void await() {
  }
  public void await(long timeout, TimeUnit unit) {
  }
  public boolean countDown() {
  }
  public void getCount() {
  }
  public String toString() {
  }
}
```
• Java supports 3 types of barrier synchronizers
  • **CountDownLatch**
    • Allows one or more threads to wait on the completion of operations in other threads
    • Supports entry & exit barriers, but not cyclic barriers
  • The CountDownLatch API is very simple
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - CountDownLatch
  - **CyclicBarrier**
    - Allows a set of threads to all wait for each other to reach a common barrier point

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/CyclicBarrier.html

**e.g., a team begins their work when the next car arrives on the assembly line**
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - **CountDownLatch**
  - **CyclicBarrier**
    - Allows a set of threads to all wait for each other to reach a common barrier point
    - Supports entry, exit, & cyclic barriers for a fixed # of threads

```
<<Java Class>>

CyclicBarrier

- CyclicBarrier(int,Runnable)
- CyclicBarrier(int)
- getParties():int
- await():int
- await(long, TimeUnit):int
- isBroken():boolean
- reset():void
- getNumberOfWaiting():int
```
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - **CountDownLatch**
  - **CyclicBarrier**
    - Allows a set of threads to all wait for each other to reach a common barrier point
    - Supports entry, exit, & cyclic barriers for a fixed # of threads
  - The CyclicBarrier API is also very simple
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - **CountDownLatch**
  - **CyclicBarrier**
  - **Phaser**
    - A more flexible, reusable, & dynamic barrier synchronizer that subsumes CyclicBarrier & CountDownLatch

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Phaser.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/Phaser.html)

e.g., crews begin their work when all the team members arrive
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - CountDownLatch
  - CyclicBarrier
  - Phaser
    - A more flexible, reusable, & dynamic barrier synchronizer that subsumes CyclicBarrier & CountDownLatch
    - Supports entry, exit, & cyclic barriers for a variable # of threads
Types of Java Barrier Synchronizers

- Java supports 3 types of barrier synchronizers
  - CountDownLatch
  - CyclicBarrier
  - Phaser

  - A more flexible, reusable, & dynamic barrier synchronizer that subsumes CyclicBarrier & CountDownLatch
  - Supports entry, exit, & cyclic barriers for a variable # of threads
  - The Phaser API is more complex..

```
<<Java Class>>

Phaser

- Phaser()
- Phaser(int)
- Phaser(Phaser)
- Phaser(Phaser,int)

- register():int
- bulkRegister(int):int
- arrive():int
- arriveAndDeregister():int
- arriveAndAwaitAdvance():int
- awaitAdvance(int):int
- awaitAdvanceInterruptibly(int):int
- awaitAdvanceInterruptibly(int,long,TimeUnit):int
- forceTermination():void
- getPhase():int
- getRegisteredParties():int
- getArrivedParties():int
- getUnarrivedParties():int
- getParent():Phaser
- getRoot():Phaser
- isTerminated():boolean
- onAdvance(int,int):boolean
- toString()
```

14
Categorizing Java Barrier Synchronizers
Java’s barrier synchronizers can be categorized in several ways:

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDown Latch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>
Java’s barrier synchronizers can be categorized in several ways:

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDownLatch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>
**Categorizing Java Barrier Synchronizers**

- Java’s barrier synchronizers can be categorized in several ways.

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Sized</td>
<td>CountDownLatch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>
Categorizing Java Barrier Synchronizers

- Java’s barrier synchronizers can be categorized in several ways

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDownLatch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>

A CountDownLatch can be used w/a variable # of parties, but it’s uncommon.
Java’s barrier synchronizers can be categorized in several ways:

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDownLatch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>
Categorizing Java Barrier Synchronizers

- Java’s barrier synchronizers can be categorized in several ways

<table>
<thead>
<tr>
<th># of Iterations</th>
<th>Fixed # of Parties</th>
<th>Variable # of Parties</th>
</tr>
</thead>
<tbody>
<tr>
<td>One-Shot</td>
<td>CountDown Latch</td>
<td>Phaser</td>
</tr>
<tr>
<td>Cyclic</td>
<td>CyclicBarrier</td>
<td>Phaser</td>
</tr>
</tbody>
</table>

*These categories are not mutually exclusive, i.e., Phaser appears multiple times*
End of Barrier Synchronization: Overview of Java Barrier Synchronizers