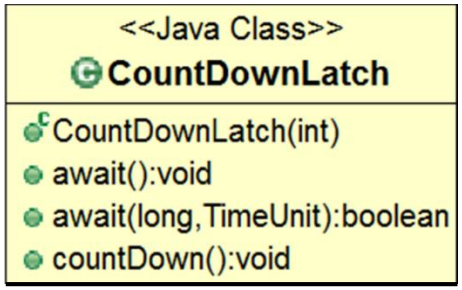
Barrier Synchronization: Overview of Java Barrier Synchronizers

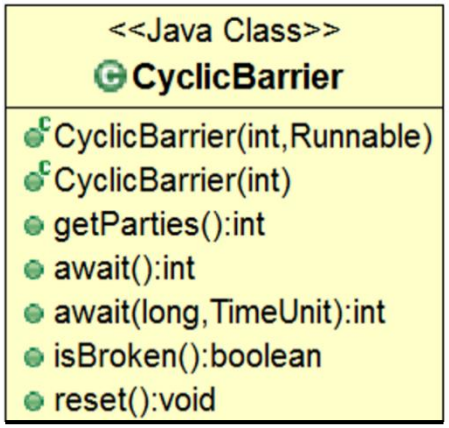
Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

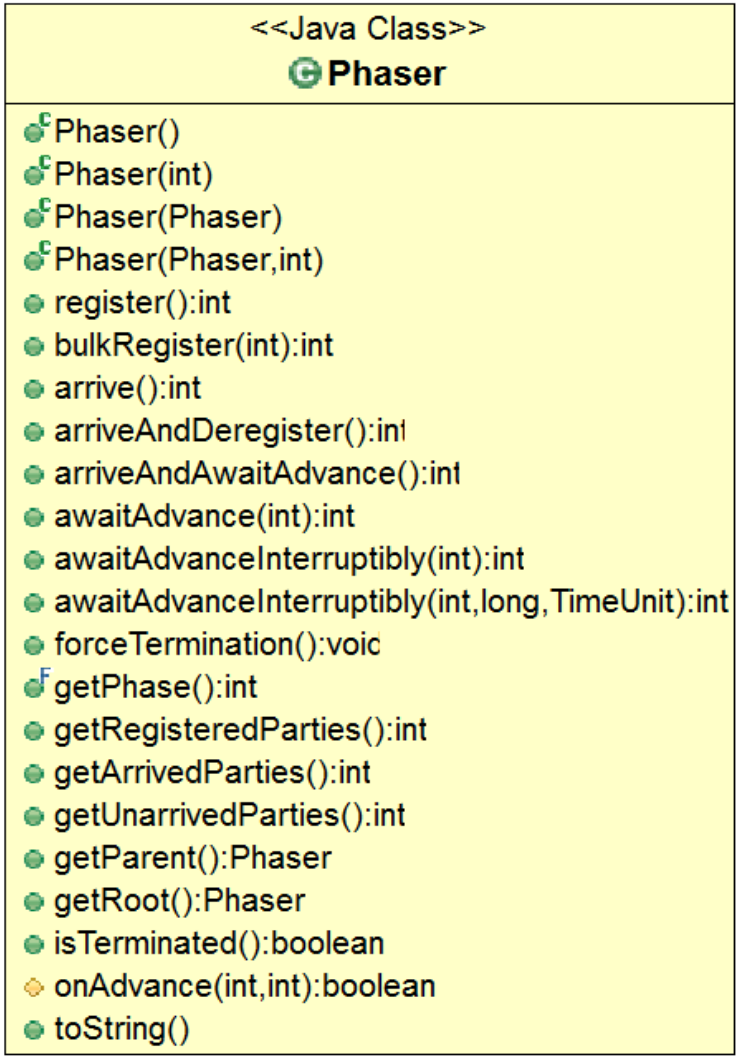
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

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

<<Java Class>>

CountDownLatch

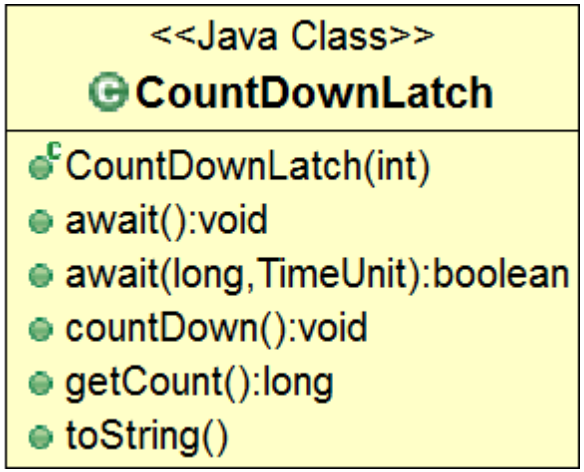
- CountDownLatch(int)
- await():void
- await(long,TimeUnit):boolean
- countDown():void
- getCount():long
- toString()

e.g., a race can't begin until all horses are at the starting gate

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

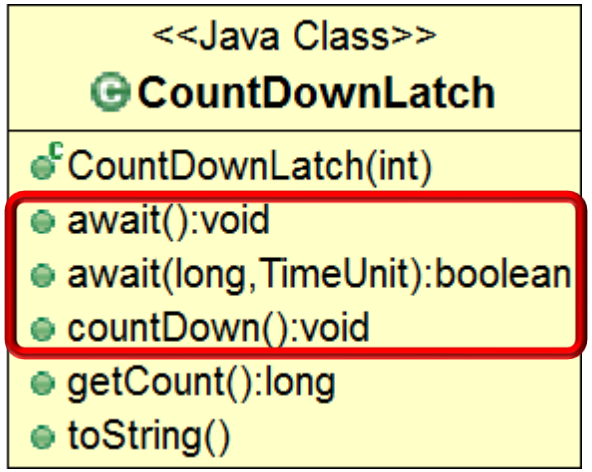
Types of Java Barrier Synchronizers

- 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



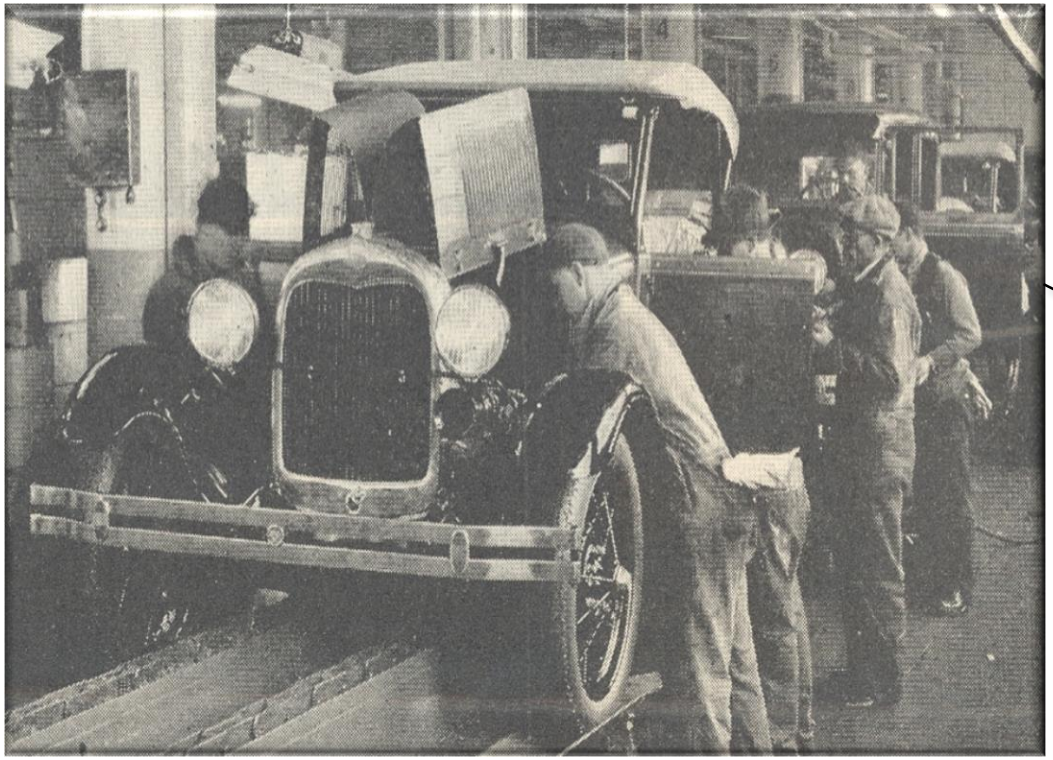
Types of Java Barrier Synchronizers

- 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
    - 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



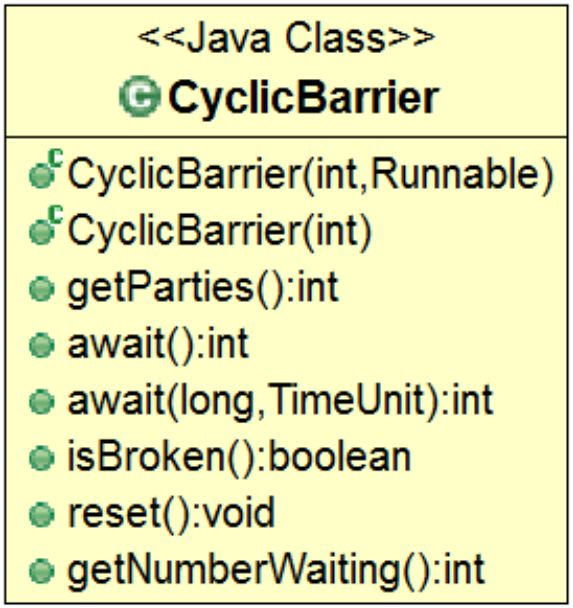
e.g., a team begins their work when the next car arrives on the assembly line

```
<<Java Class>>
CyclicBarrier
CyclicBarrier(int,Runnable)
CyclicBarrier(int)
getParties():int
await():int
await(long,TimeUnit):int
isBroken():boolean
reset():void
getNumberWaiting():int
```

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

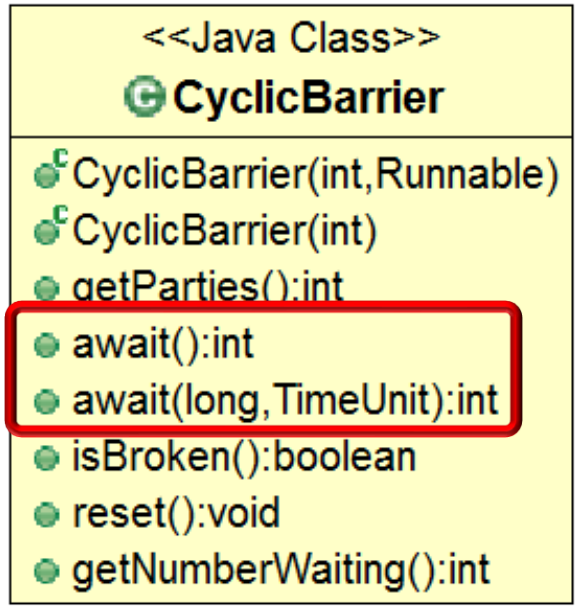
Types of Java Barrier Synchronizers

- 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



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

```
<<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():voic
getPhase():int
getRegisteredParties():int
getArrivedParties():int
getUnarrivedParties():int
getParent():Phaser
getRoot():Phaser
isTerminated():boolean
onAdvance(int,int):boolean
```

e.g., crews begin their work when all the team members arrive

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

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

<<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():voic
- getPhase():int
- getRegisteredParties():int
- getArrivedParties():int
- getUnarrivedParties():int
- getParent():Phaser
- getRoot():Phaser
- isTerminated():boolean
- onAdvance(int,int):boolean
- toString()

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()

Categorizing Java Barrier Synchronizers

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

A CountDownLatch can be used w/a variable # of parties, but it's uncommon

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

Categorizing Java Barrier Synchronizers

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

# of Iterations	Fixed # of Parties	Variable # of Parties
One-Shot	CountDown Latch	Phaser
Cyclic	CyclicBarrier	Phaser

These categories are not mutually exclusive, i.e., Phaser appears multiple times

End of Barrier Synchronization: Overview of Java Barrier Synchronizers