Types of Java Synchronizer Capabilities (Part 2)

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 Lesson

- Be aware of the Java memory model
- Understand the purpose of Java synchronizers
- Recognize the pervasiveness of Java synchronizers
- Know the types of capabilities provided by Java synchronizers

Category	Definition
Atomic operations	An action that effectively happens all at once or not at all
Mutual exclusion	Allows concurrent access & updates to shared mutable data without race conditions
Coordination	Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Barrier synchronization	Ensures that any thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier

Learning Objectives in this Part of the Lesson

- Be aware of the Java memory model
- Understand the purpose of Java synchronizers
- Recognize the pervasiveness of Java synchronizers
- Know the types of capabilities provided by Java synchronizers

Category	Definition
Atomic operations	An action that effectively happens all at once or not at all
Mutual exclusion	Allows concurrent access & updates to shared mutable data without race conditions
Coordination	Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Barrier synchronization	Ensures that any thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier

Learning Objectives in this Part of the Lesson

- Be aware of the Java memory model
- Understand the purpose of Java synchronizers
- Recognize the pervasiveness of Java synchronizers
- Know the types of capabilities provided by Java synchronizers

Category	Definition
Atomic operations	An action that effectively happens all at once or not at all
Mutual exclusion	Allows concurrent access & updates to shared mutable data without race conditions
Coordination	Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Barrier synchronization	Ensures that any thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier

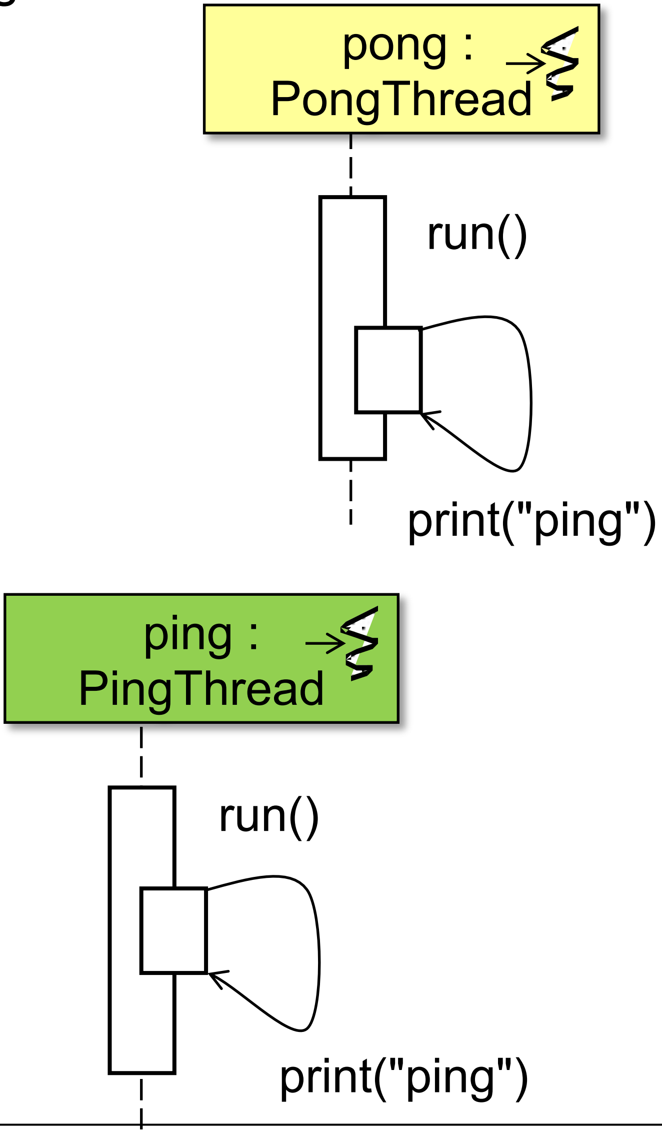
Types of Java Synchronizer Capabilities

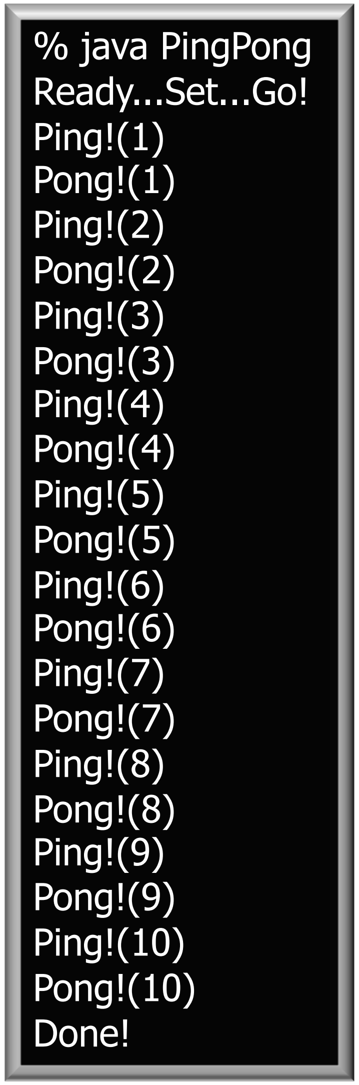
Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - **Atomic ordering**
  - **Mutual exclusion**
  - **Coordination**
    - Ensures computations run properly

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - Atomic ordering
  - Mutual exclusion
  - **Coordination**
    - Ensures computations run properly, e.g.
      - In the right order



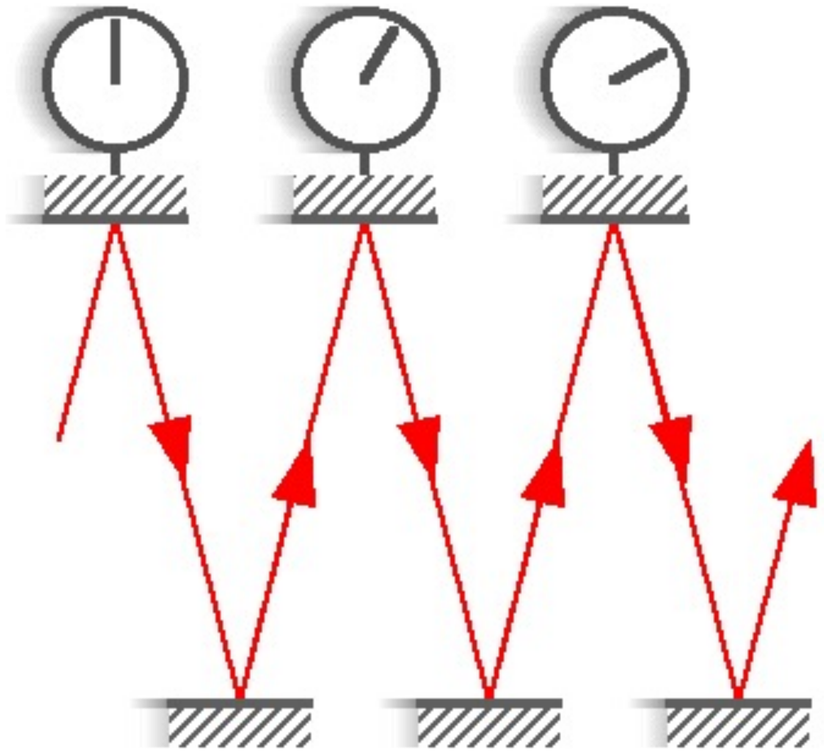


```
% java PingPong
Ready...Set...Go!
Ping!(1)
Pong!(1)
Ping!(2)
Pong!(2)
Ping!(3)
Pong!(3)
Ping!(4)
Pong!(4)
Ping!(5)
Pong!(5)
Ping!(6)
Pong!(6)
Ping!(7)
Pong!(7)
Ping!(8)
Pong!(8)
Ping!(9)
Pong!(9)
Ping!(10)
Pong!(10)
Done!
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/PingPongApplication

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - **Atomic ordering**
  - **Mutual exclusion**
  - **Coordination**
    - Ensures computations run properly, e.g.
      - In the right order
      - At the right time



See en.wikipedia.org/wiki/Real-time_computing

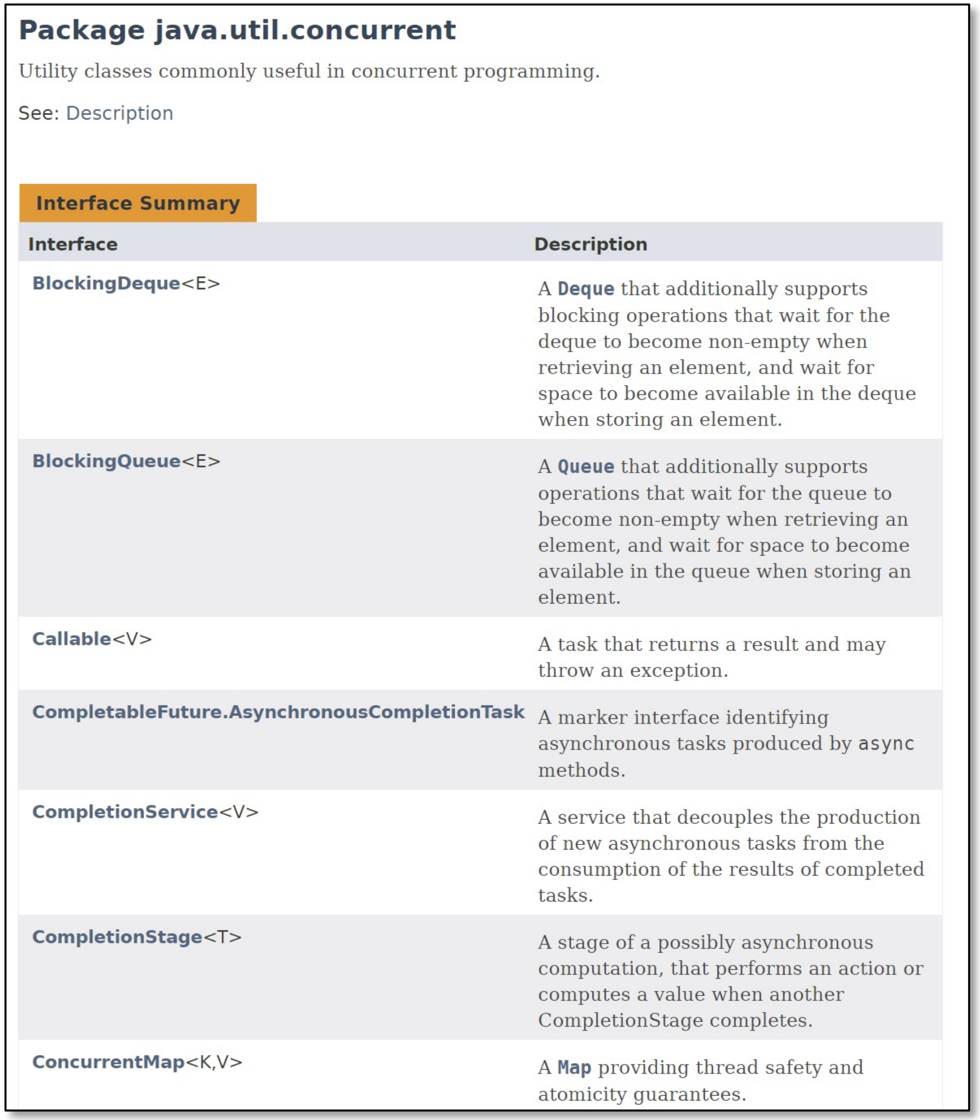
Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - **Atomic ordering**
  - **Mutual exclusion**
  - **Coordination**
    - Ensures computations run properly, e.g.
      - In the right order
      - At the right time
      - Under the right conditions

See github.com/douglascraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - Atomic ordering
  - Mutual exclusion
  - **Coordination**
    - Ensures computations run properly
    - Coordination is supported by the Java concurrent & locks packages
      - e.g., ConditionObject, Semaphore, etc.



Package java.util.concurrent

Utility classes commonly useful in concurrent programming.

See: Description

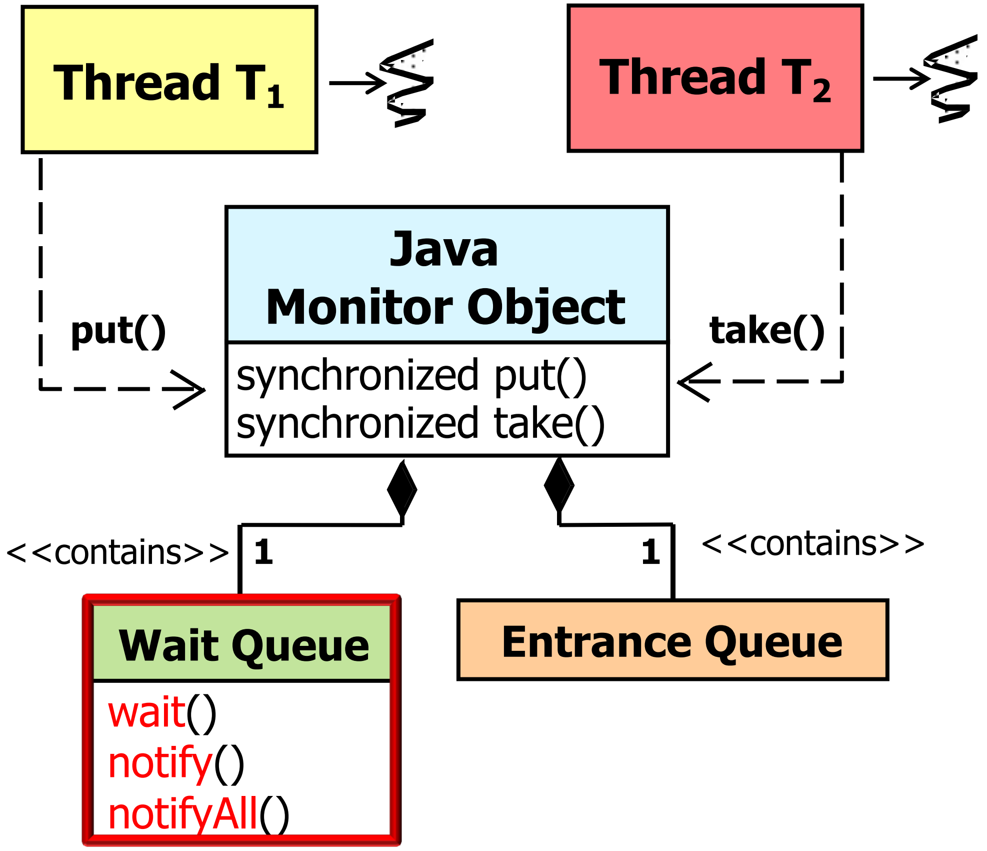
Interface Summary

Interface	Description
BlockingDeque<E>	A **Deque** that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.
BlockingQueue<E>	A **Queue** that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
Callable<V>	A task that returns a result and may throw an exception.
CompletableFuture.AsynchronousCompletionTask	A marker interface identifying asynchronous tasks produced by async methods.
CompletionService<V>	A service that decouples the production of new asynchronous tasks from the consumption of the results of completed tasks.
CompletionStage<T>	A stage of a possibly asynchronous computation, that performs an action or computes a value when another CompletionStage completes.
ConcurrentMap<K,V>	A **Map** providing thread safety and atomicity guarantees.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - **Atomic ordering**
  - **Mutual exclusion**
  - **Coordination**
    - Ensures computations run properly
    - Coordination is supported by the Java concurrent & locks packages
    - Coordination is also supported by Java built-in monitor objects



See www.artima.com/insidejvm/ed2/threadsynch.html

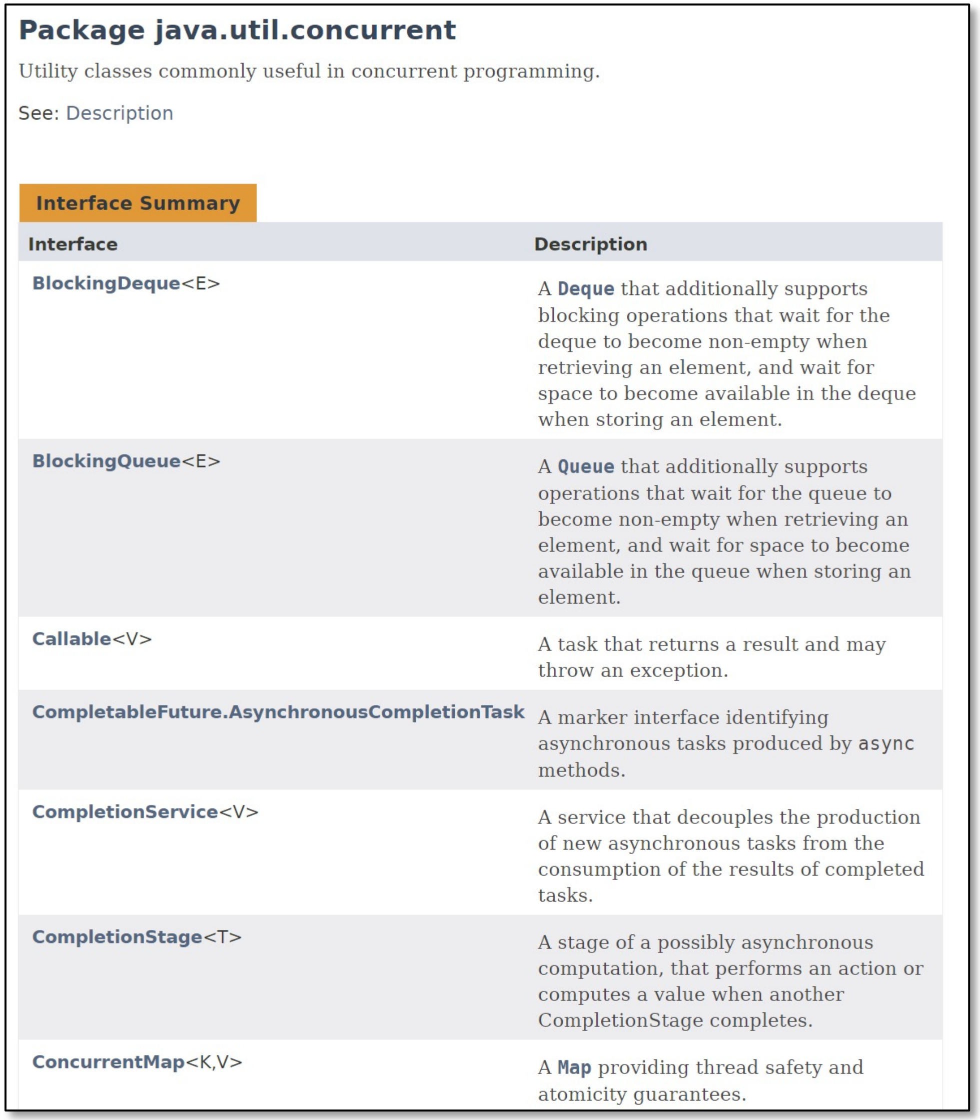
Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - Atomic ordering
  - Mutual exclusion
  - Coordination
  - **Barrier synchronization**
    - Ensures that any thread(s) must stop at a certain point & cannot proceed until all thread(s) reach the barrier

Barrier synchronization is a variant of coordination

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - Atomic ordering
  - Mutual exclusion
  - Coordination
  - **Barrier synchronization**
    - Ensures that any thread(s) must stop at a certain point & cannot proceed until all thread(s) reach the barrier
    - Barrier synchronization is supported by the Java concurrent package
      - e.g., CountDownLatch, CyclicBarrier, Phaser, etc.



Package java.util.concurrent

Utility classes commonly useful in concurrent programming.

See: Description

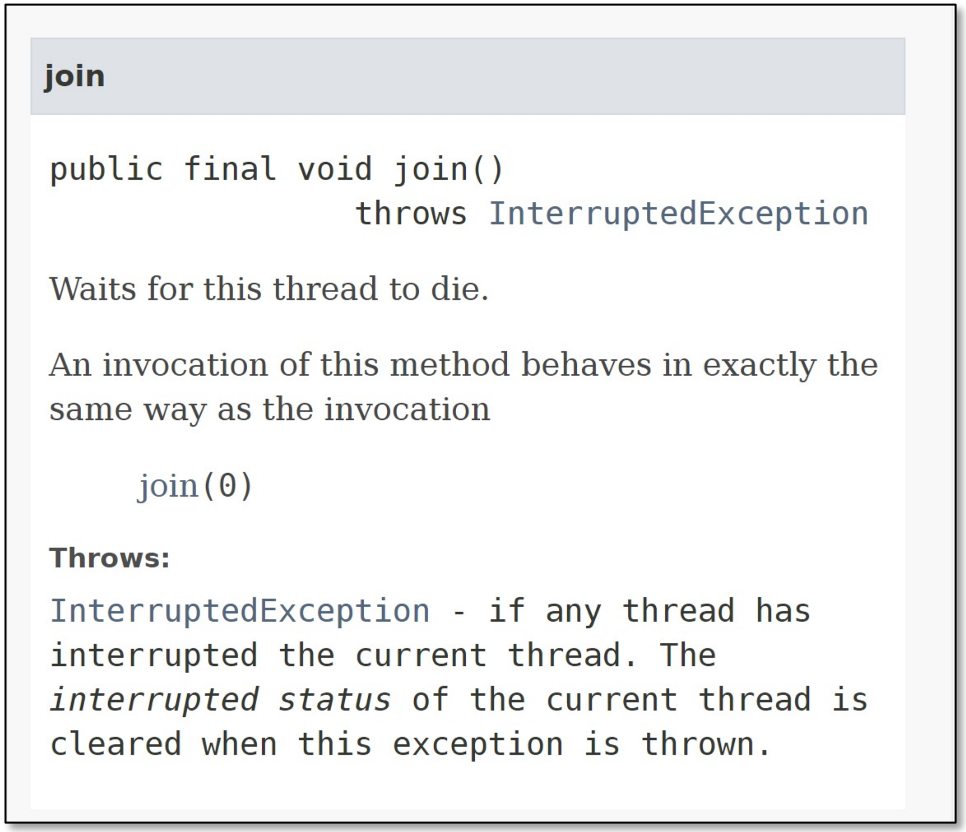
Interface Summary

Interface	Description
BlockingDeque<E>	A Deque that additionally supports blocking operations that wait for the deque to become non-empty when retrieving an element, and wait for space to become available in the deque when storing an element.
BlockingQueue<E>	A Queue that additionally supports operations that wait for the queue to become non-empty when retrieving an element, and wait for space to become available in the queue when storing an element.
Callable<V>	A task that returns a result and may throw an exception.
CompletableFuture.AsynchronousCompletionTask	A marker interface identifying asynchronous tasks produced by async methods.
CompletionService<V>	A service that decouples the production of new asynchronous tasks from the consumption of the results of completed tasks.
CompletionStage<T>	A stage of a possibly asynchronous computation, that performs an action or computes a value when another CompletionStage completes.
ConcurrentMap<K,V>	A Map providing thread safety and atomicity guarantees.

See docs.oracle.com/javase/8/docs/api/java/util/concurrent/package-summary.html

Types of Java Synchronizer Capabilities

- Java synchronizers provide various types of capabilities, e.g.
  - Atomic ordering
  - Mutual exclusion
  - Coordination
  - **Barrier synchronization**
    - Ensures that any thread(s) must stop at a certain point & cannot proceed until all thread(s) reach the barrier
    - Barrier synchronization is supported by the Java concurrent package
    - Barrier synchronization is also supported by the Thread.join() method



join

```
public final void join()
                throws InterruptedException
```

Waits for this thread to die.

An invocation of this method behaves in exactly the same way as the invocation

```
    join(0)
```

Throws:

`InterruptedException` - if any thread has interrupted the current thread. The *interrupted status* of the current thread is cleared when this exception is thrown.

See docs.oracle.com/javase/8/docs/api/java/lang/Thread.html#join

Types of Java Synchronizer Capabilities

- We'll cover all these types of Java synchronizers in this course!!

Category	Definition
Atomic operations	An action that effectively happens all at once or not at all
Mutual exclusion	Allows concurrent access & updates to shared mutable data without race conditions
Coordination	Ensures computations run properly, e.g., in the right order, at the right time, under the right conditions, etc.
Barrier synchronization	Ensures that any thread(s) must stop at a certain point & cannot proceed until all other thread(s) reach this barrier

End of Types of Java Synchronizer Capabilities (Part 2)