Key Methods of Java CyclicBarrier

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

• Understand the structure & functionality of Java CyclicBarrier

• Recognize the key methods in the Java CyclicBarrier

```java
CyclicBarrier

- CyclicBarrier(int, Runnable)
- CyclicBarrier(int)
- getParties(): int
- await(): int
- await(long, TimeUnit): int
- isBroken(): boolean
- reset(): void
```
Key Methods in Java CyclicBarrier
Overview of Java CyclicBarrier

- CyclicBarrier has a very simple API
- i.e., only a handful of methods are commonly used

```
<<Java Class>>

CyclicBarrier

- CyclicBarrier(int, Runnable)
- CyclicBarrier(int)
- getParties(): int
- await(): int
- await(long, TimeUnit): int
- isBroken(): boolean
- reset(): void
```
Overview of Java CyclicBarrier

- Constructor initializes the object to "trip" when the given # of parties wait on it

```java
public class CyclicBarrier {
    ...
    public CyclicBarrier (int parties) {
    }

    public CyclicBarrier (int parties,
                          Runnable barrierAction) {
        ...
    }
    ...
```
Overview of Java CyclicBarrier

- Constructor initializes the object to "trip" when the given # of parties wait on it.

```
public class CyclicBarrier {
    ...  
    public CyclicBarrier (int parties) {
    }
    ... 
    public CyclicBarrier (int parties, 
        Runnable barrierAction) {
    ...
    }
    ...
```

CyclicBarrier requires a fixed # of threads that is identical to the # of parties...
Overview of Java CyclicBarrier

- Constructor initializes the object to “trip” when the given # of parties wait on it
- Optionally given a barrier action to execute when barrier’s tripped

```java
class CyclicBarrier {
    public CyclicBarrier(int parties) {
        // ...}

    public CyclicBarrier(int parties, Runnable barrierAction) {
        // ...}
}
```
Overview of Java CyclicBarrier

- Constructor initializes the object to “trip” when the given # of parties wait on it
- Optionally given a *barrier action* to execute when barrier’s tripped
- Performed by the last thread entering the barrier

```java
class CyclicBarrier {
    public CyclicBarrier(int parties) {
    }

    public CyclicBarrier(
        int parties,
        Runnable barrierAction) {
        ...
    }
...}
```

*Parties are suspended when barrier action is run to avoid race conditions*
Overview of Java CyclicBarrier

- Constructor initializes the object to “trip” when the given # of parties wait on it
- Optionally given a *barrier action* to execute when barrier’s tripped
- Performed by the last thread entering the barrier
- Useful for updating any mutable shared state before any parties continue with their processing
- e.g., (re)initializing data structures, etc.

```java
public class CyclicBarrier {
    ...
    public CyclicBarrier (int parties) {
    }

    public CyclicBarrier
        (int parties,
        Runnable barrierAction) {
        ...
    }
}
```
Overview of Java CyclicBarrier

- Constructor initializes the object to “trip” when the given # of parties wait on it
- Optionally given a *barrier action* to execute when barrier’s tripped
  - Performed by the last thread entering the barrier
  - Useful for updating any mutable shared state before any parties continue with their processing
- The barrier’s count is automatically reset to initial # of parties after the barrier is tripped

```java
public class CyclicBarrier {
    ... public CyclicBarrier (int parties) {
    }

    public CyclicBarrier (int parties,
                        Runnable barrierAction) {
        ... ...
    }
}
```
Overview of Java CyclicBarrier

- Key methods block until all parties wait on the barrier & then reset it automatically after it’s tripped

```java
public class CyclicBarrier {
    ...
    public int await() { ... }

    public int await(long timeout, TimeUnit unit)
    { ... }
}
```

Threads calling `await()` decide whether to continue to the next cycle or not
Overview of Java CyclicBarrier

- Key methods block until all parties wait on the barrier & then reset it automatically after it’s tripped
- Block until all parties arrive & barrier resets
- *Unless* the thread is interrupted

```java
public class CyclicBarrier {
    ...
    public int await() { ... }
    ...
}
```
Overview of Java CyclicBarrier

- Key methods block until all parties wait on the barrier & then reset it automatically after it’s tripped
- Block until all parties arrive & barrier resets
- *Unless* the thread is interrupted

```java
public class CyclicBarrier {
    ...
    public int await() {
        ...
    }
    ...
}
```

Returns arrival index of the thread at the barrier:

```java
if (barrier.await() == 0) {
    // log completion of this iteration
}
```

Can be used in lieu of barrier action if parties need not be suspended when run
Overview of Java CyclicBarrier

- Key methods block until all parties wait on the barrier & then reset it automatically after it’s tripped
- Block until all parties arrive & barrier resets
  - Unless the thread is interrupted
  - Unless the timeout elapses
  - In which case await() throws the Java TimeoutException

```java
public class CyclicBarrier {
    ...
    public int await() {
        ...
    }
    public int await(long timeout, TimeUnit unit) {
        ...
    }
    ...
}
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/TimeoutException.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/TimeoutException.html)
Overview of Java CyclicBarrier

- Key methods block until all parties wait on the barrier & then reset it automatically after it’s tripped
- Block until all parties arrive & barrier resets

```java
public class CyclicBarrier {
    ...
    public int await() { ... }
    public int await(long timeout, TimeUnit unit)
    { ... }
    ...
}
```

There is no “non-interruptible” version of await()
Overview of Java CyclicBarrier

- It’s possible to manually reset a cyclic barrier to its initial state

```java
public class CyclicBarrier {
    ...
    public void reset() { ... }
    ...
}

If any parties are waiting at the barrier, they will return via a BrokenBarrierException rather than the “normal” return

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/BrokenBarrierException.html](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/BrokenBarrierException.html)
End of Key Methods of Java CyclicBarrier