Key Methods in Java Phaser

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 the Java Phaser barrier synchronizer
- Recognize the key methods in the Java Phaser
Key Methods in Java Phaser
Key Methods in Java Phaser

- Phaser has a more complex API than CountDownLatch or CyclicBarrier
- i.e., it has many methods that support a range of use cases
Key Methods in Java Phaser

- Phaser has a more complex API than CountDownLatch or CyclicBarrier
- i.e., it has many methods that support a range of use cases

<table>
<thead>
<tr>
<th>Java Class</th>
</tr>
</thead>
<tbody>
<tr>
<td>Phaser</td>
</tr>
<tr>
<td>Phaser()</td>
</tr>
<tr>
<td>Phaser(int)</td>
</tr>
<tr>
<td>Phaser(Phaser)</td>
</tr>
<tr>
<td>Phaser(Phaser,int)</td>
</tr>
<tr>
<td>register():int</td>
</tr>
<tr>
<td>bulkRegister(int):int</td>
</tr>
<tr>
<td>arrive():int</td>
</tr>
<tr>
<td>arriveAndDeregister():int</td>
</tr>
<tr>
<td>arriveAndWaitAdvance():int</td>
</tr>
<tr>
<td>awaitAdvance(int):int</td>
</tr>
<tr>
<td>awaitAdvanceInterruptibly(int):int</td>
</tr>
<tr>
<td>awaitAdvanceInterruptibly(int,long,TimeUnit):int</td>
</tr>
<tr>
<td>forceTermination():void</td>
</tr>
<tr>
<td>getPhase():int</td>
</tr>
<tr>
<td>getRegisteredParties():int</td>
</tr>
<tr>
<td>getArrivedParties():int</td>
</tr>
<tr>
<td>getUnarrivedParties():int</td>
</tr>
<tr>
<td>getParent():Phaser</td>
</tr>
<tr>
<td>getRoot():Phaser</td>
</tr>
<tr>
<td>isTerminated():boolean</td>
</tr>
<tr>
<td>onAdvance(int,int):boolean</td>
</tr>
<tr>
<td>toString()</td>
</tr>
</tbody>
</table>

Fortunately, many of these methods are rarely used in practice
Key Methods in Java Phaser

- Constructor initializes the phase # to 0

```java
public class Phaser {
    ...
    public Phaser(int parties) {
        ...
    }
    ...
    public Phaser() { ... }
    ...
```
Key Methods in Java Phaser

- Constructor initializes the phase # to 0
- This constructor specifies the # of parties needed to advance to the next phase

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

    public Phaser() { ... }
    ...
}
```

# of registered parties dictates when a phaser can advance to the next phase
Key Methods in Java Phaser

• Constructor initializes the phase # to 0
• This constructor specifies the # of parties needed to advance to the next phase
• This constructor is optional since parties can always register later

public class Phaser {
    ...
    public Phaser(int parties) {
        ...
    }
    public Phaser() {
        ...
    }
    ...

With Java Phaser the # of parties need not match the # of threads
Key Methods in Java Phaser

- Constructor initializes the phase # to 0
- This constructor specifies the # of parties needed to advance to the next phase
- This constructor doesn’t specify any parties initially

```java
public class Phaser {
    ...
    public Phaser(int parties) {
        ...
    }
    ...
    public Phaser() { ... }
    ...
}
```
Key Methods in Java Phaser

• Constructor initializes the phase # to 0
• This constructor specifies the # of parties needed to advance to the next phase
• This constructor doesn’t specify any parties initially
• Any phaser created via this constructor therefore needs to register with it before using it

```java
public class Phaser {
    ...
    public Phaser(int parties) {
        ...
    }
    public Phaser() { ... }
    ...
}
```
Key Methods in Java Phaser

Phaser’s key methods enable parties to register, synchronize, & terminate

```java
public class Phaser {
    ...
    public int register() { ... }

    public int bulkRegister(int parties) { ... }

    public int arriveAndAwaitAdvance() {
        ... }

    public int ArriveAndDeregister() {
        ... }

    protected boolean onAdvance(int phase, int registeredParties) {
        return registeredParties == 0;
    }
    ...
```
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
- Adds unarrived parties to phaser

```java
public class Phaser {
    ...
    public int register() { ... }
    public int bulkRegister(int parties) { ... }
}
```

# of registered parties dictates when a phaser can advance to the next phase
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
- Adds unarrived parties to phaser
- Arrive & await advance

```java
public class Phaser {
    ...  
    public int arrive() { ... }  
    public int awaitAdvance(int phase)  
        { ... }  
    public int arriveAndAwaitAdvance()  
        { ... }  
}
```

Having multiple methods provides flexibility wrt arrival & waiting to advance.
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
    - Arrives at phaser, but does not block until other parties arrive
  
```java
public class Phaser {
    ...
    public int arrive() { ... }
}
```

Can be used similarly to the `countdown()` method in `CountDownLatch`
Phaser’s key methods enable parties to register, synchronize, & terminate.

- Adds unarrived parties to phaser.
- Arrive & await advance.
  - Arrives at phaser, but does not block until other parties arrive.
  - Returns current phase # or a negative value if the phaser has already terminated.

This method is rarely used in practice.
Phaser’s key methods enable parties to register, synchronize, & terminate

- Adds unarrived parties to phaser
- Arrive & await advance
  - Arrives at phaser, but does not block until other parties arrive
  - Blocks until the phase of this phaser advances from the given phase value

Can be used similarly to the await() method in CountDownLatch
Key Methods in Java Phaser

• Phaser’s key methods enable parties to register, synchronize, & terminate
  • Adds unarrived parties to phaser
  • Arrive & await advance
    • Arrives at phaser, but does not block until other parties arrive
    • Blocks until the phase of this phaser advances from the given phase value
      • Returns immediately if current phase != given phase

public class Phaser {
  ...
  public int arrive() { ... }
  public int awaitAdvance(
    int phase
  )
  { ... }

This method is rarely used in practice
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
    - Arrives at phaser, but does not block until other parties arrive
    - Blocks until the phase of this phaser advances from the given phase value
    - Arrives at phaser & blocks until other parties arrive

```java
public class Phaser {
    ...
    public int arrive() { ... }
    public int awaitAdvance(int phase) {
        ...
    }
    public int arriveAndAwaitAdvance() {
        ...
    }
}
```

*Equivalent in effect to awaitAdvance(arrive())*
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
    - Arrives at phaser, but does not block until other parties arrive
    - Blocks until the phase of this phaser advances from the given phase value
    - Arrives at phaser & blocks until other parties arrive

```java
public class Phaser {
    ...
    public int arrive() { ... }
    public int awaitAdvance(int phase) {
        ... }
    public int arriveAndAwaitAdvance() {
        ... }
```

This method is commonly used & is similar to await() on a Java CyclicBarrier
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
  - Arrive at the phaser & deregister without waiting for others to arrive

```java
public class Phaser {
    ...
    public int arriveAndDeregister()
    {
        ...
    }
}
```
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
  - Arrive at the phaser & deregister without waiting for others to arrive
  - Reduces # of parties required to advance in future phases

```java
public class Phaser {
    ...
    public int arriveAndDeregister() {
        ...
    }
}
```

Often used by the party that controls the initialization of a phaser
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
  - Arrive at the phaser & deregister without waiting for others to arrive
  - Hook method performs an action upon pending phase advance

public class Phaser {
    ...  
    protected boolean onAdvance (
        int phase,
        int registeredParties) { 
        return registeredParties == 0;  
    }

This method is invoked upon arrival of the party advancing the phaser

All other waiting parties are “dormant” when this hook method runs
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
  - Adds unarrived parties to phaser
  - Arrive & await advance
  - Arrive at the phaser & deregister without waiting for others to arrive
  - Hook method performs an action upon pending phase advance

```java
public class Phaser {
    ...
    protected boolean onAdvance
        (int phase, int registeredParties) {
        return registeredParties == 0;
    }
}
```

This hook method is similar to the barrier action on a Java CyclicBarrier.
Phaser’s key methods enable parties to register, synchronize, & terminate

- Adds unarrived parties to phaser
- Arrive & await advance
- Arrive at the phaser & deregister without waiting for others to arrive
- Hook method performs an action upon pending phase advance
- Also terminates a Phaser by returning a ‘true’ boolean value

```
public class Phaser {
    ...
    protected boolean onAdvance (int phase, int registeredParties) {
        return registeredParties == 0;
    }
```
Key Methods in Java Phaser

- Phaser’s key methods enable parties to register, synchronize, & terminate
- Adds unarrived parties to phaser
- Arrive & await advance
- Arrive at the phaser & deregister without waiting for others to arrive
- Hook method performs an action upon pending phase advance
- Also terminates a Phaser by returning a ‘true’ boolean value

```java
public class Phaser {
    ...  
    protected boolean onAdvance (
        int phase, 
        int registeredParties) { 
        return registeredParties == 0;
    } 
}
```

*The default implementation terminates the phaser if there are no more registered parties*
End of Key Methods in Java Phaser