Example Application

Java CountDownLatch:
Learning Objectives in this Part of the Lesson

• Understand the structure & functionality of Java CountDownLatch
• Recognize the key methods in Java CountDownLatch
• Know how to program with Java CountDownLatch in practice

```java
class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                            CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
    }

    public void run() {
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown(); ...
    }
}
```
Overview of the GCD App
Overview of the GCD App

• This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms.

See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/CountDownLatch](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/CountDownLatch)
Overview of the GCD App

- This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms.
- GCD computes the largest positive integer that is a divisor of two numbers.
  - e.g., the GCD of 8 & 12 = 4

See en.wikipedia.org/wiki/Greatest_common_divisor
Overview of the GCD App

- This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
  - GCD computes the largest positive integer that is a divisor of two numbers
  - Four GCD algorithms are tested
Overview of the GCD App

• This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms

• GCD computes the largest positive integer that is a divisor of two numbers

• Four GCD algorithms are tested
  • The gcd() method defined by BigInteger

See docs.oracle.com/javase/8/docs/api/java/math/BigInteger.html#gcd
Overview of the GCD App

- This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
- GCD computes the largest positive integer that is a divisor of two numbers
- Four GCD algorithms are tested
  - The gcd() method defined by BigInteger
  - An iterative Euclid algorithm

See [en.wikipedia.org/wiki/Euclidean_algorithm](en.wikipedia.org/wiki/Euclidean_algorithm)
Overview of the GCD App

• This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
  • GCD computes the largest positive integer that is a divisor of two numbers
  • Four GCD algorithms are tested
    • The gcd() method defined by BigInteger
    • An iterative Euclid algorithm
    • A recursive Euclid algorithm

Overview of the GCD App

• This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
  • GCD computes the largest positive integer that is a divisor of two numbers
  • Four GCD algorithms are tested
    • The gcd() method defined by BigInteger
    • An iterative Euclid algorithm
    • A recursive Euclid algorithm
    • A complex GCD algorithm that uses binary arithmetic

See en.wikipedia.org/wiki/Binary_GCD_algorithm
Overview of the GCD App

• This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms

• GCD computes the largest positive integer that is a divisor of two numbers

• Four GCD algorithms are tested
  • The gcd() method defined by BigInteger
  • An iterative Euclid algorithm
  • A recursive Euclid algorithm
  • A complex GCD algorithm that uses binary arithmetic

However, the details of these algorithms are not important for our discussion.
GCDCountDownLatchTest
Class Walkthrough
GCDCountDownLatchTest Class Walkthrough

• Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest -> new Thread
            (new GCDCountDownLatchWorker
                (entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

See [GCD/CountDownLatch/app/src/test/java/edu/vandy/gcdtesttask/GCDCyclicBarrierTest.java](GCD/CountDownLatch/app/src/test/java/edu/vandy/gcdtesttask/GCDCyclicBarrierTest.java)
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ... 
        List<GCDTuple> gcdTests = makeGCDTuples();
        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
                new CountDownLatch(gcdTests.size());
        gcdTests.forEach(gcdTest -> new Thread
                (new GCDCountDownLatchWorker
                        (entryBarrier, exitBarrier, gcdTuple, this)).start());
        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
GCDCountDownLatchTest class walkthrough

- Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest ->
            new Thread(new GCDCountDownLatchWorker(
                entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

Create the entry barrier
GCDCountDownLatchTest Class Walkthrough

- Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest -> new Thread
            (new GCDCountDownLatchWorker
                (entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

Create worker threads using entry & exit barrier `CountDownLatch` objects.
Create the exit barrier.
GCDCountDownLatchTest Class Walkthrough

• Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...;
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
           new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest ->
            new Thread(new GCDCountDownLatchWorker(
                entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

• Iterate through all the GCD algorithms
GCDCountDownLatchTest Class Walkthrough

• Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest ->
            new Thread(
                new GCDCountDownLatchWorker
                    (entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

Create/start worker threads w/barriers
class GCDCountDownLatchTest {
  @Test public void testGCDCountDownLatchTester() {
    ...
    List<GCDTuple> gcdTests = makeGCDTuples();

    CountDownLatch entryBarrier = new CountDownLatch(1);
    CountDownLatch exitBarrier =
        new CountDownLatch(gcdTests.size());

    gcdTests.forEach(gcdTest -> new Thread
        (new GCDCountDownLatchWorker
            (entryBarrier, exitBarrier, gcdTuple, this)).start());

    System.out.println("Starting tests");
    entryBarrier.countDown();
    System.out.println("Waiting for results");
    exitBarrier.await();
    System.out.println("All tests done"); ...

• Create worker threads that use entry & exit barrier CountDownLatch objects
Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest -> new Thread
            (new GCDCountDownLatchWorker
                (entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();  // Let all worker threads proceed
        System.out.println("Waiting for results");
        exitBarrier.await();
        System.out.println("All tests done"); ...
    }
}
```

The countDown() method is a “latch” that lets all the worker threads start running, but it doesn’t ensure all the worker threads start at the same time.
GCDCountDownLatchTest Class Walkthrough

- Create worker threads that use entry & exit barrier CountDownLatch objects

```java
class GCDCountDownLatchTest {
    @Test public void testGCDCountDownLatchTester() {
        ...
        List<GCDTuple> gcdTests = makeGCDTuples();

        CountDownLatch entryBarrier = new CountDownLatch(1);
        CountDownLatch exitBarrier =
            new CountDownLatch(gcdTests.size());

        gcdTests.forEach(gcdTest ->
            new Thread(new GCDCountDownLatchWorker(
                entryBarrier, exitBarrier, gcdTuple, this)).start());

        System.out.println("Starting tests");
        entryBarrier.countDown();
        System.out.println("Waiting for results");
        exitBarrier.await();  // Wait for all to finish (exit barrier)
        System.out.println("All tests done"); ...
    }
}
```

After `await()` returns for a `CountDownLatch` it can’t be reused/reset without creating a new `CountDownLatch` instance.
GCDCountDownLatchWorker Class Walkthrough
This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation.

```java
class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                            CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
        ...
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown();
        ...
    }
}
```

See GCD/CountDownLatch/app/src/main/java/edu/vandy/gcdtesttask/presenter/GCDCountDownLatchWorker.java
GCDCountDownLatchWorker Class Walkthrough

- This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation.

class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                             CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
        ...
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown();
        ...
    }
}

Initialize barrier fields et al.
GCDCountDownLatchWorker Class Walkthrough

- This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation

```java
class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                            CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
        ...
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown();
        ...
    }
}
```

This hook method executes after the thread is started
GCDCountDownLatchWorker Class Walkthrough

- This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation.

```java
class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                            CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
        ...
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown();
        ...
    }
}
```

This entry barrier causes the worker thread to wait until main thread is ready, though worker threads may not start simultaneously.

See the upcoming lesson on “Java CyclicBarrier” for a solution to this problem.
GCDCountDownLatchWorker Class Walkthrough

• This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation

class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier,
                            CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
        ...
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();  // Run the GCD algorithm associated with this object
        mExitBarrier.countDown();
        ...
    }
}
GCDCountDownLatchWorker Class Walkthrough

• This class applies two entry & exit barrier CountDownLatch objects to coordinate the benchmarking of a given GCD algorithm implementation

```java
class GCDCountDownLatchWorker implements Runnable {
    private final CountDownLatch mEntryBarrier;
    private final CountDownLatch mExitBarrier;
    ...

    GCDCountDownLatchWorker(CountDownLatch entryBarrier, CountDownLatch exitBarrier, ...) {
        mEntryBarrier = entryBarrier; mExitBarrier = exitBarrier;
    }

    public void run() {
        ...
        mEntryBarrier.await();
        runTest();
        mExitBarrier.countDown();
        ...
    }
}
```

Decrement the count, which lets the main thread proceed when the count reaches 0
End of CountDownLatch: Example Application