Java CountDownLatch: Example Application

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

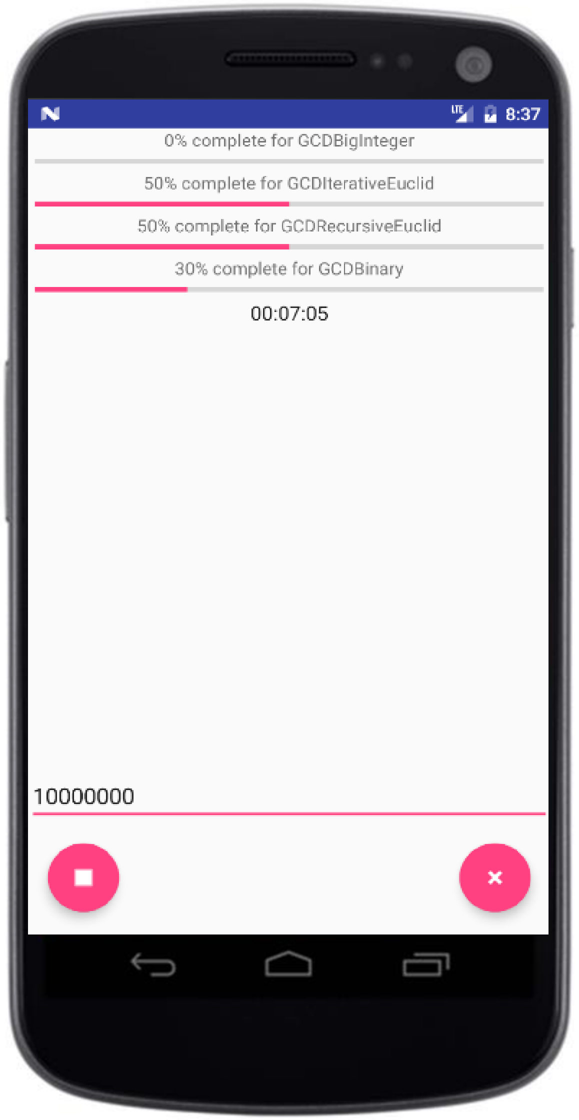
- Understand the structure & functionality of Java CountDownLatch
- Recognize the key methods in Java CountDownLatch
- Know how to program with Java CountDownLatch in practice

```
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



0% complete for GCDBigInteger

50% complete for GCDIterativeEuclid

50% complete for GCDRecursiveEuclid

30% complete for GCDBinary

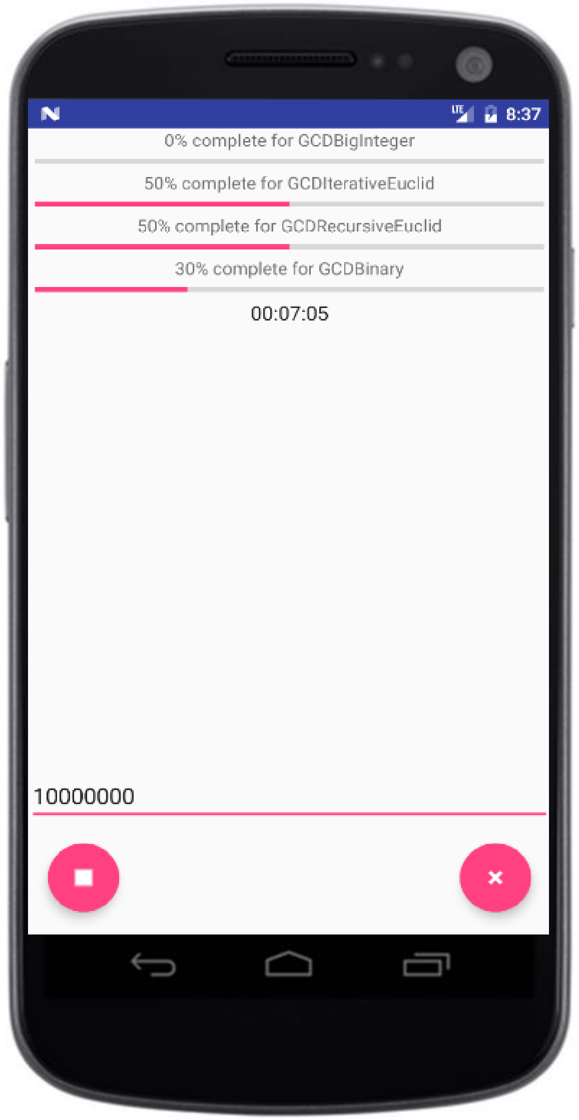
00:07:05

10000000

See github.com/douglascraigschmidt/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

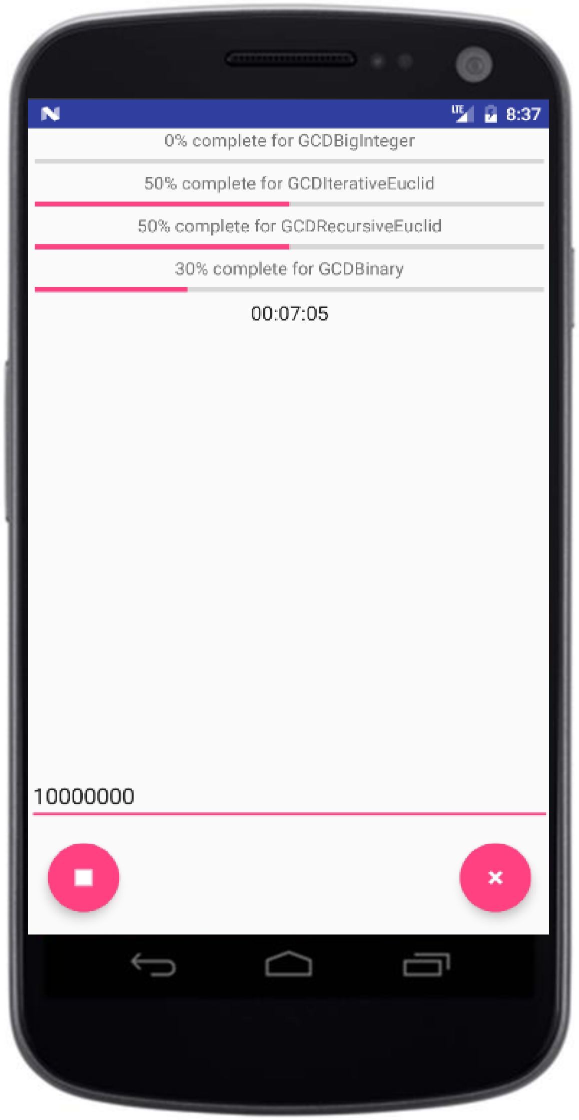


0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

See en.wikipedia.org/wiki/Greatest_common_divisor

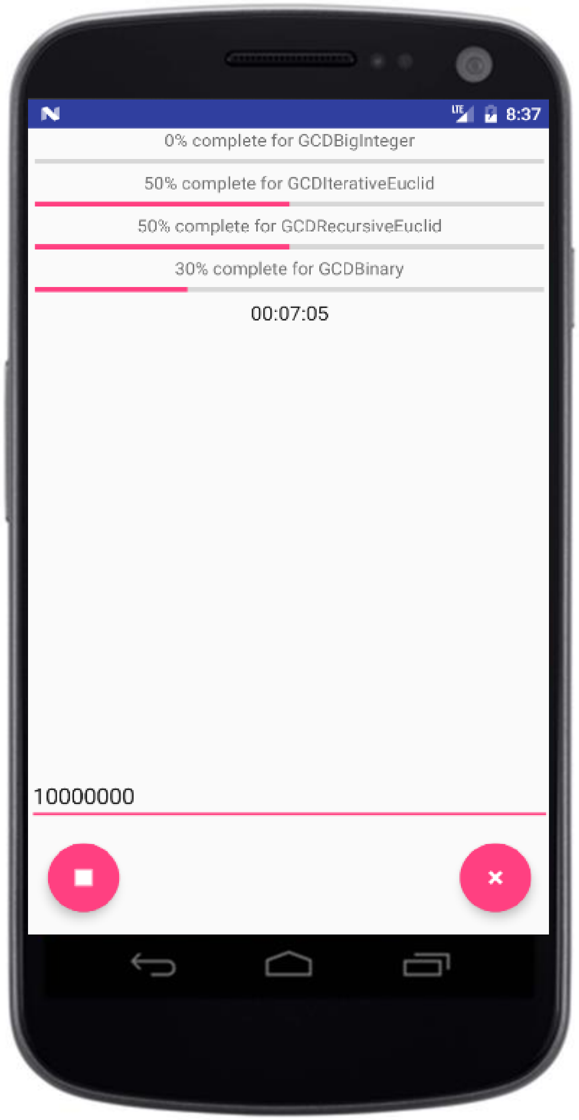
Overview of the GCD App

- 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


8:37
0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

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


8:37

0% complete for GCDBigInteger

50% complete for GCDIterativeEuclid

50% complete for GCDRecursiveEuclid

30% complete for GCDBinary

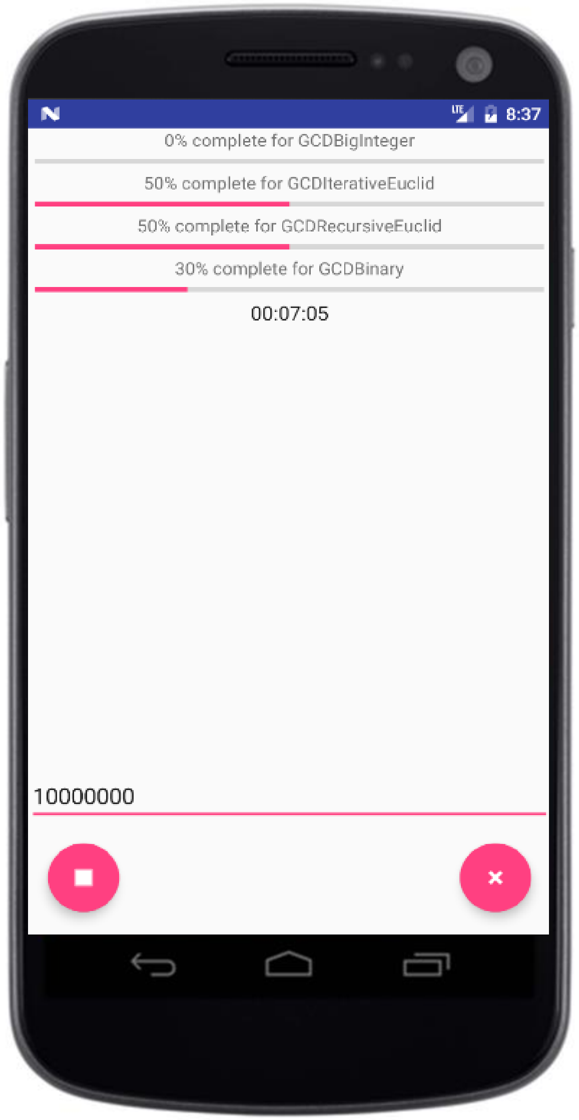
00:07:05

10000000

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

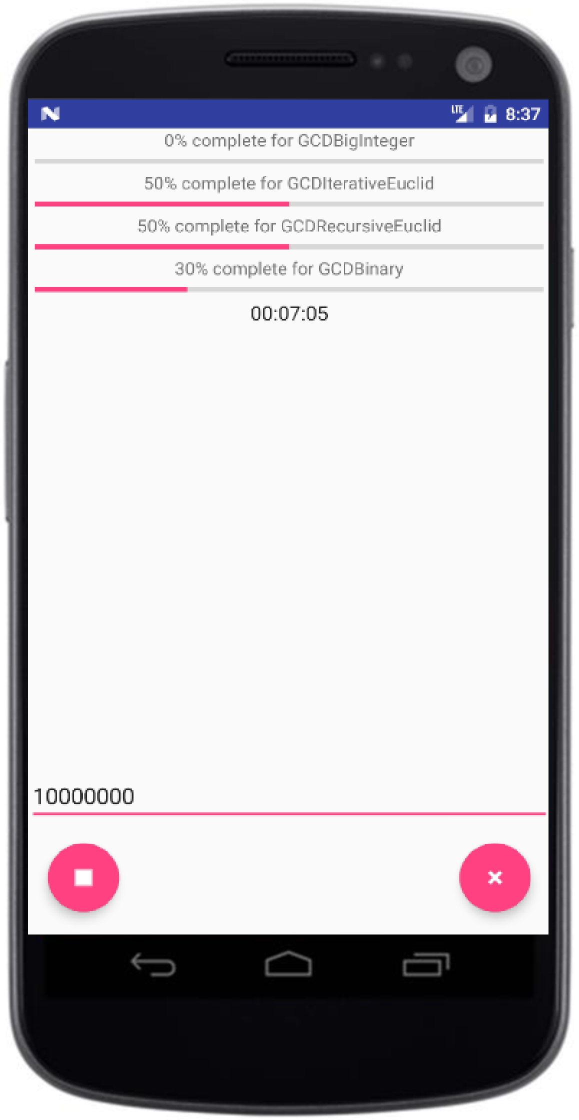


0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

See en.wikipedia.org/wiki/Euclidean_algorithm

Overview of the GCD App

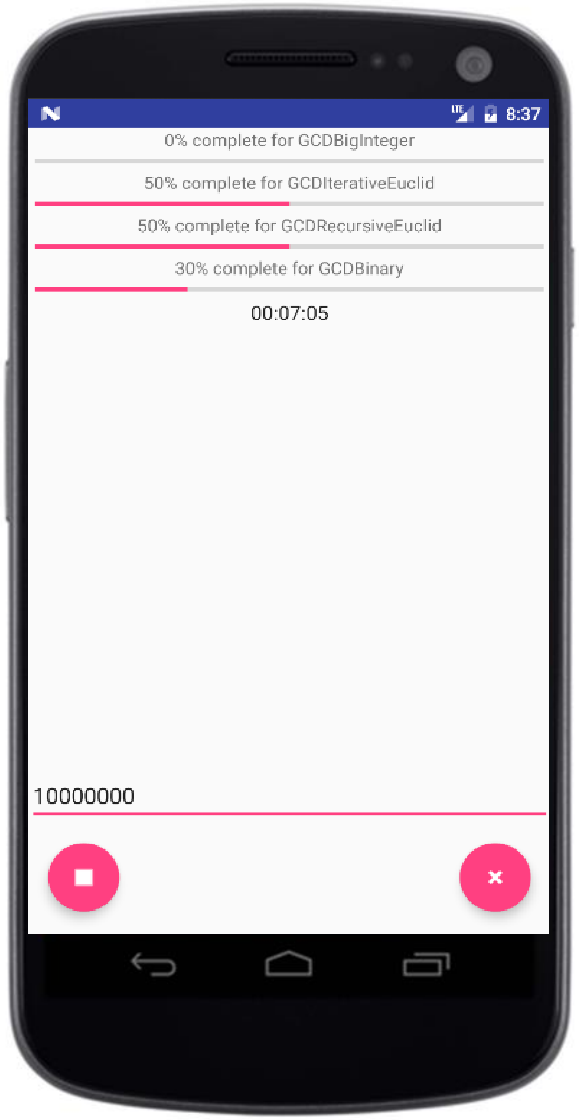
- This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
  - GCD computes the largest positive integer that is a divisor of two numbers
  - Four GCD algorithms are tested
    - The gcd() method defined by BigInteger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


0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

See codedost.com/java/methods-and-recursion-in-java/java-program-to-find-gcd-hcf-using-euclidean-algorithm-using-recursion

Overview of the GCD App

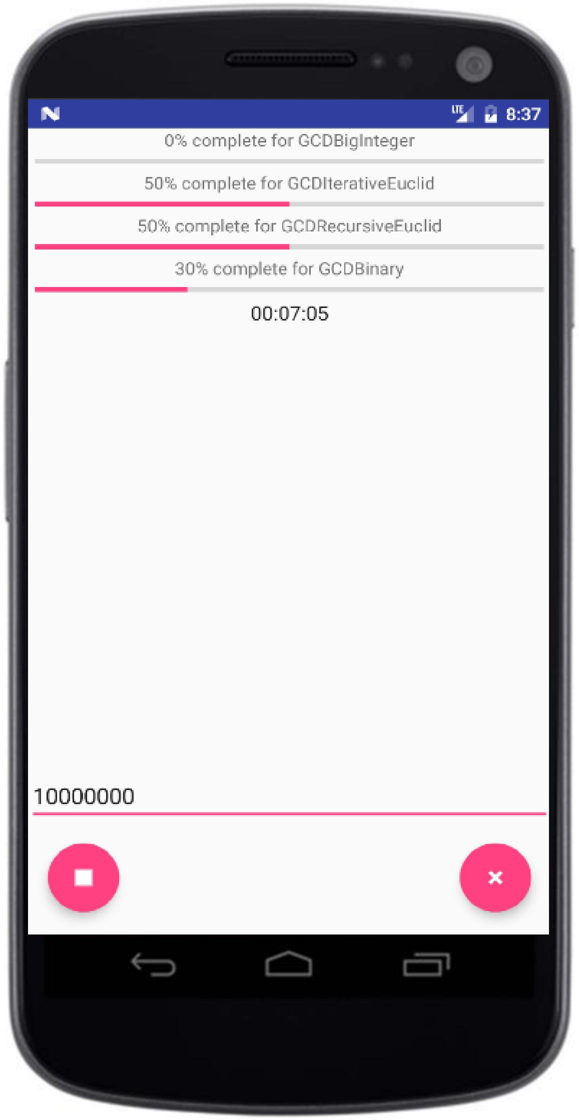
- This Android app uses two CountDownLatch objects to coordinate the concurrent benchmarking of four Greatest Common Divisor (GCD) algorithms
  - GCD computes the largest positive integer that is a divisor of two numbers
  - Four GCD algorithms are tested
    - The gcd() method defined by BigInteger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


8:37
0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

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 BigI ger
    - An iterative Euclid
    - A re
    - A c bina


0% complete for GCDBigInteger
50% complete for GCDIterativeEuclid
50% complete for GCDRecursiveEuclid
30% complete for GCDBinary
00:07:05
10000000

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

```
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]()

GCDCountDownLatchTest Class Walkthrough

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

```
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"); ...
```

Entry point into test

GCDCountDownLatchTest Class Walkthrough

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

```
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"); ...
```

Initialize all the GCD algorithms

GCDCountDownLatchTest Class Walkthrough

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

```
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

```
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 exit barrier

GCDCountDownLatchTest Class Walkthrough

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

```
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

```
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

GCDCountDownLatchTest Class Walkthrough

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

```
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"); ...
```

The worker threads don't start just yet

GCDCountDownLatchTest Class Walkthrough

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

```
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"); ...
```

Let all worker threads proceed

The countDown() method is a "latch" that let's 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

```
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"); ...
```

Wait for all to finish (exit barrier)

After await() returns for a CountDownLatch it can't be reused/ reset without creating a new CountDownLatch instance

GCDCountDownLatchWorker Class Walkthrough

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();
    ...
```

Define a worker that runs in a thread

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

```
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

```
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();
    mExitBarrier.countDown();
    ...
```

Run the GCD algorithm associated with this object

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();
    ...
```

Decrement the count, which lets the main thread proceed when the count reaches 0

End of CountDownLatch: Example Application