

# Java Thread: Overview of the Case Study App



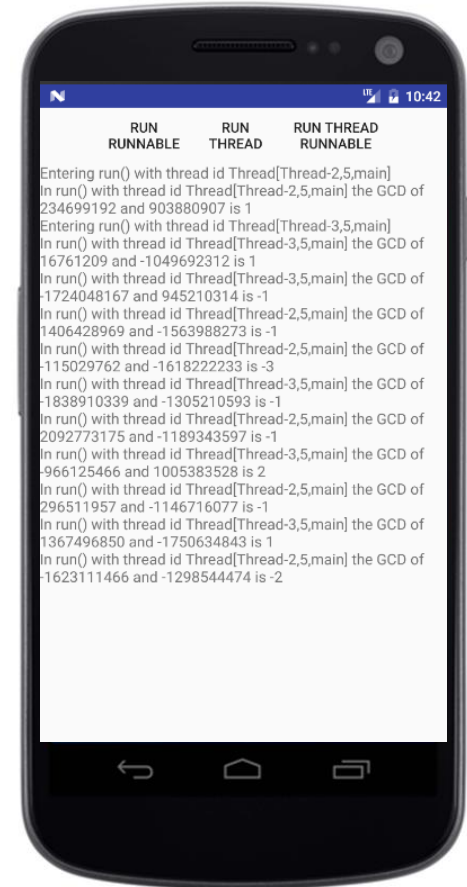
**Douglas C. Schmidt**  
**[d.schmidt@vanderbilt.edu](mailto:d.schmidt@vanderbilt.edu)**  
**[www.dre.vanderbilt.edu/~schmidt](http://www.dre.vanderbilt.edu/~schmidt)**

**Institute for Software  
Integrated Systems  
Vanderbilt University  
Nashville, Tennessee, USA**



# Learning Objectives in this Part of the Lesson

- Understand how Java threads support concurrency
- Learn how our case study app works



See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/Concurrent](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/Concurrent)

---

# Runtime Behavior of the GCD Concurrent App

# Runtime Behavior of the GCD Concurrent App

- Concurrently compute the greatest common divisor (GCD) of two #'s, which is the largest integer that divides two integers without a remainder



<<Java Class>>

**Thread**

```
● Syield():void
● ScurrentThread():Thread
● Ssleep(long):void
● Ssleep(long,int):void
● CThread()
● CThread(Runnable)
● CThread(String)
● start():void
● run():void
■ Sexit():void
● interrupt():void
● Sinterrupted():boolean
● isInterrupted():boolean
● FisAlive():boolean
● FsetPriority(int):void
● FgetPriority():int
● Fjoin(long):void
● Fjoin(long,int):void
● Fjoin():void
● FsetDaemon(boolean):void
● FisDaemon():boolean
```

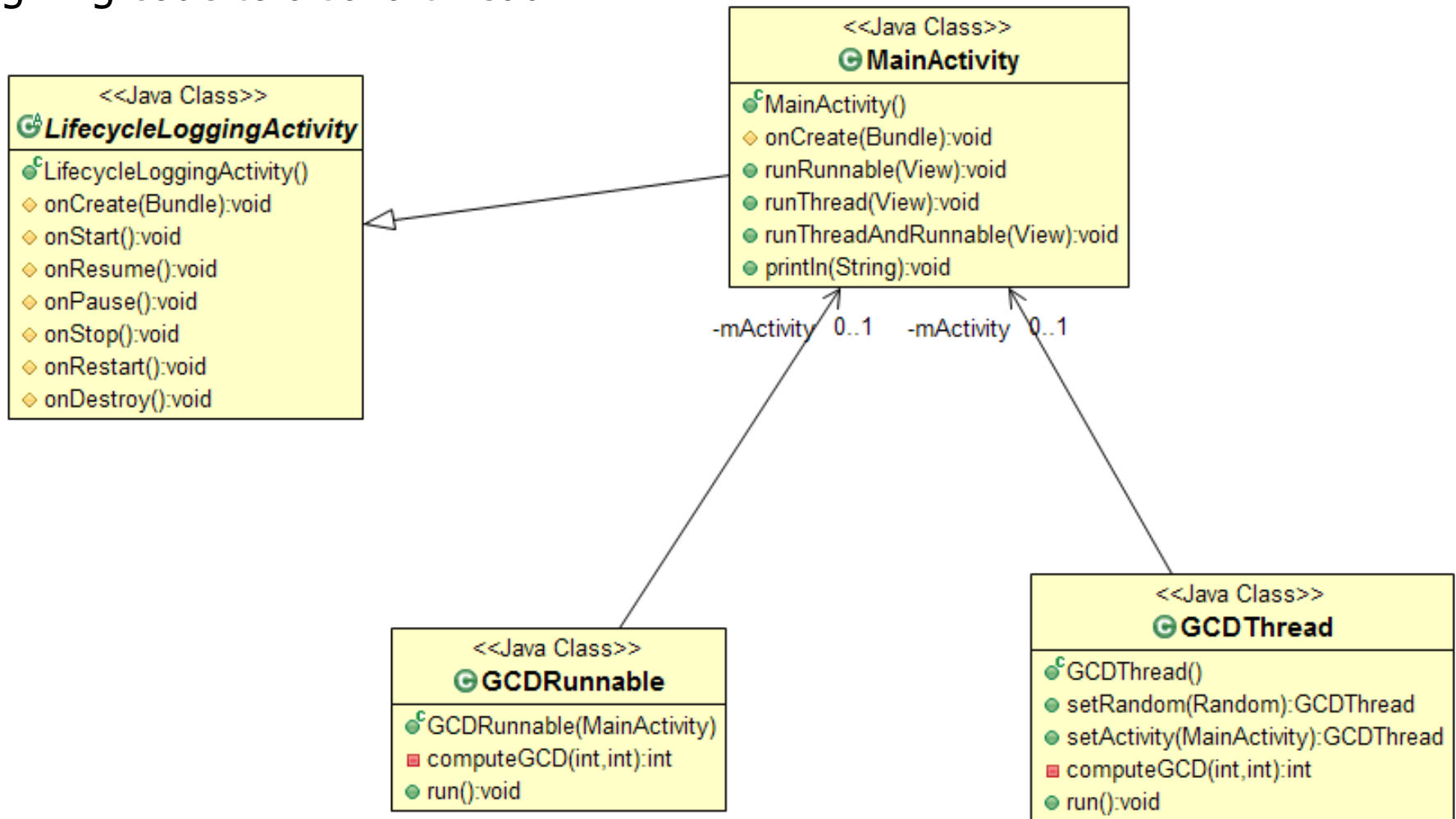
See [en.wikipedia.org/wiki/Greatest\\_common\\_divisor](https://en.wikipedia.org/wiki/Greatest_common_divisor)

---

# Design of the GCD Concurrent App

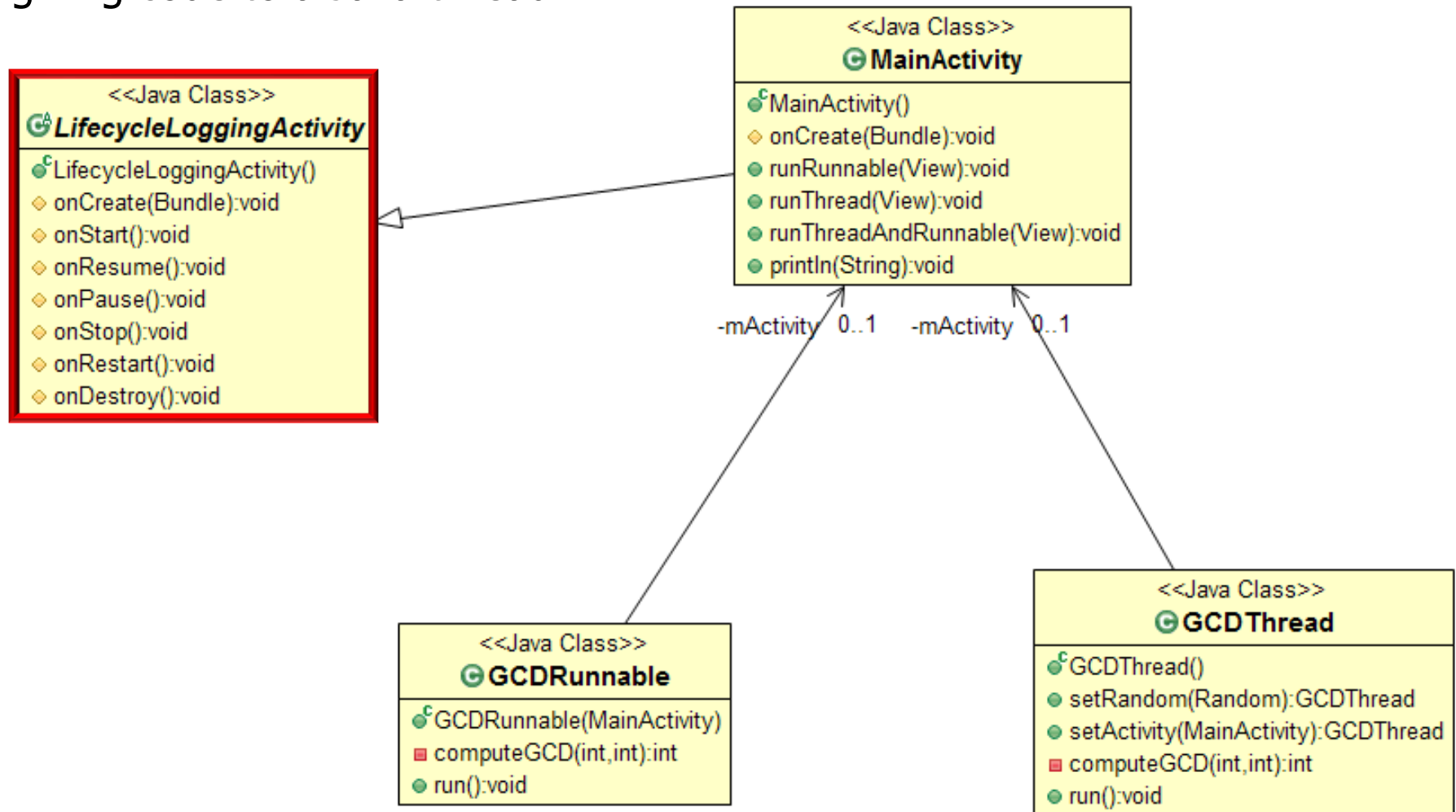
# Design of the GCD Concurrent App

- This app shows various methods in Java's Thread class & alternative ways of giving code to a Java thread



# Design of the GCD Concurrent App

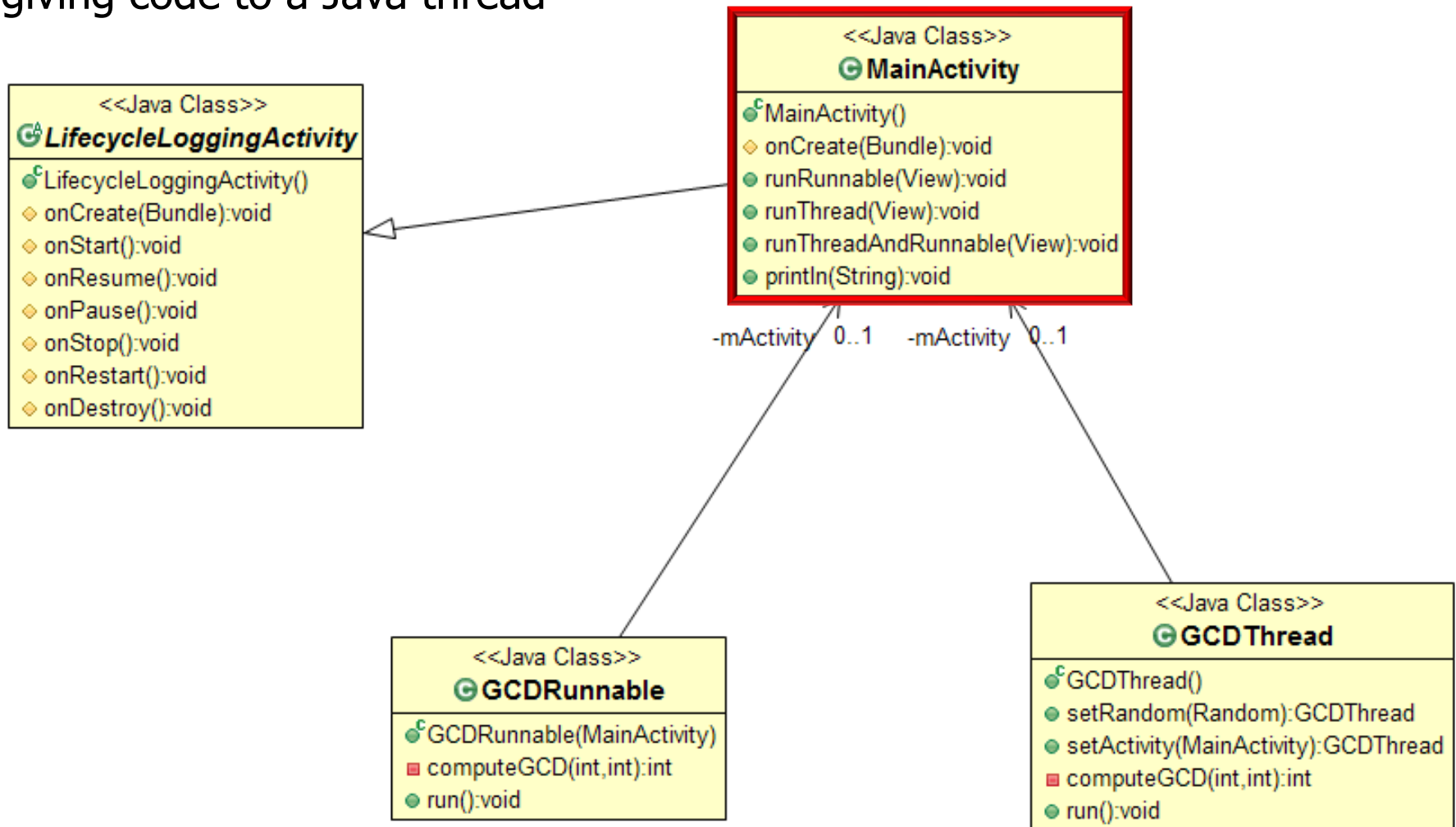
- This app shows various methods in Java's Thread class & alternative ways of giving code to a Java thread



Super class that logs various activity lifecycle hook methods to aid debugging

# Design of the GCD Concurrent App

- This app shows various methods in Java's Thread class & alternative ways of giving code to a Java thread

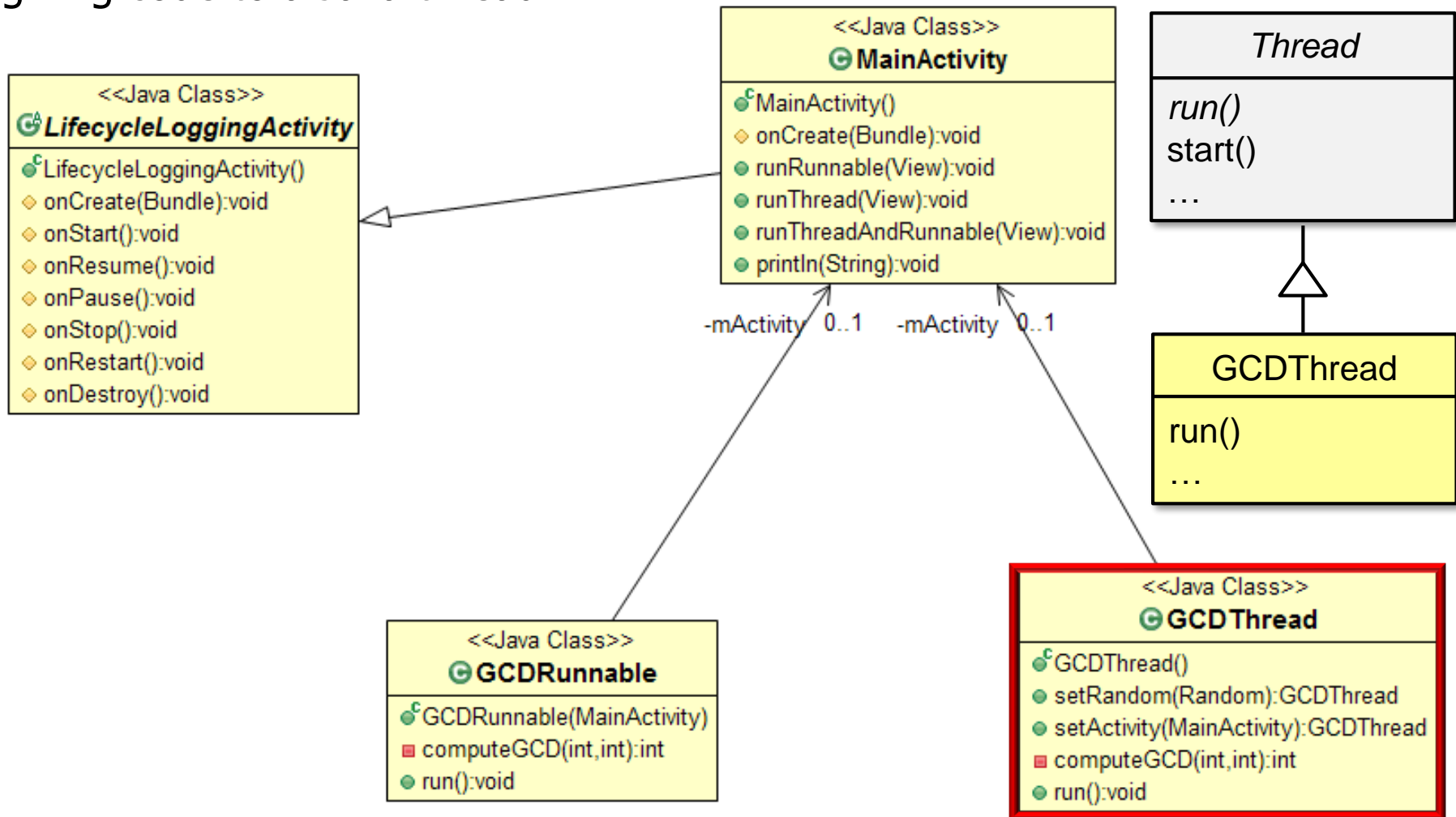


Main entry point into the app that handles button presses from the user



# Design of the GCD Concurrent App

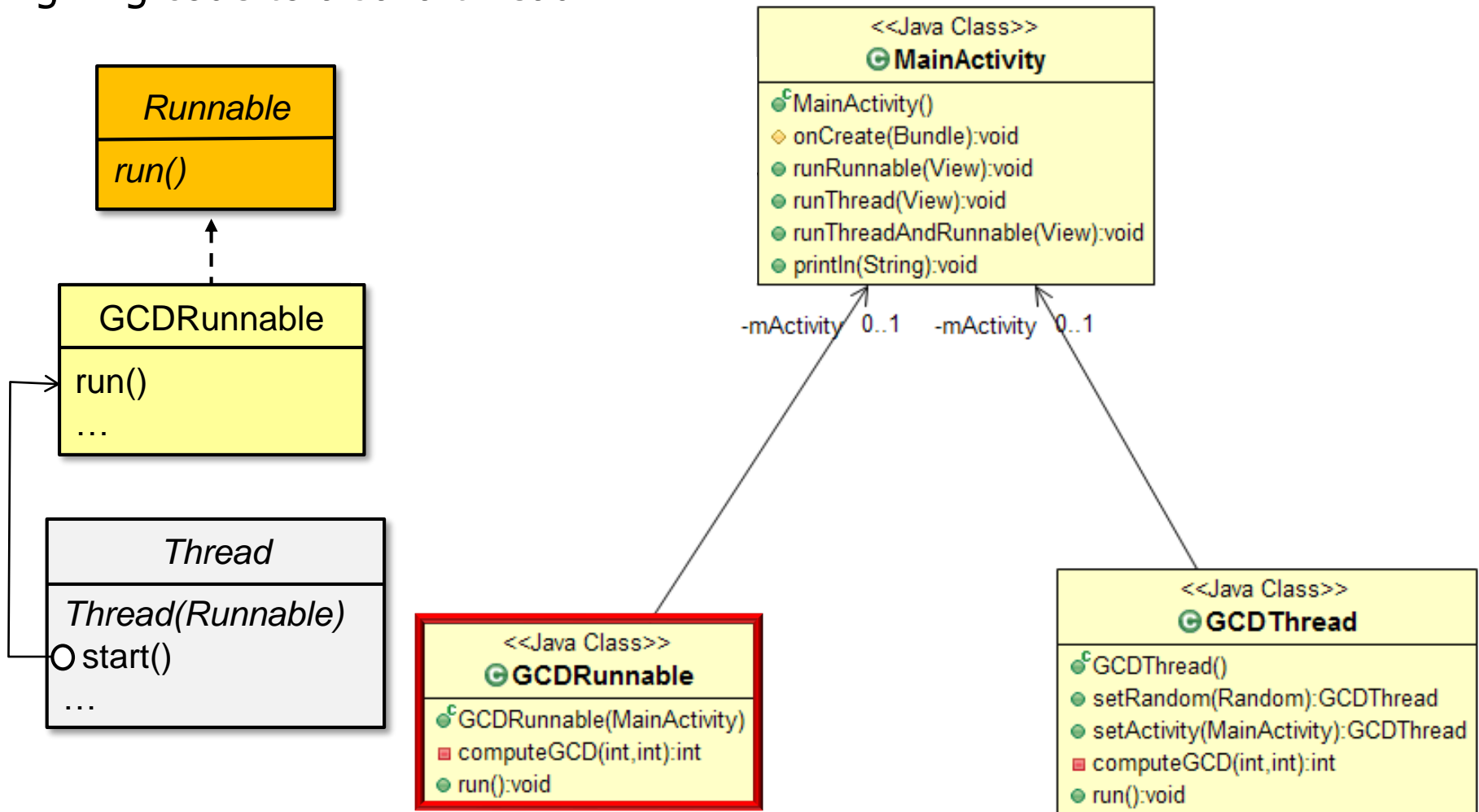
- This app shows various methods in Java's Thread class & alternative ways of giving code to a Java thread



Computes the GCD of two numbers by extending the Thread super class

# Design of the GCD Concurrent App

- This app shows various methods in Java's Thread class & alternative ways of giving code to a Java thread



Computes the GCD of two numbers by implementing the Runnable interface

# Design of the GCD Concurrent App

- We'll explore the implementations of these threading alternatives shortly

```
/**
 * Computes the greatest common divisor (GCD) of two numbers, which is
 * the largest positive integer that divides two integers without a
 * remainder. This implementation extends Random and implements the
 * Runnable interface's run() hook method.
 */
public class GCDRunnable
    extends Random // Inherits random number generation capabilities.
    implements Runnable {
    /**
     * A reference to the MainActivity.
     */
    private final MainActivity mActivity;

    /**
     * Number of times to iterate, which is 100 million to ensure the
     * program runs for a while.
     */
    private final int MAX_ITERATIONS = 100000000;

    /**
     * Number of times to iterate before calling print, which is 10
     * million to ensure the program runs for a while.
     */
    private final int MAX_PRINT_ITERATIONS = 10000000;

    /**
     * Hook method that runs for MAX_ITERATIONS computing the GCD of
     * randomly generated numbers.
     */
    public void run() {
        final String threadString = " with thread id " + Thread.currentThread();

        mActivity.println("Entering run()" + threadString);

        // Generate random numbers and compute their GCDs.

        for (int i = 0; i < MAX_ITERATIONS; ++i) {
            // Generate two random numbers.
            int number1 = nextInt();
            int number2 = nextInt();

            // Print results every 10 million iterations.
            if ((i % MAX_PRINT_ITERATIONS) == 0)
                mActivity.println("In run()"
                    + threadString
                    + " the GCD of "
                    + number1
                    + " and "
                    + number2
                    + " is "
                    + computeGCD(number1,
                        number2));
        }

        mActivity.println("Leaving run() " + threadString);
    }
}
```

```
/**
 * Computes the greatest common divisor (GCD) of two numbers, which is
 * the largest positive integer that divides two integers without a
 * remainder. This implementation extends Thread and overrides its
 * run() hook method.
 */
public class GCDThread
    extends Thread {
    /**
     * A reference to the MainActivity.
     */
    private MainActivity mActivity;

    /**
     * Generate random numbers.
     */
    private Random mRandom;

    /**
     * Number of times to iterate, which is 100 million to ensure the
     * program runs for a while.
     */
    private final int MAX_ITERATIONS = 100000000;

    /**
     * Number of times to iterate before calling print, which is 10
     * million to ensure the program runs for a while.
     */
    private final int MAX_PRINT_ITERATIONS = 10000000;

    /**
     * Hook method that runs for MAX_ITERATIONS computing the GCD of
     * randomly generated numbers.
     */
    public void run() {
        final String threadString = " with thread id " + Thread.currentThread();

        mActivity.println("Entering run()" + threadString);

        // Generate random numbers and compute their GCDs.

        for (int i = 0; i < MAX_ITERATIONS; ++i) {
            // Generate two random numbers.
            int number1 = mRandom.nextInt();
            int number2 = mRandom.nextInt();

            // Print results every 10 million iterations.
            if ((i % MAX_PRINT_ITERATIONS) == 0)
                mActivity.println("In run()"
                    + threadString + " the GCD of "
                    + number1 + " and " + number2 + " is "
                    + computeGCD(number1,
                        number2));
        }

        mActivity.println("Leaving run() " + threadString);
    }
}
```

See [github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/Concurrent](https://github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/Concurrent)

# Design of the GCD Concurrent App

- First, however, we'll show how to build & run the app

The screenshot displays the Android Studio interface for the 'Concurrent' project. The left sidebar shows the project structure with the following hierarchy:

- app
  - manifests
  - java
    - vandy.mooc (androidTest)
    - vandy.mooc (test)
      - vandy.mooc.gcd
        - activities
          - GCDRunnable
          - GCDThread
          - LifecycleLoggingActivity
          - MainActivity
        - utils
      - java (generated)
      - res
      - res (generated)
    - Gradle Scripts

The main editor shows the `GCDThread.java` file with the following code:

```
1 package vandy.mooc.gcd.activities;
2
3 import java.util.Random;
4
5 /**
6  * Computes the greatest common divisor
7  * the largest positive integer that divides
8  * remainder. This implements the
9  * run() hook method.
10 */
11 public class GCDThread
12     extends Thread {
13
14     /**
15      * A reference to the MainActivity
16      */
17     private MainActivity mActivity;
18
19     /**
20      * Generate random numbers
21      */
22     private Random random;
```

Two virtual device emulators are shown. The left emulator displays a screen with three buttons: 'RUN RUNNABLE', 'RUN THREAD', and 'RUN THREAD RUNNABLE'. The right emulator displays a logcat window with the following output:

```
Entering run() with thread id Thread[Thread-2,5,main]
In run() with thread id Thread[Thread-2,5,main] the GCD of
234699192 and 903880907 is 1
Entering run() with thread id Thread[Thread-3,5,main]
In run() with thread id Thread[Thread-3,5,main] the GCD of
16761209 and -1049692312 is 1
In run() with thread id Thread[Thread-3,5,main] the GCD of
-1724048167 and 945210314 is -1
In run() with thread id Thread[Thread-2,5,main] the GCD of
1406428969 and -156398273 is -1
In run() with thread id Thread[Thread-2,5,main] the GCD of
-115029762 and -1618222233 is -3
In run() with thread id Thread[Thread-3,5,main] the GCD of
-1838910339 and -1305210593 is -1
In run() with thread id Thread[Thread-2,5,main] the GCD of
2092773175 and -1189343597 is -1
In run() with thread id Thread[Thread-3,5,main] the GCD of
-966125466 and 1005383528 is 2
In run() with thread id Thread[Thread-2,5,main] the GCD of
296511957 and -1146716077 is -1
In run() with thread id Thread[Thread-3,5,main] the GCD of
1367496850 and -1750634843 is 1
In run() with thread id Thread[Thread-2,5,main] the GCD of
-1623111466 and -1298544474 is -2
```

---

# End of Java Thread: Overview of the Case Study App