Overview of Java Threads

(Part 3)

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 how Java threads support concurrency
- Learn how our case study app works
- Know alternative ways of giving code to a thread
- Learn how to pass parameters to a Java thread
- Know how to run a Java thread
- Recognize common thread mechanisms
- Appreciate Java thread “happens-before” orderings
- Understand the implementation of the GCD concurrent app
Learning Objectives in this Part of the Lesson

• Understand how Java threads support concurrency
• Learn how our case study app works
• Know alternative ways of giving code to a thread
• Learn how to pass parameters to a Java thread
• Know how to run a Java thread
• Recognize common thread mechanisms
• Appreciate Java thread “happens-before” orderings
• Understand the implementation of the GCD concurrent app
• Know the pros & cons of Java thread programming models
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

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

• This app shows various Java Thread methods & alternative implementations

See [github.com/stephenmlewis/POSAGCD](https://github.com/stephenmlewis/POSAGCD)
Implementation of the GCD Concurrent App

- This app shows various Java Thread methods & alternative implementations

Super class that logs various activity lifecycle hook methods to aid debugging
Implementation of the GCD Concurrent App

- This app shows various Java Thread methods & alternative implementations

Main entry point into the app that handles button presses from the user
Implementation of the GCD Concurrent App

- This app shows various Java Thread methods & alternative implementations

Computes the GCD of two numbers by extending the Thread super class
Implementation of the GCD Concurrent App

• This app shows various Java Thread methods & alternative implementations

Computes the GCD of two numbers by implementing the Runnable interface
Implementation of the GCD Concurrent App

• This app shows various Java Thread methods & alternative implementations

```java
/**
 * 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 mMainActivity;
    /**
     * 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();
        mMainActivity.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)
                mMainActivity.println("In run() " + threadString + " the GCD of "
                        + number1 + " and " + number2 + " is "
                        + computeGCD(number1, number2));
        }
        mMainActivity.println("Leaving run() " + threadString);
    }
}
```

```java
/**
 * 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 mMainActivity;
    /**
     * 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();
        mMainActivity.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)
                mMainActivity.println("In run() " + threadString + " the GCD of "
                        + number1 + " and " + number2 + " is "
                        + computeGCD(number1, number2));
        }
        mMainActivity.println("Leaving run() " + threadString);
    }
}
```

Pros & Cons of Java Thread Programming Models
Pros & Cons of Java Thread Programming Models

- Now that we’ve examined the source code for the GCD concurrent app we’ll summarize the pros & cons of the various Java thread programming models.
Pros & Cons of Java Thread Programming Models

• Pros with extending Thread

```java
public class GCDThread extends Thread {

    private MainActivity mMainActivity;

    public GCDThread setActivity (MainActivity activity) {
        mMainActivity = activity;
        return this;
    }

    private int computeGCD (int number1, number2) {
        ...
    }

    public void run() {
        ...
    }
}
```
Pros & Cons of Java Thread Programming Models

- Pros with extending Thread
  - It’s straightforward to extend the Thread super class

```java
public class GCDThread
    extends Thread {
    ...
    private MainActivity mActivity;

    public GCDThread setActivity(
        MainActivity activity) {
        mActivity = activity;
        return this;
    }

    private int computeGCD(
        int number1, number2) {
        ...
    }

    public void run() {
        ...
    }
}
```
Pros & Cons of Java Thread Programming Models

• Pros with extending Thread
  • It’s straightforward to extend the Thread super class
  • Just override the run() hook method!

```java
public class GCDThread extends Thread {
    ...
    private MainActivity mActivity;
    public GCDThread setActivity (MainActivity activity) {
        mActivity = activity;
        return this;
    }

    private int computeGCD (int number1, number2) {
        ...
    }

    public void run ()
    {
        ...
    }
}
```
Pros & Cons of Java Thread Programming Models

- Pros with extending Thread
  - It’s straightforward to extend the Thread super class
  - It also consolidates all state & methods in one place

```java
public class GCDThread extends Thread {
    ...
    private MainActivity mActivity;

    public GCDThread setActivity (MainActivity activity) {
        mActivity = activity;
        return this;
    }
    ...

    // Main app
    Thread thread = new GCDThread()
        .setActivity(this)...

    thread.start();
    ...
```
Pros & Cons of Java Thread Programming Models

• Pros with extending Thread
  • It’s straightforward to extend the Thread super class
  • It also consolidates all state & methods in one place
  • Enables central allocation & management of the thread

```java
public class GCDThread extends Thread {
    ...
    private MainActivity mActivity;

    public GCDThread setActivity (MainActivity activity) {
        mActivity = activity;
        return this;
    }
    ...

    // Main app
    Thread thread = new GCDThread()
        .setActivity(this)....;

    thread.start();
    ...
```
Pros & Cons of Java Thread Programming Models

• Pros with extending Thread
  • It’s straightforward to extend the Thread super class
  • It also consolidates all state & methods in one place
  • Enables central allocation & management of the thread
  • This design is useful when the thread must be updated during runtime configuration changes

public class GCDThread extends Thread {
    ...
    private MainActivity mActivity;

    public GCDThread setActivity (MainActivity activity) {
        mActivity = activity;
        return this;
    }
    ...

    // Main app
    Thread thread = new GCDThread()
        .setActivity(this)...;

    thread.start();

    ...
}
Pros & Cons of Java Thread Programming Models

- **Pros with extending Thread**
  - It’s straightforward to extend the Thread super class
  - It also consolidates all state & methods in one place
  - Enables central allocation & management of the thread
  - This design is useful when the thread must be updated during runtime configuration changes
    - e.g., interrupting/restarting a running thread & reading/writing its state

```java
public class GCDThread extends Thread {
    ...

    private MainActivity mAActivity;

    public GCDThread setActivity (MainActivity activity) {
        mAActivity = activity;
        return this;
    }

    ...

    // Main app
    Thread thread = new GCDThread()
        .setActivity(this)...

    thread.start();

    ...
```

See the upcoming lessons on “Managing the Java Lifecycle” & “Managing Multi-threaded Activity State”
Pros & Cons of Java Thread Programming Models

• Cons with extending Thread

```java
public class GCDThread extends Thread {
    ...
    private int computeGCD(int number1, number2) {
        ...
    }
    public void run() {
        ...
    }
    ...
}
```
Pros & Cons of Java Thread Programming Models

- Cons with extending Thread
  - A subclass must extend the Thread superclass

```java
public class GCDThread extends Thread {
    ...
    private int computeGCD(int number1, number2) {
        ...
    }

    public void run() {
        ...
    }
}
```
Pros & Cons of Java Thread Programming Models

- Cons with extending Thread
  - A subclass must extend the Thread superclass
  - This is restrictive since Java only allows one superclass per subclass!

```java
public class GCDThread extends Thread {
  ...

  private int computeGCD(int number1, number2) {
    ...
  }

  public void run() {
    ...
  }
}
```

See docs.oracle.com/javase/tutorial/java/IandI/subclasses.html
Pros & Cons of Java Thread Programming Models

• Pros of implementing Runnable

```java
public class GCDRunnable
    implements Runnable,
    implements Serializable,
    extends Random {

    ... // Code

    private int computeGCD
        (int number1, number2) {
    ... // Code

    }

    public void run() {
    ... // Code

    }

    ... // Code
```
Pros & Cons of Java Thread Programming Models

- Pros of implementing Runnable
  - A subclass can implement multiple interfaces

```java
public class GCDRunnable
    implements Runnable, Serializable, Random {

    private int computeGCD
        (int number1, number2) {

        ...
    }

    public void run() {

        ...
    }

    ...

See docs.oracle.com/javase/tutorial/java/concepts/interface.html
Pros & Cons of Java Thread Programming Models

- Pros of implementing Runnable
  - A subclass can implement multiple interfaces
  - Which enables it to extend a different superclass

```java
public class GCDRunnable implements Runnable, Serializable, extends Random {
    private int computeGCD(int number1, number2) {
        ...
    }

    public void run() {
        ...
    }
    ...
}
```

See [docs.oracle.com/javase/tutorial/java/concepts/interface.html](docs.oracle.com/javase/tutorial/java/concepts/interface.html)
Pros & Cons of Java Thread Programming Models

- Pros of implementing Runnable
  - A subclass can implement multiple interfaces
  - Runnables are flexible since they can be reused in other contexts

```java
public class GCDRunnable implements Runnable, ...
    ...
    ...
    GCDRunnable runnableCommand = new GCDRunnable(...);

    ExecutorService executor = Executors.newFixedThreadPool
        (POOL_SIZE);
    ...
    executor.execute (runnableCommand);
```

See upcoming lesson on “the Java Executor framework”
Pros & Cons of Java Thread Programming Models

• Cons of implementing Runnable

```java
public class GCDRunnable
    implements Runnable,
    ...
    {
        ...
    }
    ...

    GCDRunnable runnableCommand =
        new GCDRunnable(...);

    Thread thr =
        new Thread(runnableCommand);
    ...
    thr.start();
```
Pros & Cons of Java Thread Programming Models

- Cons of implementing Runnable
- Yields more “moving parts”

```java
public class GCDRunnable
    implements Runnable,
        ...
    {
        ...
    }
...
GCDRunnable runnableCommand = new GCDRunnable(...);
Thread thr =
    new Thread(runnableCommand);
...
thr.start();
```
Pros & Cons of Java Thread Programming Models

• Cons of implementing Runnable
  • Yields more “moving parts”
    • e.g., Runnable & Thread are separate entities & must be managed/accessed separately

This decoupling get complicated if a program needs to access the state of a runnable, but only holds a reference to the thread object..

```java
public class GCDRunnable implements Runnable,
    ...
{
    ...
}

GCDRunnable runnableCommand =
    new GCDRunnable(...);

Thread thr =
    new Thread(runnableCommand);
...
thr.start();
```
Pros & Cons of Java Thread Programming Models

- In practice, Java & Android software often implements Runnable rather than extending Thread.
Pros & Cons of Java Thread Programming Models

• In practice, Java & Android software often implements Runnable rather than extending Thread

• Lambda expressions are becoming popular with Java 8-based platforms

End of Overview of Java Threads (Part 3)