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](en.wikipedia.org/wiki/Greatest_common_divisor)
Implementation of the GCD Concurrent App
Implementation of the GCD Concurrent App

- This app showcases various Java Thread methods & alternative ways of giving code to a Java thread

Implementation of the GCD Concurrent App

- This app showcases various Java Thread methods & alternative ways of giving code to a Java thread

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

- This app showcases various Java Thread methods & alternative ways of giving code to a Java thread

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

- This app showcases various Java Thread methods & alternative ways of giving code to a Java thread

```
<<Java Class>>
MainActivity

- MainActivity()
- onCreate(Bundle): void
- runRunnable(View): void
- runThread(View): void
- runThreadAndRunnable(View): void
- println(String): void

-mActivity 0..1
```

```
<<Java Class>>
LifecycleLoggingActivity

- LifecycleLoggingActivity()
- onCreate(Bundle): void
- onStart(): void
- onResume(): void
- onPause(): void
- onStop(): void
- onRestart(): void
- onDestroy(): void
```

```
<<Java Class>>
GCDRunnable

- GCDRunnable(MainActivity)
- computeGCD(int, int): int
- run(): void
```

```
<<Java Class>>
GCDThread

- GCDThread()
- setRandom(Random): GCDThread
- setActivity(MainActivity): GCDThread
- computeGCD(int, int): int
- run(): void
```

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

- This app showcases various Java Thread methods & alternative ways of giving code to a Java thread

**Class Diagram:**

**LifecycleLoggingActivity**
- LifecycleLoggingActivity()
- onCreate(Bundle): void
- onStart(): void
- onResume(): void
- onPause(): void
- onStop(): void
- onRestart(): void
- onDestroy(): void

**MainActivity**
- MainActivity()
- onCreate(Bundle): void
- runRunnable(View): void
- runThread(View): void
- runThreadAndRunnable(View): void
- println(String): void

**GCDThread**
- GCDThread()
- setRandom(Random): GCDThread
- setActivity(MainActivity): GCDThread
- computeGCD(int, int): int
- run(): void

**GCDRunnable**
- GCDRunnable(MainActivity)
- computeGCD(int, int): int
- run(): void

 Computes the GCD of two numbers by implementing the Runnable interface
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 mActivity;

    public void setActivity (MainActivity activity) {
        mActivity = activity;
    }

    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 void setActivity (MainActivity activity) {
        mActivity = activity;
    }

    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 void setActivity(MainActivity activity) {
    mActivity = activity;
  }

  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
  • This is useful when the thread must be manipulated during runtime configuration changes

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

    public void setActivity(MainActivity activity) {
        mActivity = activity;
    }

    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
• This is useful when the thread must be manipulated during runtime configuration changes
  • e.g., interrupting/restarting a running thread & reading/writing its state

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

    public void setActivity(MainActivity activity) {
        mActivity = activity;
    }

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

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

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() {
        ...
    }
    ...
}
```
• 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 {

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

    public void run() {
        ...
    }

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

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

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

See docs.oracle.com/javase/tutorial/java/concepts/interface.html

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

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

  public void run() {
    ...
  }

  ...
```

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();
```
• 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

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

    GCDRunnable runnableCommand = new GCDRunnable(...);

    Thread thr =
            new Thread(runnableCommand);
...
    thr.start();
```

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

See www.drdobbs.com/jvm/lambda-expressions-in-java-8/240166764
End of Overview of Java Threads (Part 3)