Evaluating Java Thread Programming Models

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 methods
- Be aware of the different types of Java threads
- Know the pros & cons of Java thread programming models
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 mAActivity;

    public GCDThread setActivity (MainActivity activity) {
        mAActivity = 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 mAActivity;

    public GCDThread setActivity
        (MainActivity activity) {
        mAActivity = 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 setMainActivity(
      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
  • All state & methods are consolidated in one place

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

    public GCDThread setActivity
        (MainActivity activity) {
        m.Activity = 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
  • All state & methods are consolidated in one place
  • Enables central allocation & management of the thread

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
  - All state & methods are consolidated in one place
  - Enables central allocation & management of the thread
  - This design is useful when the thread must be updated during runtime configuration changes

```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
  • All state & methods are consolidated 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 Thread Lifecycle”
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 {

    ...

    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

```
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
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
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 lessons 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 have become a popular to provide computations to threads on modern Java platforms

```java
new Thread(() ->
    System.out.println("hello world"))
    .start();
```

Define a computation that will run in a separate Java thread

End of Evaluating Java Thread Programming Models