Overview of Java Threads

(Part 1)

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

Concurrent apps use threads to simultaneously run multiple computations that potentially interact with each other.
Learning Objectives in this Part of the Lesson

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

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
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
Introduction to Java Threads
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java

See en.wikipedia.org/wiki/Thread_(computing)
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java

A process is a unit of resource allocation & protection

See en.wikipedia.org/wiki/Process_(computing)
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java

See docs.oracle.com/javase/tutorial/essential/concurrency/procthread.html
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java

Java threads running in the same process can communicate with each other via shared objects or message passing

Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java.

Java threads running in different processes can communicate with each other via shared memory or inter-process communication (IPC) mechanisms.

We’ll focus later on Android-centric forms of shared memory & IPC.
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java.

See en.wikipedia.org/wiki/Thread_(computing)#Processes.2C_kernel_threads.2C_user_threads.2C_and_fibers

Each Java thread leverages unique “state” from the underlying Linux kernel thread, e.g., a runtime stack, an instruction counter, & other registers.
Introduction to Java Threads

- Threads are the most basic way of obtaining concurrency in Java.

Java dynamic & static objects can be shared across Java threads (i.e., this “state” is common).

See en.wikipedia.org/wiki/Thread_(computing)#Processes.
  2C_kernel_threads.2C_user_threads.2C_and_fibers
The GCD Concurrent App Case Study
• This app shows various methods in Java’s Thread class & alternative ways of giving code to a Java thread

See github.com/douglasraigschmidt/POSA/tree/master/ex/M3/GCD/Concurrent
The GCD Concurrent App Case Study

- This app shows various methods in Java’s Thread class & alternative ways of giving code to a Java thread, e.g.
  - By implementing the Runnable interface

The GCD Concurrent App Case Study

- This app shows various methods in Java’s Thread class & alternative ways of giving code to a Java thread, e.g.
  - By implementing the Runnable interface
  - By inheriting from the Thread class

Ways of Giving Code to Java Threads
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run

```java
Thread t = new Thread();
t.start();
```

*Do not use the no argument Thread constructor directly!!!*

Ways of Giving Code to Java Threads

• Java threads *must* be given code to run

There are alternative programming models for giving code to Java threads
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.

  1. Extend the Thread class

```java
public class GCDThread extends Thread {
    public void run() {
        // code to run goes here
    }
}
```

See docs.oracle.com/javase/8/docs/api/java/lang/Thread.html
Ways of Giving Code to Java Threads

- Java threads must be given code to run, e.g.
  1. Extend the Thread class

```java
public class GCDThread extends Thread {
    public void run() {
        // code to run goes here
    }
}
```

Override the run() hook method in the subclass & define the thread's computations.

See [wiki.c2.com/?HookMethod](http://wiki.c2.com/?HookMethod)
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class

```java
public class GCDThread extends Thread {
    public void run() {
        // code to run goes here
    }
}

GCDThread gCDThread = new GCDThread();
gCDThread.start();
```

Create & start a thread using a named subclass of Thread
Ways of Giving Code to Java Threads

- Java threads must be given code to run, e.g.
  1. Extend the Thread class

```java
class GCDThread extends Thread {
    public void run() {
        // code to run goes here
    }
}
new GCDThread().start();
```

You can also write a one-liner to create & start an anonymous thread.
Ways of Giving Code to Java Threads

Java threads *must* be given code to run, e.g.

1. Extend the Thread class
2. Implement the Runnable interface

See [docs.oracle.com/javase/8/docs/api/java/lang/Thread.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Thread.html)
Ways of Giving Code to Java Threads

- Java threads must be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface

```java
Runnable
run()

GCDRunnable
run()
...
```

Implement the `run()` hook method of an interface to define the thread's computations

See [docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html](https://docs.oracle.com/javase/8/docs/api/java/lang/Runnable.html)
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface

```java
public class GCDRunnable implements Runnable {
    public void run() {
        // code to run goes here
    }
}

Runnable gCDRunnable = new GCDRunnable();
```

Create an instance of a named class as the runnable
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface

```java
public class GCDRunnable implements Runnable {
    public void run() {
        // code to run goes here
    }
}
Runnable gCDRunnable = new GCDRunnable();
new Thread(gCDRunnable).start();
```

Pass that runnable to a new thread object & start it
Ways of Giving Code to Java Threads

- Java threads must be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface

```java
new Thread(new Runnable() {
    public void run()
    {
        // code to run goes here
    }
}).start();
```

Create & start a thread using an anonymous inner class as the runnable
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface

```java
new Thread(new Runnable() {
    public void run(){
        // code to run goes here
    }
}).start();
```

This anonymous inner class idiom is used extensively in Java & Android code.
Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface
  3. Use Java 8 lambda expressions (variant of #2)

```java
Runnable run()

new Thread(() -> {
    // code to run goes here
}).start();
```

A lambda expression is an unnamed block of code (with optional parameters) that can be passed around & executed later

Ways of Giving Code to Java Threads

- Java threads *must* be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface
  3. Use Java 8 lambda expressions (variant of #2)

new Thread(() -> {
    // code to run goes here
}).start();

This approach is unwieldy if the code to run is long, complex, or needs to be used multiple times!
Ways of Giving Code to Java Threads

- Java threads must be given code to run, e.g.
  1. Extend the Thread class
  2. Implement the Runnable interface
  3. Use Java 8 lambda expressions (variant of #2)

Runnable `r = () -> {
    // code to run goes here
};
new Thread(r).start();

You can therefore store the runnable in a variable & pass it to the Thread constructor
Passing Parameters to a Java Thread
Passing Parameters to a Java Thread

- The `run()` methods defined in Java Thread & Runnable take no parameters.

This raises the question of how to pass parameters to a Java thread!
Passing Parameters to a Java Thread

- Parameters passed to run() can be supplied via one of two other means
Passing Parameters to a Java Thread

- Parameters passed to run() can be supplied via one of two other means, e.g.
  - As parameters to a class constructor

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

Parameters passed to run() can be supplied via one of two other means, e.g.

- As parameters to a class constructor

```java
public class GCDRunnable extends Random implements Runnable {
    private final MainActivity mActivity;
    ...
}
```

*Define field(s) to store parameters passed to a runnable or thread object*
Passing Parameters to a Java Thread

- Parameters passed to run() can be supplied via one of two other means, e.g.
- As parameters to a class constructor

```java
public class GCDRunnable extends Random implements Runnable {
    private final MainActivity mActivity;

    public GCDRunnable(MainActivity mainActivity) {
        mActivity = mainActivity;
    }

    ...
}
```

Add the parameter(s) to the constructor signature & store them in the field(s)
Passing Parameters to a Java Thread

• Parameters passed to run() can be supplied via one of two other means, e.g.
  • As parameters to a class constructor

```java
public class GCDRunnable extends Random implements Runnable {
    private final MainActivity mActivity;

    public GCDRunnable(MainActivity mainActivity) {
        mActivity = mainActivity;
    }

    public void run() {
        final String threadString =
            " with thread id " + Thread.currentThread();
        mActivity.println("Entering run()" + threadString);
        ...
    }
}
```

Use the field(s) within the thread's run() hook method to customize its behavior
Parameters passed to run() can be supplied via one of two other means, e.g.

- As parameters to a class constructor

```java
public class GCDRunnable extends Random implements Runnable {
    private final MainActivity mActivity;

    public GCDRunnable(MainActivity mainActivity) {
        mActivity = mainActivity;
    }

    public void run() {
        final String threadString =
            " with thread id " + Thread.currentThread();
        mActivity.println("Entering run()" + threadString);
    }
}
```

```java
public class MainActivity ... { ... 
    public void runRunnable(View v) { ... 
        new Thread(new GDCRunnable(this));
    }
}
```

Pass the parameter(s) when the runnable or thread is created.
Passing Parameters to a Java Thread

- Parameters passed to `run()` can be supplied via one of two other means, e.g.
  - As parameters to a class constructor
  - As parameters to “setter” methods

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

Passing Parameters to a Java Thread

- Parameters passed to run() can be supplied via one of two other means, e.g.
  - As parameters to a class constructor
  - As parameters to “setter” methods

```java
public class GCDThread extends Thread {
    private MainActivity mMainActivity; private Random mRandom;
    ...
```

Define field(s) to store parameters passed to a runnable or thread object
Passing Parameters to a Java Thread

• Parameters passed to run() can be supplied via one of two other means, e.g.
  • As parameters to a class constructor
  • As parameters to "setter" methods

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

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

    public GCDThread setRandom(Random random) {
        mRandom = random; return this;
    }

    ...
```

Define setter methods that update field(s)
Passing Parameters to a Java Thread

• Parameters passed to run() can be supplied via one of two other means, e.g.
  • As parameters to a class constructor
  • As parameters to “setter” methods

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

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

    public GCDThread setRandom(Random random)
    { mRandom = random; return this; }
    ...
```

See en.wikipedia.org/wiki/Fluent_interface

Note use of “fluent interfaces”
Passing Parameters to a Java Thread

- Parameters passed to run() can be supplied via one of two other means, e.g.
  - As parameters to a class constructor
  - As parameters to “setter” methods

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

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

    public GCDThread setRandom(Random random) {
        mRandom = random; return this;
    }

    public void run() {
        mActivity.println("Entering run()" + threadString);
        int number1 = mRandom.nextInt();
        int number2 = mRandom.nextInt();
    }
}
```

Use the fields within the thread’s run() hook method to customize its behavior.
• Parameters passed to run() can be supplied via one of two other means, e.g.
  • As parameters to a class constructor
  • As parameters to “setter” methods

    public class GCDThread extends Thread {
        ...
    }

    public class MainActivity ... { ...
        public void runThread(View v) { ...
            Thread thread =
                new GCDThread()
                .setActivity(this)
                .setRandom(new Random());
        ...
    }

    Use the fluent interface to pass parameter(s) when the runnable or thread is created
End of Overview of Java Threads (Part 1)