Java Platform Threads
vs. Virtual Threads (Part 2)

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 the differences between Java platform & virtual threads
• Be aware of how to create Java platform & virtual threads
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 the differences between Java platform & virtual threads
  • Be aware of how to create Java platform & virtual threads
• Recognize virtual Thread best practices
Creating Java Platform
Threads vs. Virtual Threads
Creating Java Platform Threads vs. Virtual Threads

• Java platform threads can be created in two different ways
Creating Java Platform Threads vs. Virtual Threads

• Java platform threads can be created in two different ways
• The traditional way

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

Thread gcdThread = new GCDThread();
gcdThread.start();
```

Create & start a thread using GCDThread, which is a named subclass of Thread
Creating Java Platform Threads vs. Virtual Threads

- Java platform threads can be created in two different ways
- The traditional way

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

Thread gcdThread = new GCDThread();
gcdThread.start();
```

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

Runnable gcdRunnable = new GCDRunnable();
new Thread(gcdRunnable).start();
```

See [en.wikipedia.org/wiki/Thread_(computing)#User_threads](en.wikipedia.org/wiki/Thread_(computing)#User_threads)
Creating Java Platform Threads vs. Virtual Threads

- Java platform threads can be created in two different ways
  - The traditional way

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

Thread gcdThread = new GCDThread();
gcdThread.start();

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

Runnable gcdRunnable = new GCDRunnable();
new Thread(gcdRunnable).start();
```

Java threads are relatively “heavyweight”
Java platform threads can be created in two different ways:

- The traditional way
- The Java 19 way

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

Runnable gcdRunnable = new GCDRunnable();
new Thread(gcdRunnable).start();
```

A familiar way to create & start a Java platform thread so it executes `gcdRunnable`

By default, a traditional Java Thread is a platform thread!
Creating Java Platform Threads vs. Virtual Threads

- Java platform threads can be created in two different ways
  - The traditional way
  - The Java 19 way

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

Runnable gcdRunnable = new GCDRunnable();
Thread.ofPlatform().start(gcdRunnable);
```

A more flexible way to create & start a platform thread so it executes gcdRunnable

See [docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#ofPlatform()](http://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#ofPlatform())
Creating Java Platform Threads vs. Virtual Threads

Java platform threads can be created in two different ways:

- The traditional way
- The Java 19 way

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

Runnable gcdRunnable = new GCDRunnable();

Thread thread = Thread.ofPlatform().unstarted(gcdRunnable);
...
thread.start();
```

Create an “unstarted” platform thread & then start it so it executes gcdRunnable
Creating Java Platform Threads vs. Virtual Threads

- Java platform threads can be created in two different ways
  - The traditional way
  - The Java 19 way

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

Runnable gcdRunnable = new GCDRunnable();

Thread thread = Thread.ofPlatform()
    .unstarted(gcdRunnable);
...
thread.start();
```

Java platform threads are also relatively “heavyweight”
Creating Java Platform Threads vs. Virtual Threads

- Java virtual threads can also be created in Java 19

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

Runnable gcdRunnable = new GCDRunnable();

Thread.startVirtualThread(gcdRunnable);
```

A concise way to create & start a Java virtual thread so it executes `gcdRunnable`

See [docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#startVirtualThread](docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#startVirtualThread)
• Java virtual threads can also be created in Java 19

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

Runnable gcdRunnable = new GCDRunnable();

Thread.ofVirtual().start(gcdRunnable);
```

A more flexible way to create & start a virtual thread so it executes gcdRunnable

See [docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#ofVirtual()](http://docs.oracle.com/en/java/javase/19/docs/api/java.base/java/lang/Thread.html#ofVirtual())
Java virtual threads can also be created in Java 19.

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

Runnable gcdRunnable = new GCDRunnable();

Thread thread = Thread.ofVirtual().unstarted(gcdRunnable);
...
thread.start();
```

Create an “unstarted” virtual thread & then start it so it executes gcdRunnable.
Java virtual threads can also be created in Java 19

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

Runnable gcdRunnable = new GCDRunnable();

Thread thread = Thread.ofVirtual()
    .unstarted(gcdRunnable);
... thread.start();
```

Java virtual threads are relatively “lightweight”
Virtual Thread
Best Practices
Virtual Thread Best Practices

- Follow certain “best practices” when using Java virtual threads

See howtodoinjava.com/java/multi-threading/virtual-threads/#5-best-practices
Virtual Thread Best Practices

- Follow certain “best practices” when using Java virtual threads
- Do not pool virtual threads!
  - Creating virtual threads is inexpensive, so there is never a need to pool them

Virtual Thread Best Practices

• Follow certain “best practices” when using Java virtual threads
  • Do not pool virtual threads!
  • Avoid using thread-local variables
  • If an app uses ThreadLocal & creates 1 million virtual threads then 1 million Thread Local instances are created!

See howtodoinjava.com/java/multi-threading/virtual-threads/#52-avoid-using-thread-local-variables
Virtual Thread Best Practices

- Follow certain “best practices” when using Java virtual threads
  - Do not pool virtual threads!
  - Avoid using thread-local variables
  - Use ReentrantLock instead of synchronized blocks
  - Synchronized blocks “pin” a virtual thread to a platform thread...

```java
public synchronized void m() {
    try {
        // ... access resource
    } finally {
        //
    }
}
...

private final ReentrantLock lock = new ReentrantLock();

public void m() {
    lock.lock();
    try {
        // ... access resource
    } finally {
        lock.unlock();
    }
}
```

End of Java Platform
Threads vs. Virtual Threads (Part 2)