Applying Java Platform Threads & Virtual Threads: Case Study ex1

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 program Java platform & virtual threads

```java
Thread makeThread
    (Runnable runnable,
     boolean virtual) {
    if (virtual)
        return Thread.ofVirtual()
            .unstarted(runnable);
    else
        return Thread.ofPlatform()
            .unstarted(runnable);
}
Applying Java Platform
Threads & Virtual Threads
// Create a List of many Thread objects.
List<Thread> threads = IntStream
    .rangeClosed(1, Options.instance().numberOfElements())
    .mapToObj(i ->
        // Make a new Thread (either virtual or
        // platform) for each int and give it a large
        // random number to check for primality.
        makeThread(makeRunnable(integer: sRANDOM_INTEGERS.get(i - 1),
            virtual))
            // Trigger intermediate processing and collect the Thread
            // objects into a List.
        .toList();

    // Start all the Thread objects.
    threads.forEach(action: Thread::start);

    // Join all the Thread objects (barrier synchronizer).
    threads.forEach(action: rethrowConsumer(consumer: Thread::join));
End of Applying Java Platform Threads & Virtual Threads