Introduction to Java Threads

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
Introduction to Java Threads
Introduction to Java Threads

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

A Java thread is a unit of computation that runs in the context of a process.

See [en.wikipedia.org/wiki/Thread_(computing)](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 in Java

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

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

A Java thread runs on one or more cores during its lifetime

See [en.wikipedia.org/wiki/Multi-core_processor](en.wikipedia.org/wiki/Multi-core_processor)
Introduction to Java Threads

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

Java enables multiple threads to run in multiple processes atop multiple cores.

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.

See www.javatpoint.com/inter-thread-communication-example & web.mit.edu/6.005/www/fa14/classes/20-queues-locks/message-passing
Introduction to Java Threads

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

Java threads in different processes communicate via shared memory or inter-process communication (IPC) mechanisms

See developer.android.com/guide/components/aidl
Threads are the most basic way of obtaining concurrency in Java.

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

See [en.wikipedia.org/wiki/Thread_(computing)#Processes.2C_kernel_threads.2C_user_threads.2C_and_fibers](en.wikipedia.org/wiki/Thread_(computing)#Processes.2C_kernel_threads.2C_user_threads.2C_and_fibers)
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](en.wikipedia.org/wiki/Thread_(computing)#Processes.2C_kernel_threads.2C_user_threads.2C_and_fibers)
End of Introduction to Java Threads