Overview of Concurrency in Java

Douglas C. Schmidt
d.schmidt@vanderbilt.edu
www.dre.vanderbilt.edu/~schmidt

Professor of Computer Science
Institute for Software Integrated Systems
Vanderbilt University
Nashville, Tennessee, USA
Learning Objectives in this Lesson

• Understand the meaning of key concurrency concepts
Learning Objectives in this Lesson

- Understand the meaning of key concurrency concepts
- Recognize how these concepts are supported in Java
An Overview of Concurrency
An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously.

See [en.wikipedia.org/wiki/Concurrency_(computer_science)](en.wikipedia.org/wiki/Concurrency_(computer_science))
An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously

```java
new Thread() ->
    someComputations().
    start();
```

*A thread is a unit of execution for instruction streams that can run concurrently on processor cores*

See [docs.oracle.com/javase/tutorial/essential/concurrency/threads.html](https://docs.oracle.com/javase/tutorial/essential/concurrency/threads.html)
An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously.
- Often used to offload work from the user interface (UI) thread to background thread(s).

See developer.android.com/topic/performance/threads.html
An Overview of Concurrency

• Concurrency is a form of computing where threads can run simultaneously

• Often used to offload work from the user interface (UI) thread to background thread(s), e.g.
  • Background thread(s) can block

See developer.android.com/training/multiple-threads/communicate-ui.html
An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously.
- Often used to offload work from the user interface (UI) thread to background thread(s), e.g.
  - Background thread(s) can block.
  - The UI thread does not block.

See [developer.android.com/training/multiple-threads/communicate-ui.html](https://developer.android.com/training/multiple-threads/communicate-ui.html)
An Overview of Concurrency in Java
An Overview of Concurrency in Java

• A Java thread is an object

```java
public class Thread
extends Object
implements Runnable

A thread is a thread of execution in a program. The Java Virtual Machine allows an application to have multiple threads of execution running concurrently.

Every thread has a priority. Threads with higher priority are executed in preference to threads with lower priority. Each thread may or may not also be marked as a daemon. When code running in some thread creates a new Thread object, the new thread has its priority initially set equal to the priority of the creating thread, and is a daemon thread if and only if the creating thread is a daemon.
```

See docs.oracle.com/javase/8/docs/api/java/lang/Thread.html
A Java thread is an object, e.g.

- It contains methods & fields

See blog.jamesdbloom.com/JVMInternals.html

Each Java thread has its own stack, registers, etc.
An Overview of Concurrency in Java

• A Java thread is an object, e.g.
  • It contains methods & fields
  • It can also be in one of various “states”

See docs.oracle.com/javase/8/docs/api/java/lang/Thread.State.html
An Overview of Concurrency in Java

• Concurrent Java threads interact via shared objects and/or message passing

See docs.oracle.com/javase/8/docs/api/?java/util/concurrent/package-summary.html
An Overview of Concurrency in Java

- Concurrent Java threads interact via shared objects and/or message passing

- **Shared objects**
  - Synchronize concurrent operations on objects to ensure certain properties

See [en.wikipedia.org/wiki/Synchronization_(computer_science)](en.wikipedia.org/wiki/Synchronization_(computer_science))
An Overview of Concurrency in Java

- Concurrent Java threads interact via shared objects and/or message passing

- **Shared objects**
  - Synchronize concurrent operations on objects to ensure certain properties, e.g.
    - *Mutual exclusion*
      - Interactions between threads won’t corrupt shared mutable data

See [en.wikipedia.org/wiki/Monitor_(synchronization)#Mutual_exclusion](en.wikipedia.org/wiki/Monitor_(synchronization)#Mutual_exclusion)
An Overview of Concurrency in Java

- Concurrent Java threads interact via shared objects and/or message passing

  **Shared objects**
  - Synchronize concurrent operations on objects to ensure certain properties, e.g.
    - *Mutual exclusion*
    - *Coordination*
      - Operations occur in the right order, at the right time, & under the right conditions

See [en.wikipedia.org/wiki/Monitor_(synchronization)#Condition_variables](en.wikipedia.org/wiki/Monitor_(synchronization)#Condition_variables)
An Overview of Concurrency in Java

• Concurrent Java threads interact via shared objects and/or message passing

• **Shared objects**
  • Synchronize concurrent operations on objects to ensure certain properties
  • Examples of Java synchronizers:
    • Synchronized statements/methods
    • Reentrant locks & intrinsic locks
    • Atomic operations
    • Semaphores
    • Condition objects
    • “Compare-and-swap” (CAS) operations in sun.misc.unsafe

See [dzone.com/articles/the-java-synchronizers](dzone.com/articles/the-java-synchronizers)
An Overview of Concurrency in Java

• Concurrent Java threads interact via shared objects and/or message passing

  • Shared objects
  • Message passing
    • Send message(s) from producer thread(s) to consumer thread(s) via a thread-safe queue

See en.wikipedia.org/wiki/Message_passing
An Overview of Concurrency in Java

- Concurrent Java threads interact via shared objects and/or message passing
  
  **Shared objects**
  
  **Message passing**
  - Send message(s) from producer thread(s) to consumer thread(s) via a thread-safe queue
  - Examples of Java thread-safe queues
    - Array & linked blocking queues
    - Priority blocking queue
    - Synchronous queue
    - Concurrent linked queue

See [docs.oracle.com/javase/tutorial/collections/implementations/queue.html](docs.oracle.com/javase/tutorial/collections/implementations/queue.html)
An Overview of Concurrency in Java

• Key goals of using Java shared objects and/or message passing are to share resources safely & avoid hazards

See en.wikipedia.org/wiki/Thread_safety
An Overview of Concurrency in Java

- Key goals of using Java shared objects and/or message passing are to share resources safely & avoid hazards, e.g.
  - Race conditions
    - Race conditions occur when a program depends upon the sequence or timing of threads for it to operate properly

See [en.wikipedia.org/wiki/Race_condition#Software](en.wikipedia.org/wiki/Race_condition#Software)
An Overview of Concurrency in Java

- Key goals of using Java shared objects and/or message passing are to share resources safely & avoid hazards, e.g.
- Race conditions
  - Race conditions occur when a program depends upon the sequence or timing of threads for it to operate properly

This test program induces race conditions due to lack of synchronization between producer & consumer threads accessing a bounded queue

See github.com/douglascraigschmidt/LiveLessons/tree/master/BuggyQueue
An Overview of Concurrency in Java

• Key goals of using Java shared objects and/or message passing are to share resources safely & avoid hazards, e.g.
  • Race conditions
  • Memory inconsistencies
    • These errors occur when different threads have inconsistent views of what should be the same data

An Overview of Concurrency in Java

- Key goals of using Java shared objects and/or message passing are to share resources safely & avoid hazards, e.g.
  - Race conditions
  - Memory inconsistencies
  - Deadlocks
    - Occur when 2+ competing threads are waiting for the other(s) to finish, & thus none ever do

See en.wikipedia.org/wiki/Deadlock
End of Overview of Concurrency in Java