Overview of Concurrent Programming

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 Part of the Lesson

• Understand the meaning of key concurrent programming concepts
An Overview of Sequential Programming
An Overview of Sequential Programming

- Sequential programming is a form of computing that executes the same sequence of instructions & always produces the same results.

See [en.wikipedia.org/wiki/Sequential_algorithm](en.wikipedia.org/wiki/Sequential_algorithm)
An Overview of Sequential Programming

- Sequential programming is a form of computing that executes the same sequence of instructions & always produces the same results
- i.e., execution is deterministic

See screenprism.com/insights/article/what-is-the-ludovico-technique-and-how-does-it-work
An Overview of Sequential Programming

• Sequential programs have two characteristics
An Overview of Sequential Programming

- Sequential programs have two characteristics:
  - The textual order of statements specifies their order of execution

```java
public E get(int index) {
    rangeCheck(index);
    return elementData(index);
}
```

*e.g., the rangeCheck() method must be called before the elementData() method*
An Overview of Sequential Programming

• Sequential programs have two characteristics:
  • The textual order of statements specifies their order of execution
  • Successive statements must execute without any temporal overlap
An Overview of Concurrent Programming
An Overview of Concurrent Programming

- Concurrent programming is a form of computing where threads can run simultaneously

See en.wikipedia.org/wiki/Concurrency_(computer_science)
An Overview of Concurrent Programming

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

```java
for (int i = 0; i < sMax; i++)
    new Thread(() ->
        someComputation()).start();
```

A thread is a unit of execution for instruction streams that can run concurrently on 1+ processor cores.

See [docs.oracle.com/javase/tutorial/essential/concurrency/threads.html](docs.oracle.com/javase/tutorial/essential/concurrency/threads.html)
An Overview of Concurrent Programming

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

```java
for (int i = 0; i < sMax; i++)
    new Thread(() ->
        someComputation()).
    start();
```

*Threads may be multiplexed over one core, though this is increasingly rare.*

An Overview of Concurrent Programming

- Different executions of a concurrent program may produce different instruction orderings

See en.wikipedia.org/wiki/Nondeterministic_algorithm
Different executions of a concurrent program may produce different instruction orderings:

The textual order of the source code doesn’t define the order of execution:

- new Thread() ->
  - computationA().
  - start();

- new Thread() ->
  - computationB().
  - start();

- new Thread() ->
  - computationC().
  - start();

*computationA(), computationB(), & computationC() can run in any order once they start their execution.*
An Overview of Concurrent Programming

- Different executions of a concurrent program may produce different instruction orderings:
  - The textual order of the source code doesn’t define the order of execution
  - Operations are permitted to overlap in time
An Overview of Concurrent Programming

- Concurrent programming is 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 Concurrent Programming

- Concurrent programming is 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
Concurrent programming is 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
Concurrent programming is 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
- Any mutable state shared between these threads must be protected to avoid concurrency hazards

See upcoming lesson on "Overview of Concurrency in Java"
End of Overview of Concurrent Programming