Background on Java Concurrency & Parallelism

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

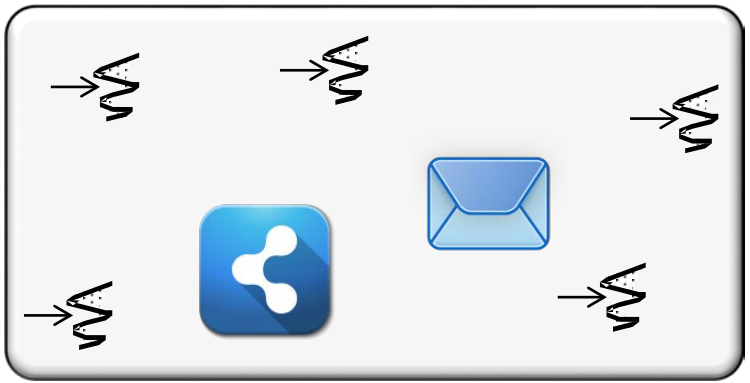
Professor of Computer Science

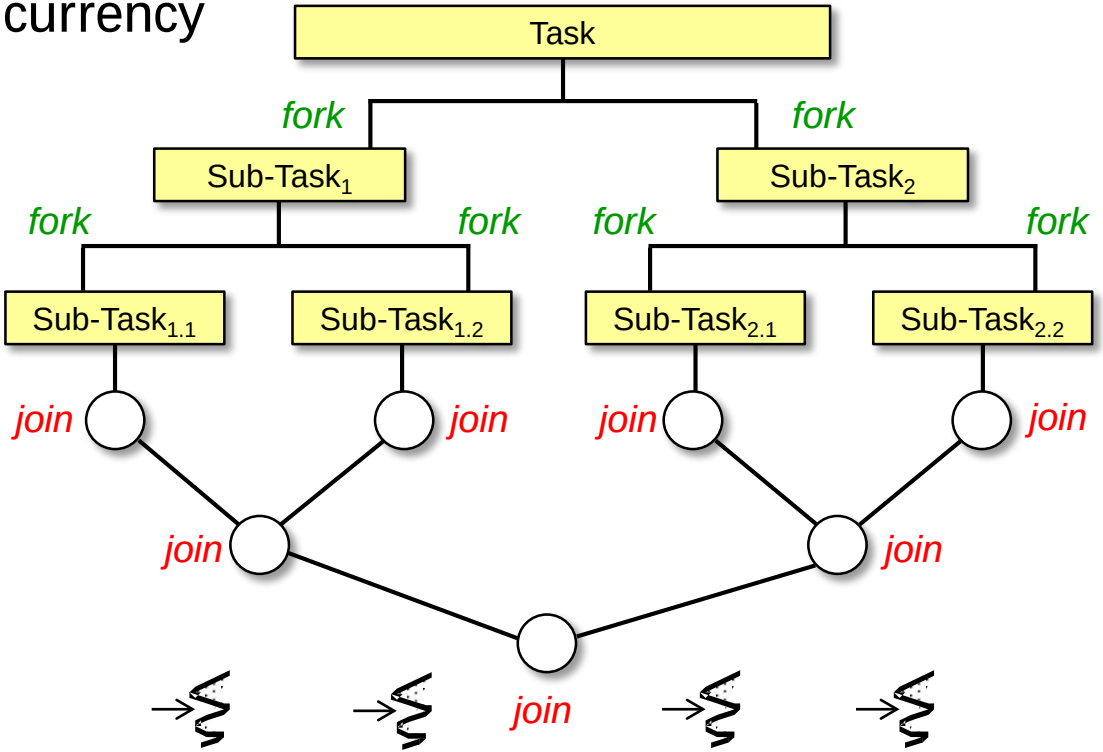
Institute for Software Integrated Systems

Vanderbilt University
Nashville, Tennessee, USA

Learning Objectives in this Lesson

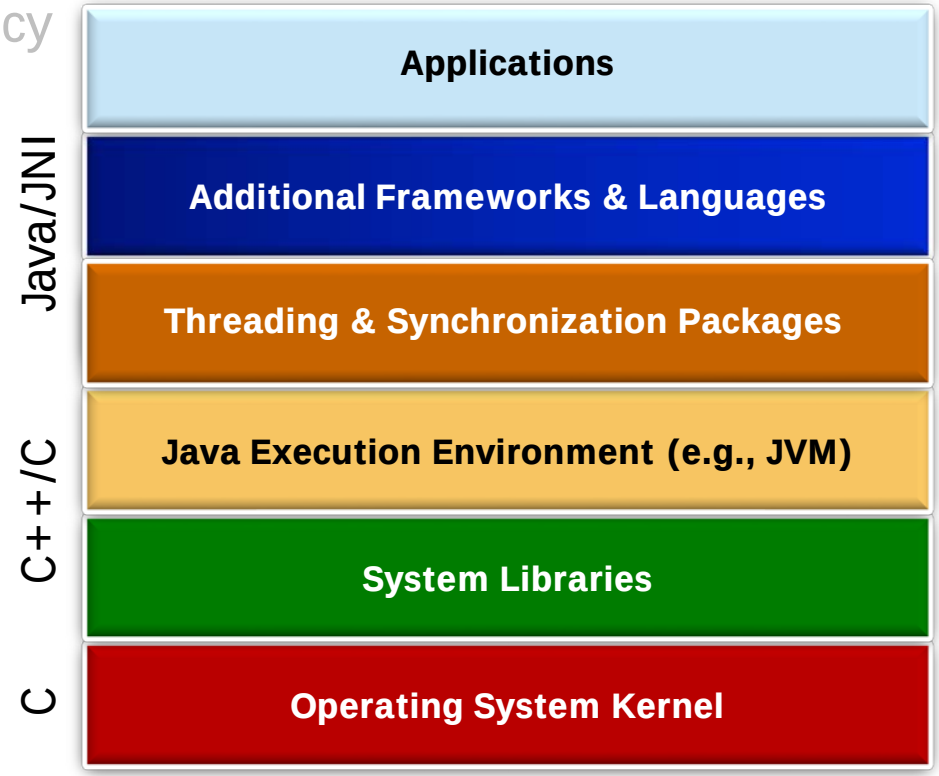
- Understand the meaning of concurrency & parallelism





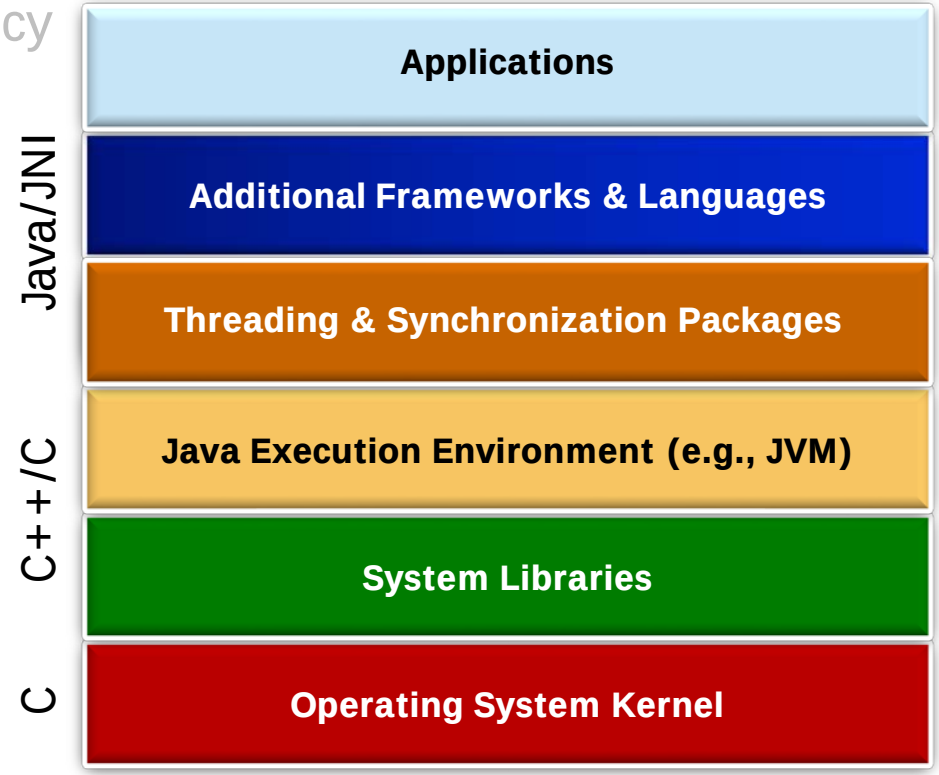
Learning Objectives in this Lesson

- Understand the meaning of concurrency & parallelism
- Know the history of Java concurrency & parallelism



Learning Objectives in this Lesson

- Understand the meaning of concurrency & parallelism
- Know the history of Java concurrency & parallelism

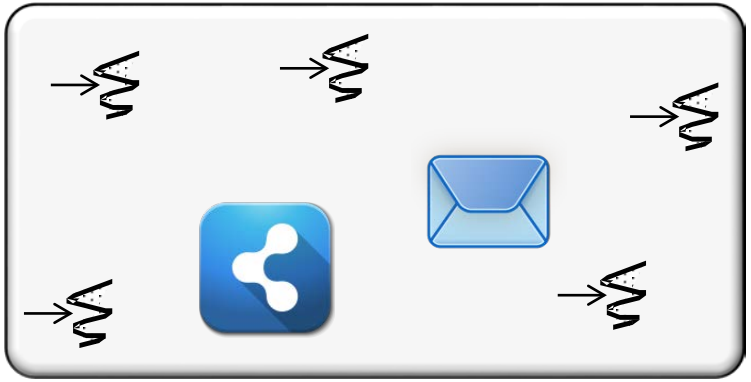


Hopefully, you'll already know much of this!!!

An Overview of Concurrency

An Overview of Concurrency

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

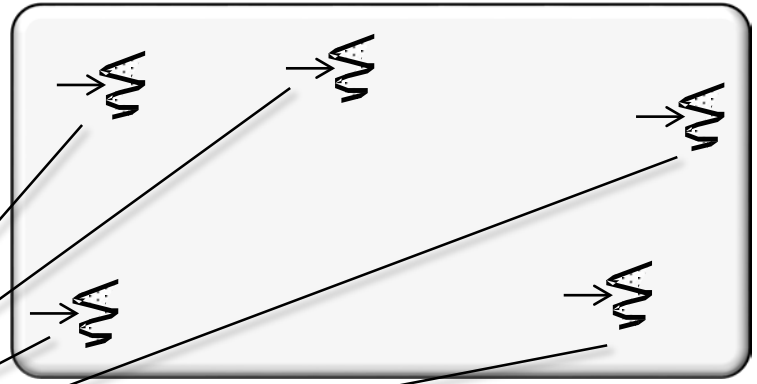


See en.wikipedia.org/wiki/Concurrency_(computer_science)

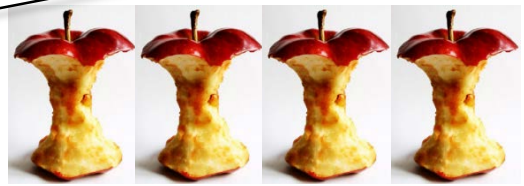
An Overview of Concurrency

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

```
new Thread(() ->
           someComputations());
```



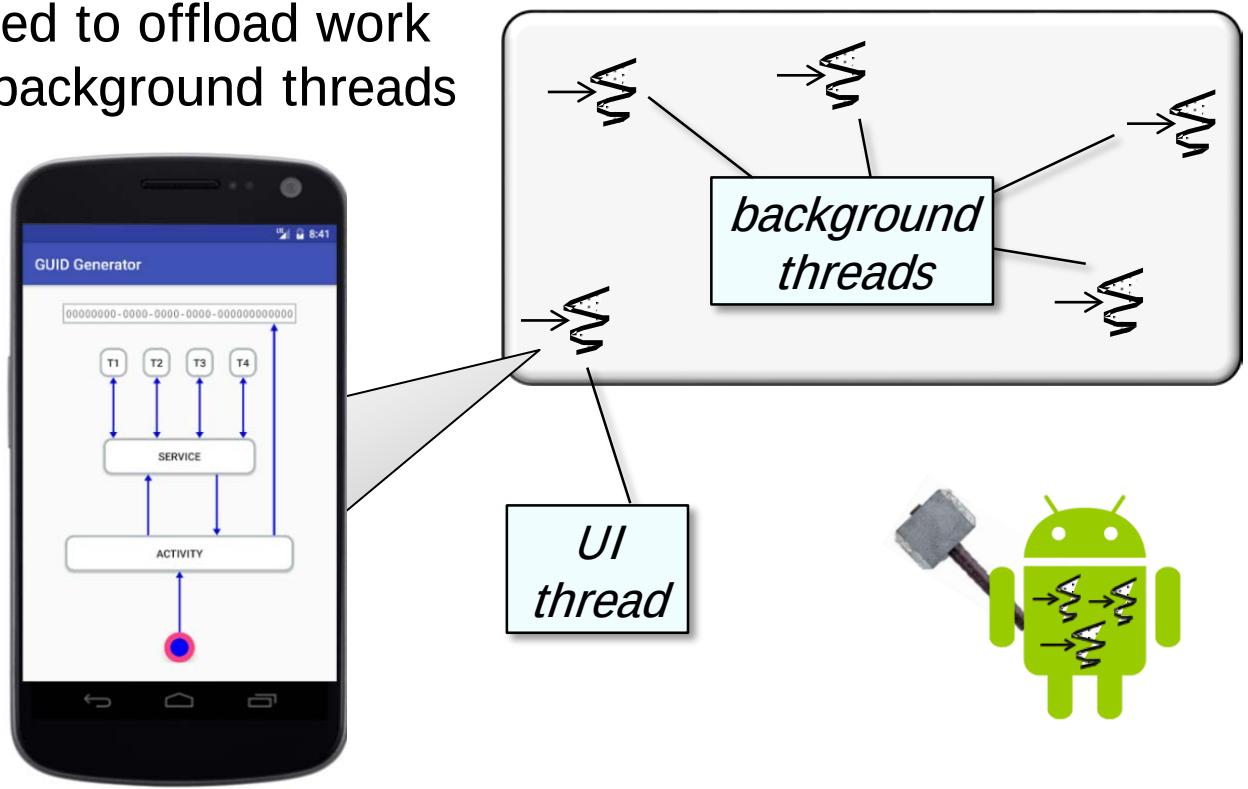
A Java threads are units of execution for instruction streams that can run concurrently on processor cores



See docs.oracle.com/javase/tutorial/essential/concurrency/threads.html

An Overview of Concurrency

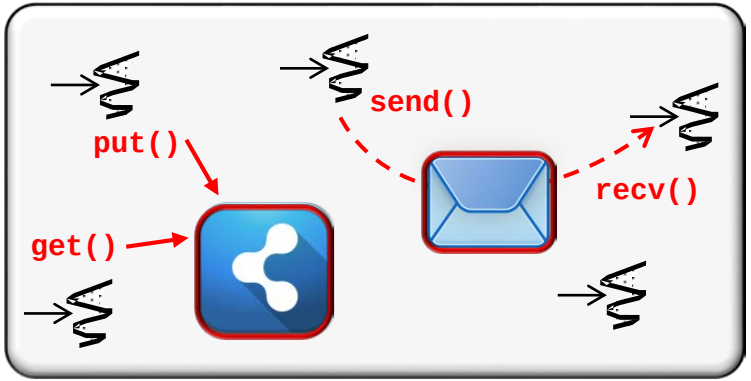
- Concurrency is a form of computing where threads can run simultaneously
  - Concurrency often used to offload work from main thread to background threads



See developer.android.com/topic/performance/threads.html

An Overview of Concurrency

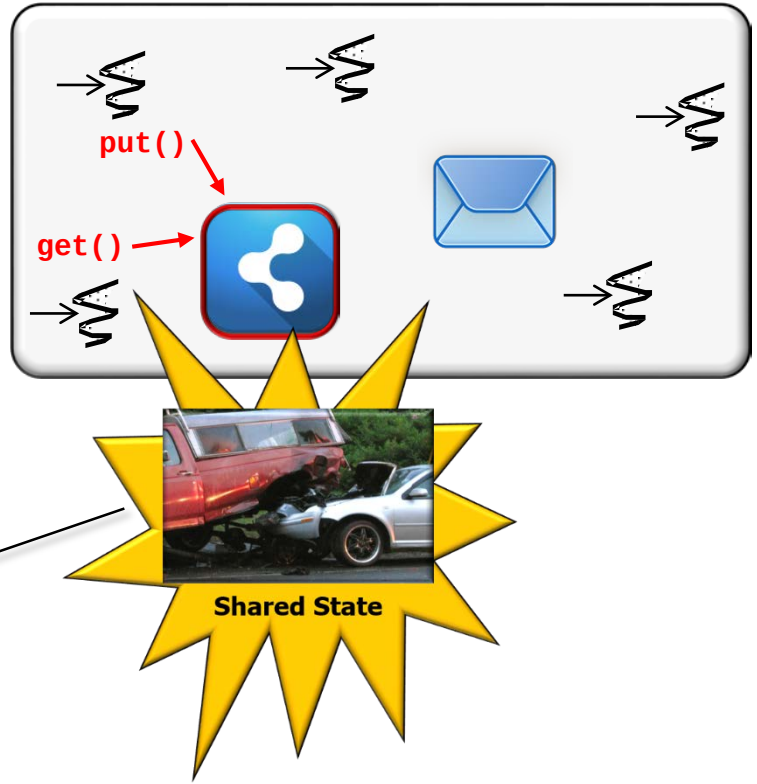
- Concurrency is a form of computing where threads can run simultaneously
  - Concurrency often used to offload work from main thread to background threads
  - Java threads interact with each other via shared objects and/or message passing



See docs.oracle.com/javase/8/docs/api/?java/util/concurrent/package-summary.html

An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously
  - Concurrency often used to offload work from main thread to background threads
  - Java threads interact with each other via shared objects and/or message passing
  - Key goal is to share resources safely & efficiently to avoid race conditions

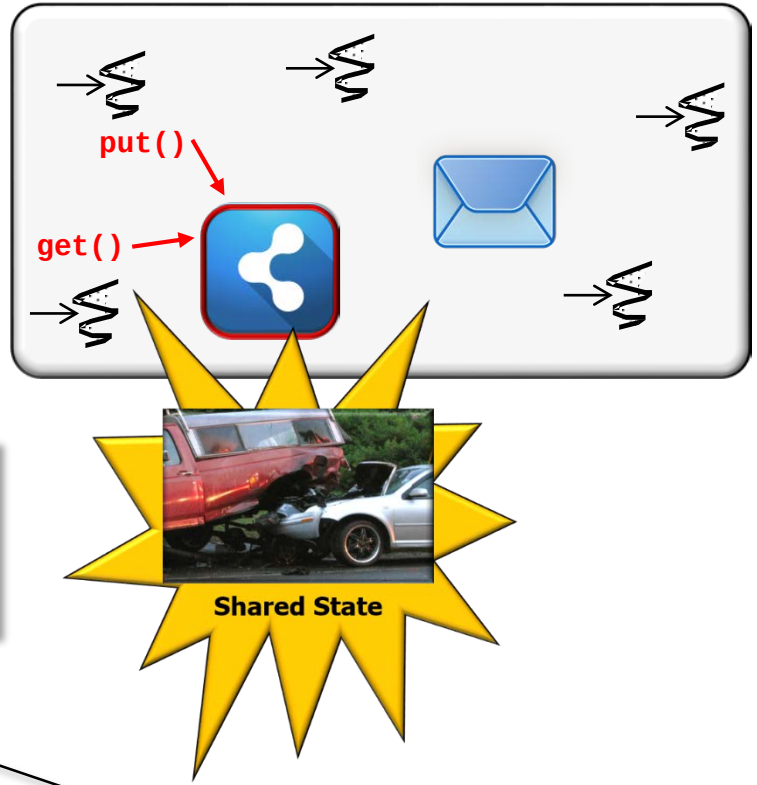


Race conditions occur when a program depends upon the sequence or timing of threads for it to operate properly

See en.wikipedia.org/wiki/Race_condition#Software

An Overview of Concurrency

- Concurrency is a form of computing where threads can run simultaneously
  - Concurrency often used to offload work from main thread to background threads
  - Java threads interact with each other via shared objects and/or message passing
  - Key goal is to share resources safely & efficiently to avoid race conditions



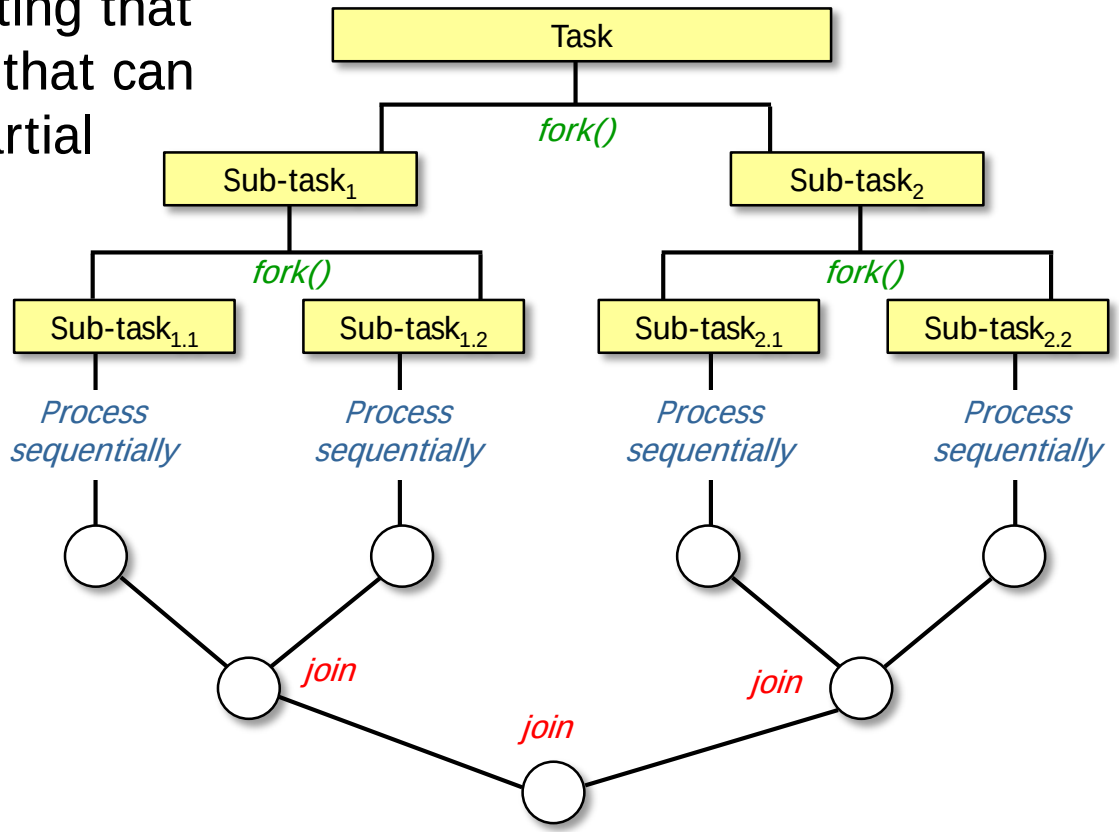
This test program induces race conditions due to lack of synchronization between producer & consumer threads accessing a bounded queue

See github.com/douglascraigschmidt/LiveLessons/tree/master/BuggyQueue

An Overview of Parallelism

An Overview of Parallelism

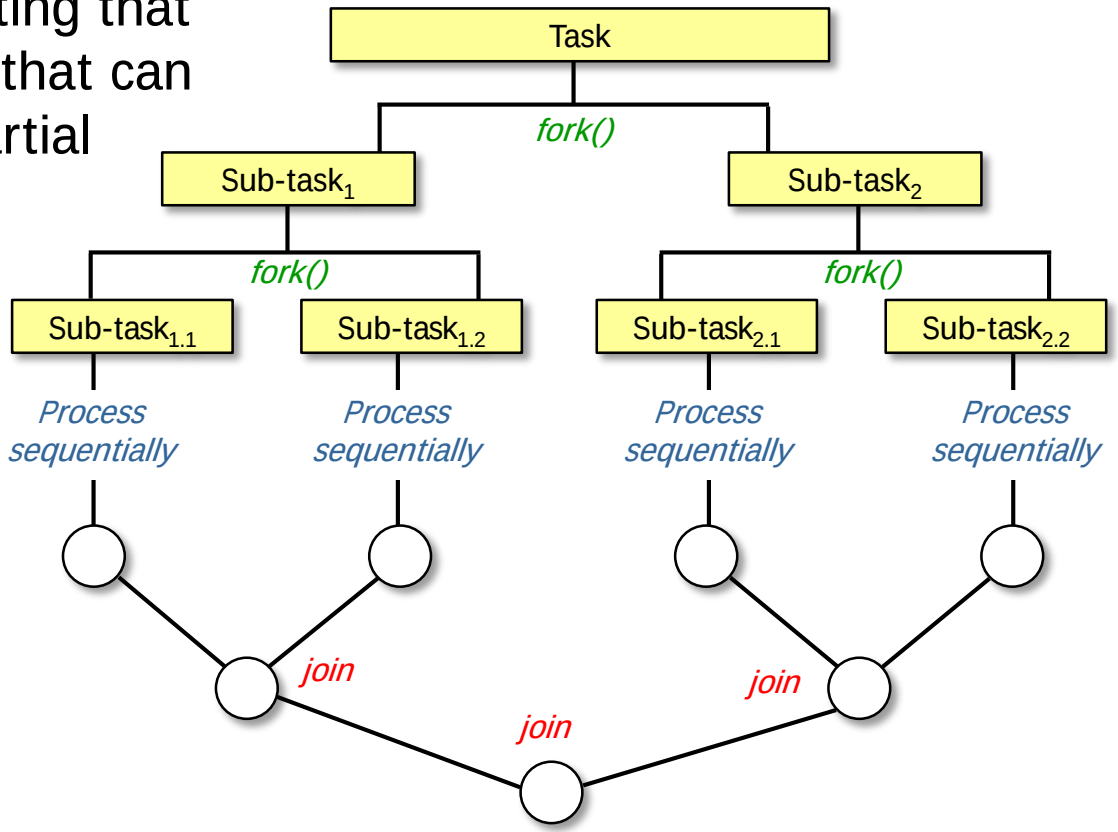
- Parallelism is a form of computing that partitions tasks into sub-tasks that can run independently & whose partial results are combined



See en.wikipedia.org/wiki/Parallel_computing

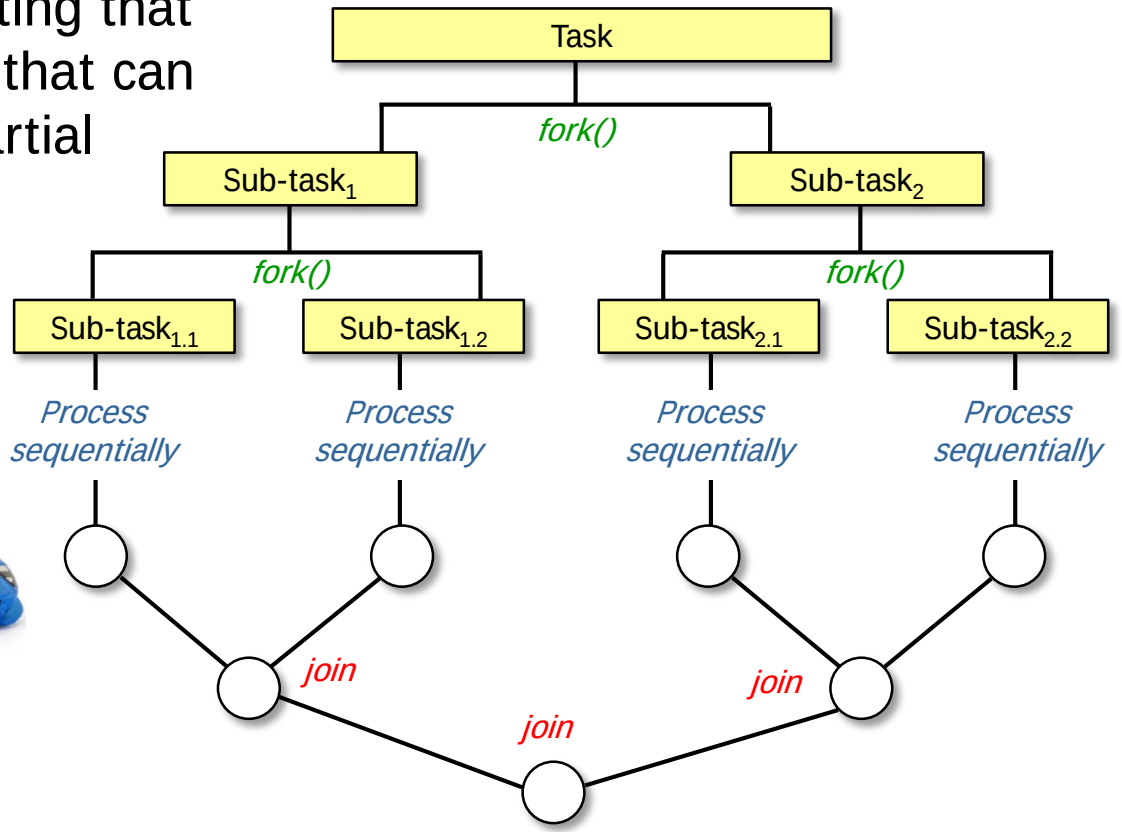
An Overview of Parallelism

- Parallelism is a form of computing that partitions tasks into sub-tasks that can run independently & whose partial results are combined
  - Key goal is to *efficiently* (1) partition tasks into sub-tasks & (2) combine results



An Overview of Parallelism

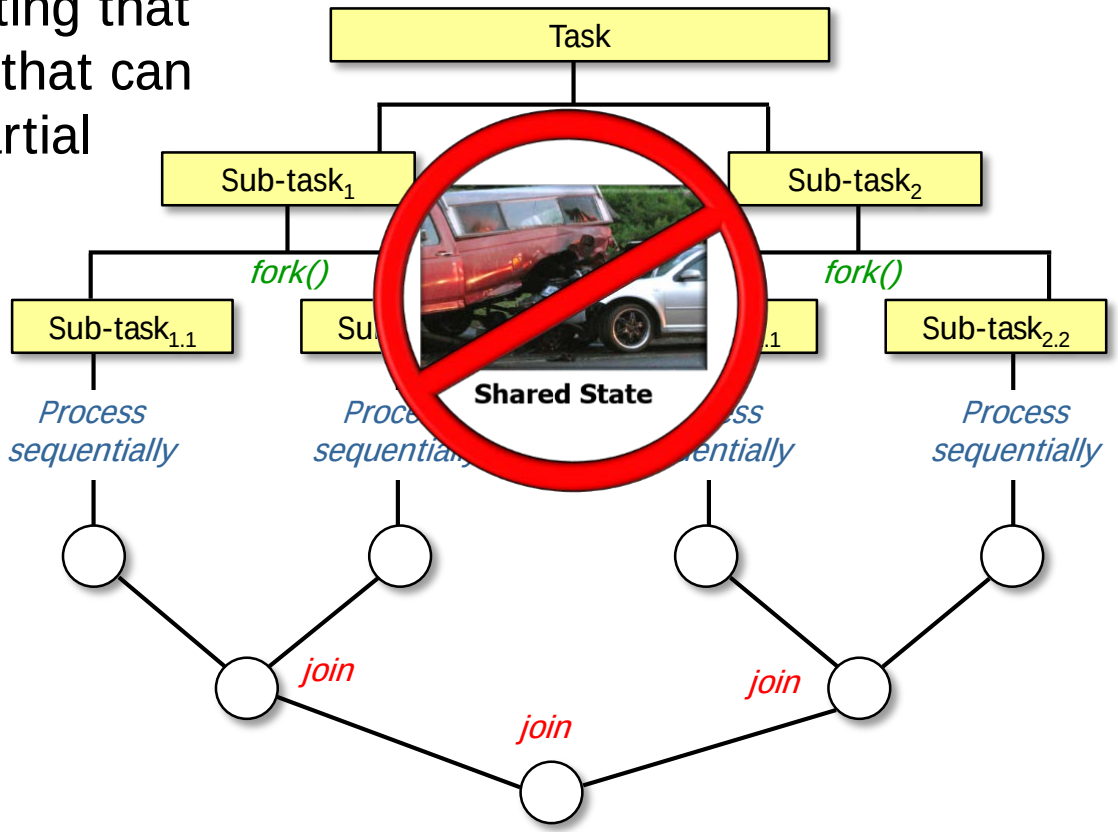
- Parallelism is a form of computing that partitions tasks into sub-tasks that can run independently & whose partial results are combined
  - Key goal is to *efficiently* (1) partition tasks into sub-tasks & (2) combine results



Parallelism is a performance optimization (e.g., throughput, scalability, & latency)

An Overview of Parallelism

- Parallelism is a form of computing that partitions tasks into sub-tasks that can run independently & whose partial results are combined
  - Key goal is to *efficiently* (1) partition tasks into sub-tasks & (2) combine results
  - Parallelism works best when there's no shared mutable state between threads



See henrikeichenhardt.blogspot.com/2013/06/why-shared-mutable-state-is-root-of-all.html

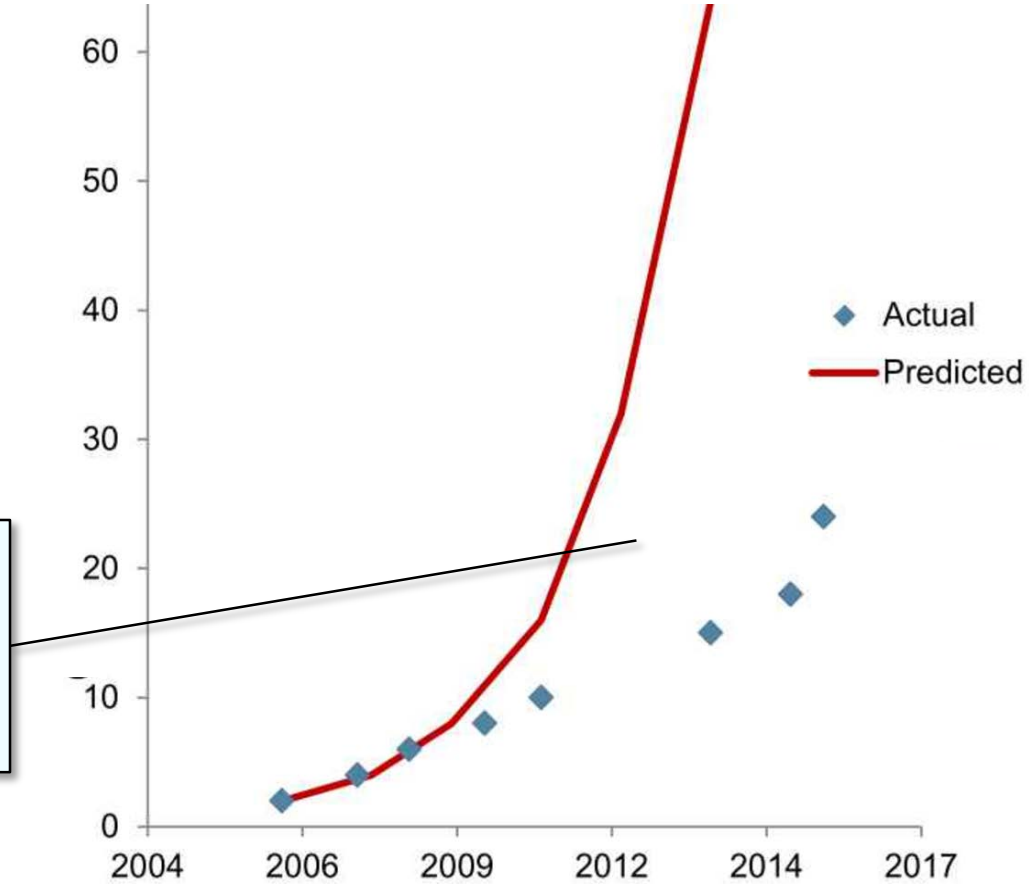
An Overview of Parallelism

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing

See www.youtube.com/watch?v=NsDE7E8sIdQ

An Overview of Parallelism

- Brian Goetz has an excellent talk about the evolution of Java from concurrent to parallel computing



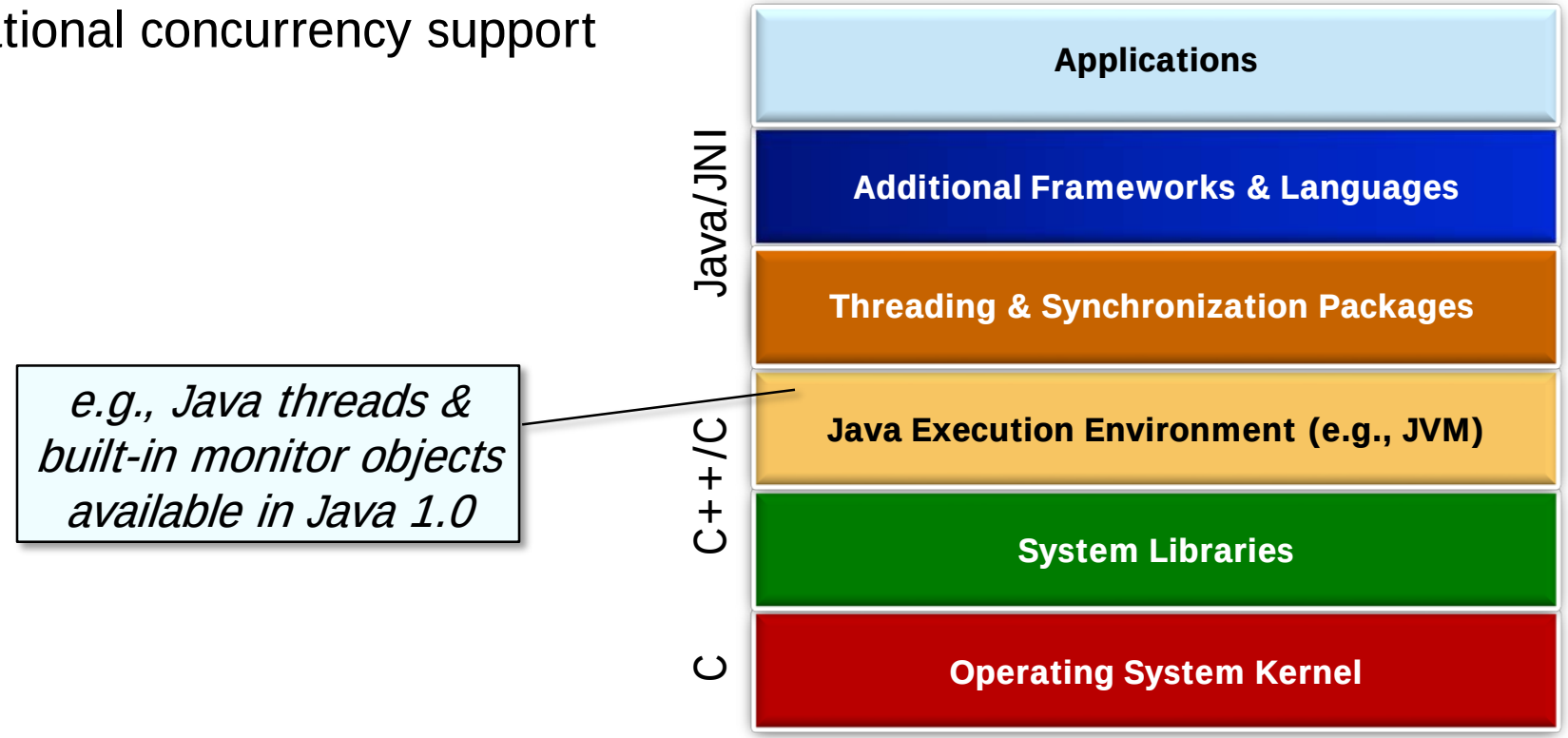
His talk emphasizes that Java 8 combines functional programming with fine-grained data parallelism to leverage many-core processors

See www.infoq.com/presentations/parallel-java-se-8

A Brief History of Concurrency & Parallelism in Java

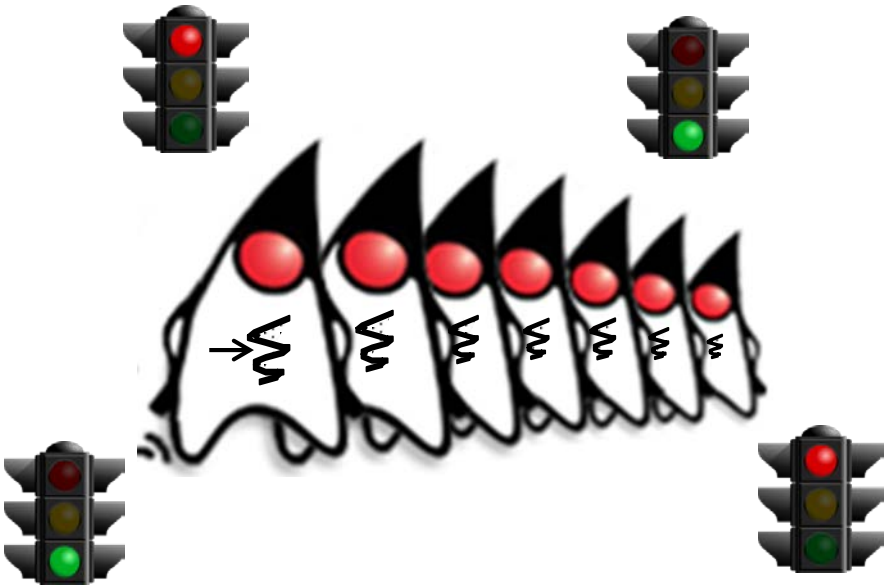
A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support



A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
  - Focus on basic multi-threading & synchronization primitives



See docs.oracle.com/javase/tutorial/essential/concurrency

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
  - Focus on basic multi-threading & synchronization primitives

Allow multiple threads to communicate via a bounded buffer

```
SimpleBlockingBoundedQueue<Integer>
  simpleQueue = new
    SimpleBlockingBoundedQueue<>();

Thread[] threads = new Thread[] {
  new Thread(new Producer<>
                  (simpleQueue)),
  new Thread(new Consumer<>
                  (simpleQueue))
};

for (Thread thread : threads)
  thread.start();

for (Thread thread : threads)
  thread.join();
```

See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
  - Focus on basic multi-threading & synchronization primitives

```
SimpleBlockingBoundedQueue<Integer>
  simpleQueue = new
    SimpleBlockingBoundedQueue<>();

Thread[] threads = new Thread[] {
  new Thread(new Producer<>
                  (simpleQueue)),
  new Thread(new Consumer<>
                  (simpleQueue))
};

for (Thread thread : threads)
  thread.start();

for (Thread thread : threads)
  thread.join();
```

Start & join these multiple threads

See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
  - Focus on basic multi-threading & synchronization primitives

```
class SimpleBlockingBoundedQueue
                                    <E> {
  public E take() ...{
    synchronized(this) {
      while (mList.isEmpty())
        wait();

      notifyAll();

      return mList.poll();
    }
  }
```

Built-in monitor object mutual exclusion & coordination primitives

See github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue

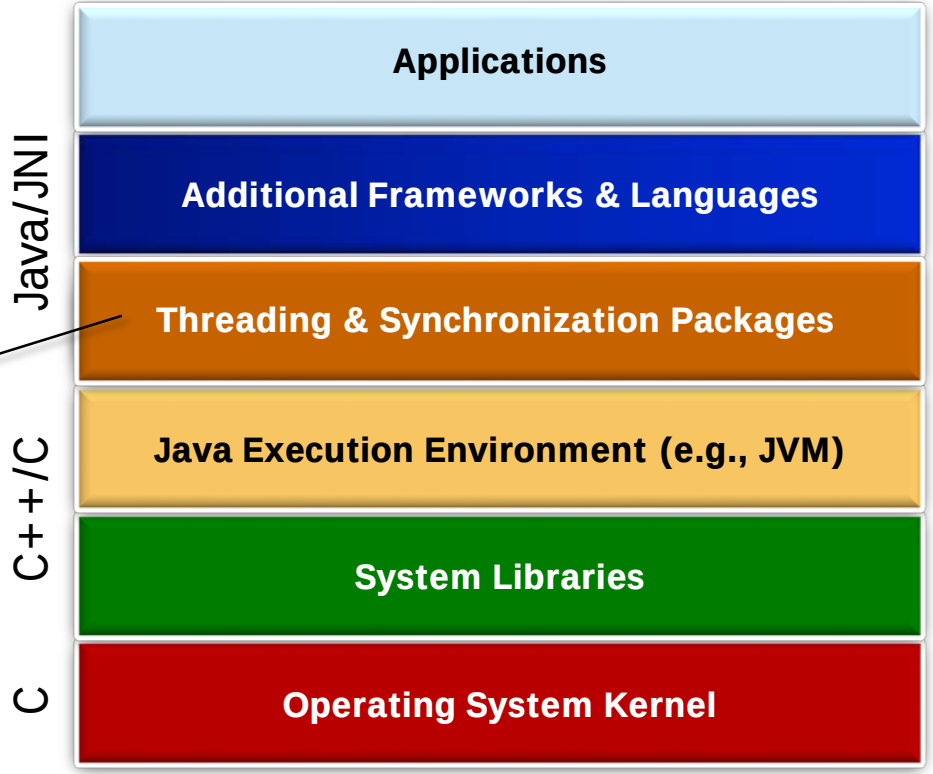
A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
  - Focus on basic multi-threading & synchronization primitives
  - Efficient, but low-level & very limited in capabilities

A Brief History of Concurrency & Parallelism in Java

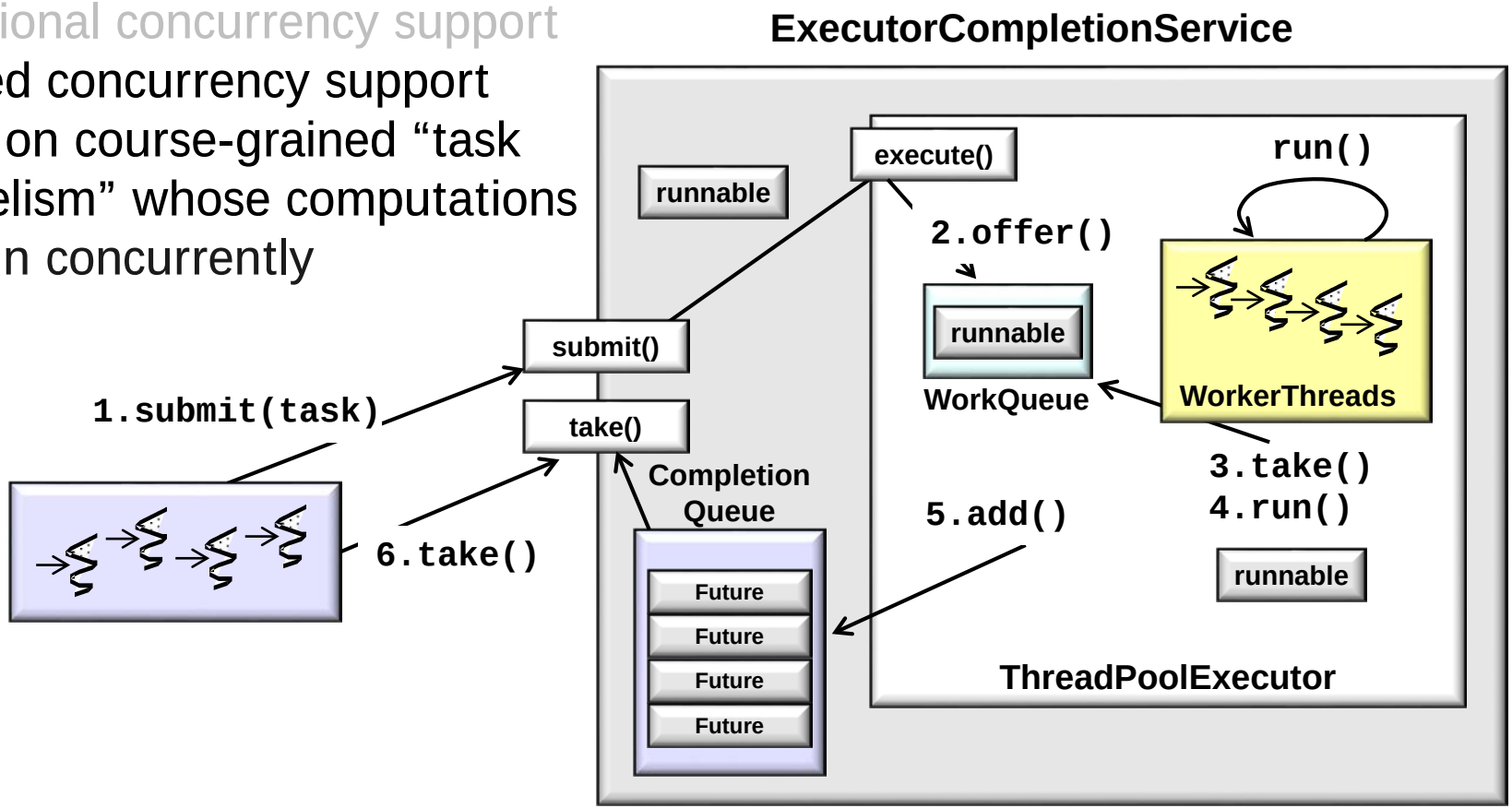
- Foundational concurrency support
- Advanced concurrency support

e.g., Java executor framework, synchronizers, blocking queues, atomics, & concurrent collections available in Java 1.5+



A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
  - Focus on course-grained “task parallelism” whose computations can run concurrently



See en.wikipedia.org/wiki/Task_parallelism

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
  - Focus on course-grained "task parallelism" whose computations can run concurrently

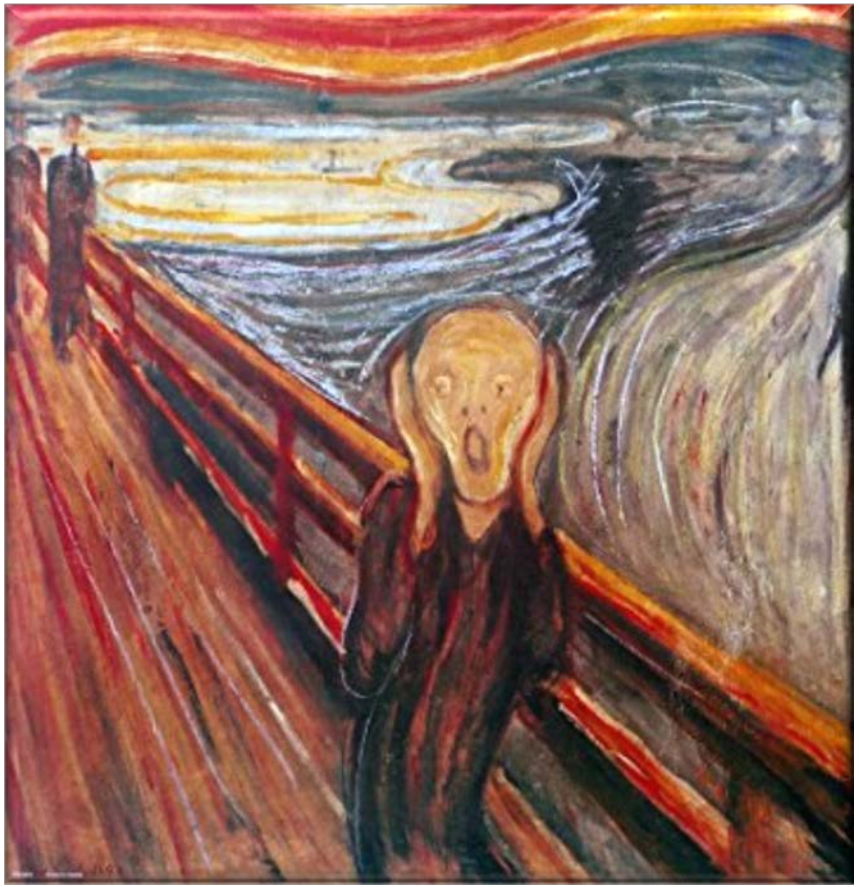
```
ExecutorService executor =
  Executors.newFixedThreadPool
    (numOfBeings,
     mThreadFactory);
...
CyclicBarrier entryBarrier =
  new CyclicBarrier(numOfBeings+1);

CountDownLatch exitBarrier =
  new CountDownLatch(numOfBeings);

for (int i=0; i < beingCount; ++i)
  executor.execute
    (makeBeingRunnable(i,
     entryBarrier,
     exitBarrier));
```

Create a fixed-sized thread pool & also coordinate the starting & stopping of multiple tasks that acquire/release shared resources

See github.com/douglascraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication

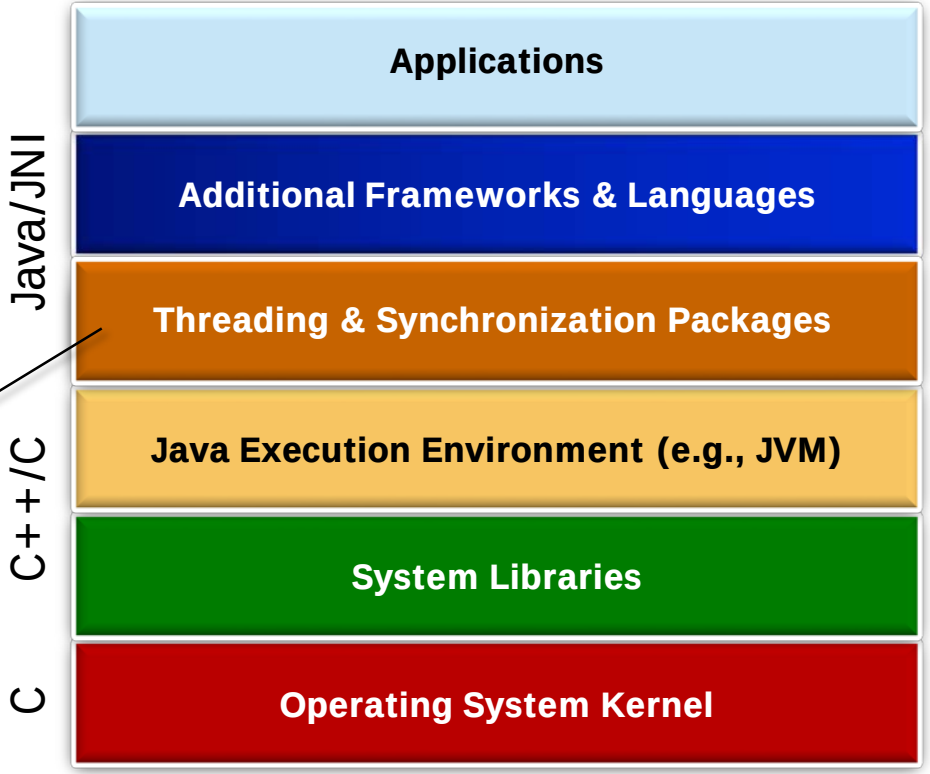
A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
  - Focus on course-grained “task parallelism” whose computations can run concurrently
  - Feature-rich & optimized, but also tedious & error-prone to program



A Brief History of Concurrency & Parallelism in Java

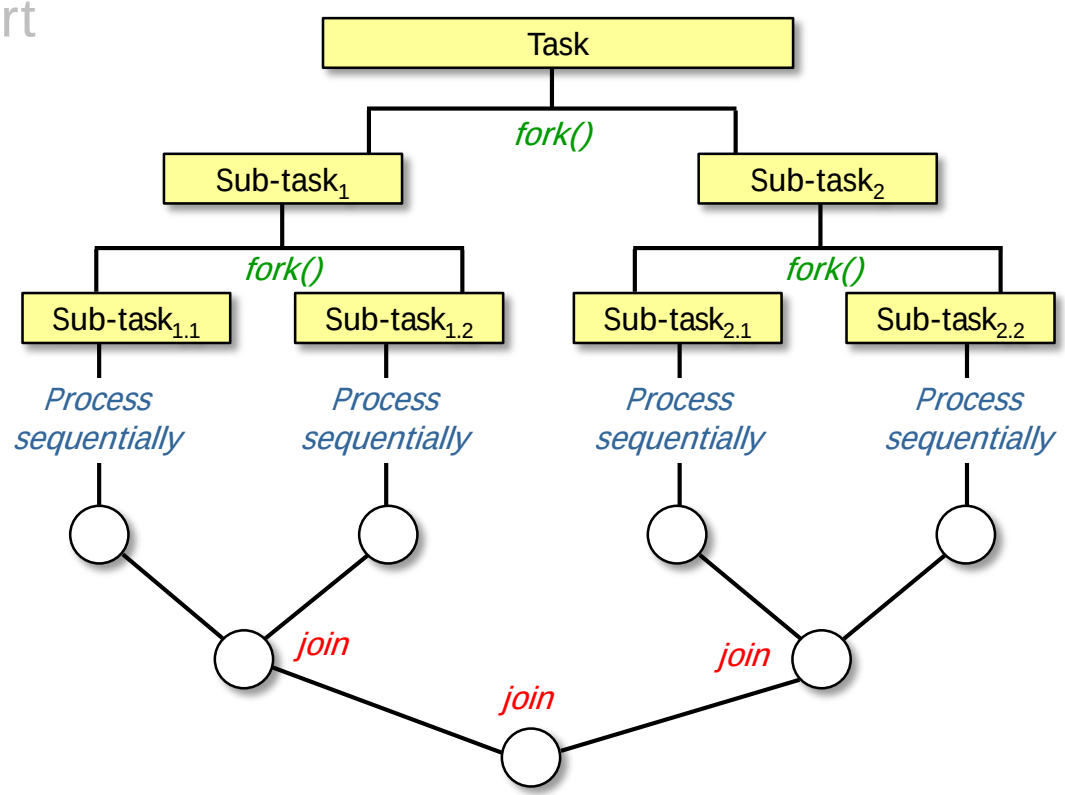
- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support



e.g., Java fork-join pool available in Java 1.7

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
  - Focus on data parallelism that runs the same task on different data elements



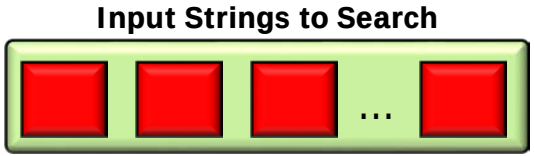
See en.wikipedia.org/wiki/Data_parallelism

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
  - Focus on data parallelism that runs the same task on different data elements

```
List<List<SearchResults>>
  listOfListOfSearchResults =
    ForkJoinPool
      .commonPool()
      .invoke(new
        SearchWithForkJoinTask
          (inputList,
           mPhrasesToFind, ...));
```

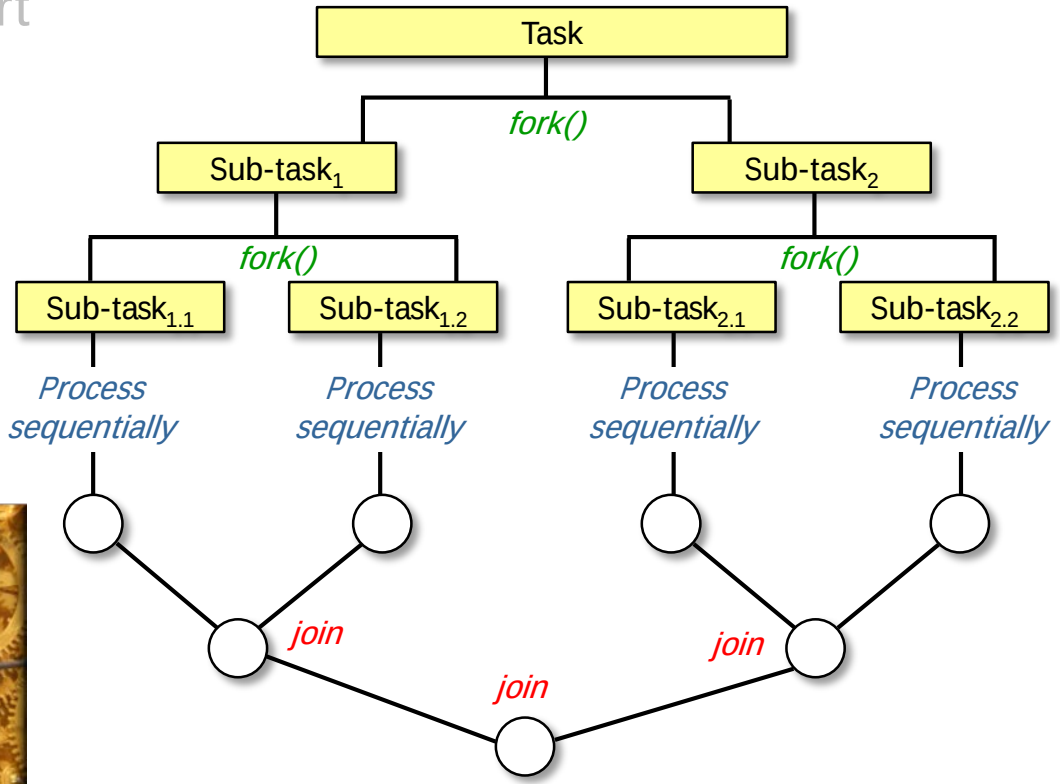
Use a common fork-join pool to search input strings to locate phrases that match



See github.com/douglascraigschmidt/LiveLessons/tree/master/SearchStreamForkJoin

A Brief History of Concurrency & Parallelism in Java

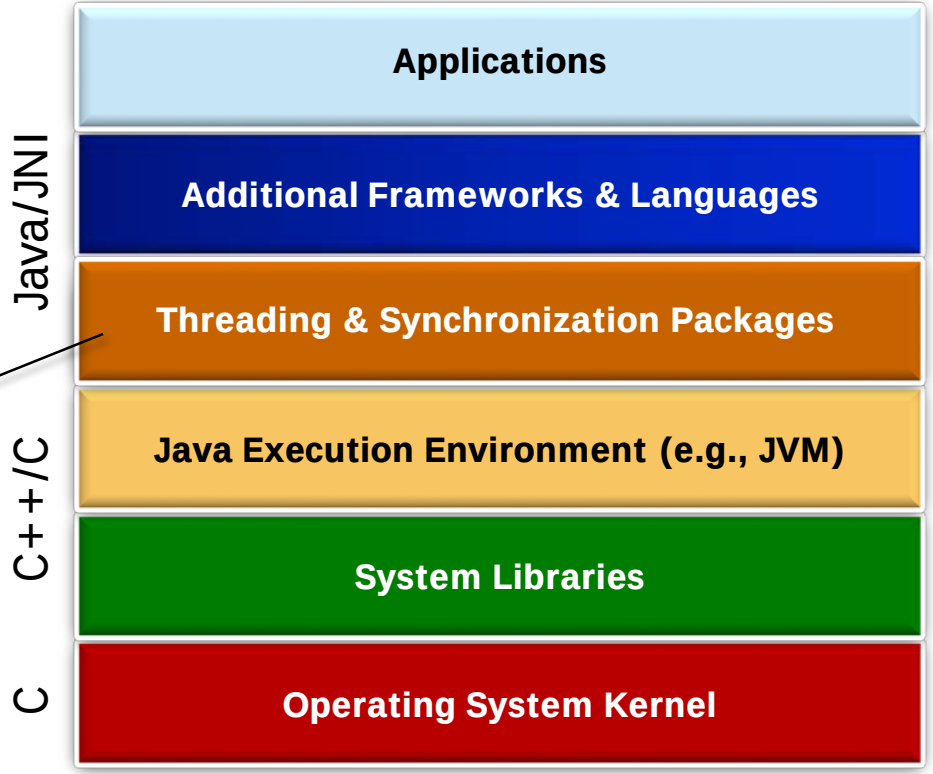
- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
  - Focus on data parallelism that runs the same task on different data elements
  - Powerful & scalable, but tricky to program correctly



A Brief History of Concurrency & Parallelism in Java

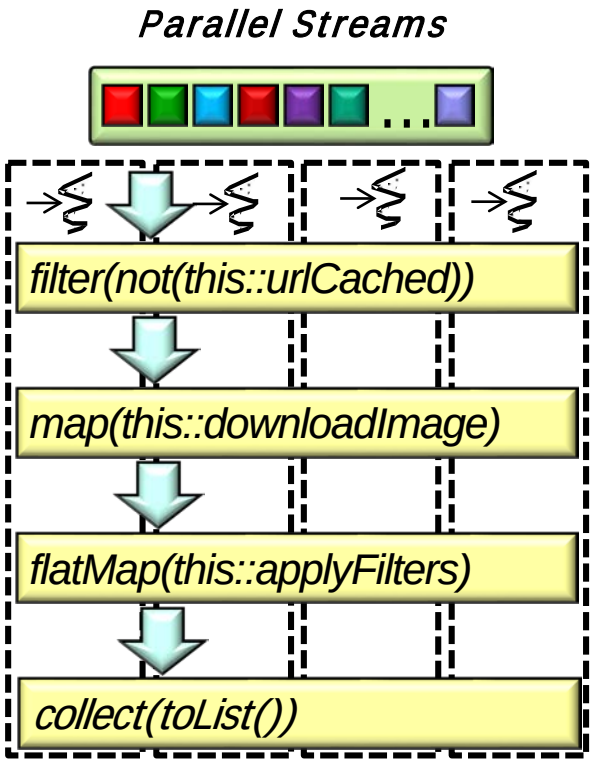
- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
- Advanced parallelism support

e.g., Java parallel streams & completable futures available in Java 1.8



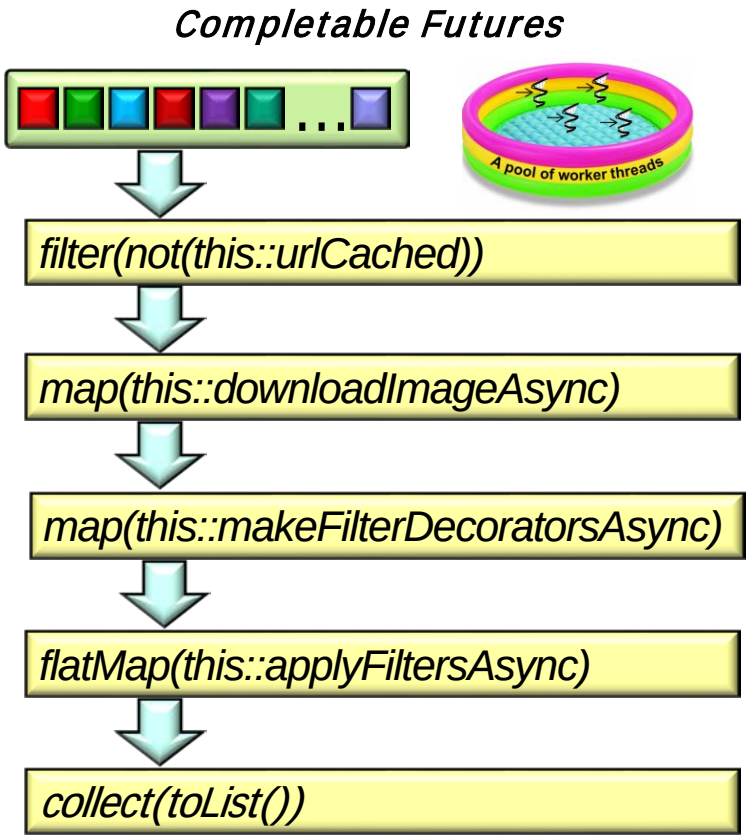
A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
- Advanced parallelism support
  - Focus on functional programming for data parallelism



A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
- Advanced parallelism support
  - Focus on functional programming for data parallelism & asynchrony



A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
- Advanced parallelism support
  - Focus on functional programming for data parallelism & asynchrony

```
List<Image> images = urls
   .parallelStream()
   .filter(not(urlCached()))
   .map(this::downloadImage)
   .flatMap(this::applyFilters)
   .collect(toList());
```

Download images that aren't already cached from a list of URLs & process/store the images in parallel

See github.com/douglascraigschmidt/LiveLessons/tree/master/ImageStreamGang

A Brief History of Concurrency & Parallelism in Java

- Foundational concurrency support
- Advanced concurrency support
- Foundational parallelism support
- Advanced parallelism support
  - Focus on functional programming for data parallelism & asynchrony
  - Strikes an effective balance between productivity & performance

End of Background on Java Concurrency & Parallelism