Learning Objectives in this Part of the Lesson

- Learn the history of Java concurrency & parallelism

JAVA HISTORY

Applications
- Additional Frameworks & Languages
- Threading & Synchronization Packages
- Java Execution Environment (e.g., JVM)
- System Libraries
- Operating System Kernel
Learning Objectives in this Part of the Lesson

- Learn the history of Java concurrency & parallelism

Additional Frameworks & Languages
- Java/JNI
- C++/C
- Java Execution Environment (e.g., JVM)
- System Libraries
- Operating System Kernel

Applications

Hopefully, you’ll already know some of this!!!
A Brief History of Concurrency in Java
A Brief History of Concurrency in Java

- Foundational concurrency support

See [en.wikipedia.org/wiki/Java_version_history#JDK_1.0](en.wikipedia.org/wiki/Java_version_history#JDK_1.0)
A Brief History of Concurrency 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 in Java

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

```java
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/douglasicraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue](https://github.com/douglasicraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue)
A Brief History of Concurrency in Java

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

SimpleBlockingBoundedQueue<Integer>

```java
SimpleBlockingBoundedQueue<> 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](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue)
A Brief History of Concurrency in Java

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

```java
class SimpleBlockingBoundedQueue{

    public E take() {...
        synchronized (this) {
            while (mList.isEmpty())
                wait();
            notifyAll();
            return mList.poll();
        }
    }
}
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue](https://github.com/douglascraigschmidt/LiveLessons/tree/master/SimpleBlockingQueue)
A Brief History of Concurrency 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 in Java

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

See en.wikipedia.org/wiki/No_Silver_Bullet
A Brief History of Concurrency in Java

- Advanced concurrency support

e.g., Java executor framework, synchronizers, blocking queues, atomics, & concurrent collections that became available in Java 5+

See [en.wikipedia.org/wiki/Java_version_history#J2SE_5.0](en.wikipedia.org/wiki/Java_version_history#J2SE_5.0)
A Brief History of Concurrency in Java

- Advanced concurrency support
- Focus on course-grained “task parallelism”

See en.wikipedia.org/wiki/Task_parallelism
A Brief History of Concurrency in Java

- Advanced concurrency support
- Focus on course-grained “task parallelism”
  - e.g., concurrently run various tasks at the same time
A Brief History of Concurrency in Java

- Advanced concurrency support
- Focus on course-grained “task parallelism”
- e.g., concurrently run various tasks at the same time

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

```java
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));
```

See [github.com/douglascraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication](https://github.com/douglascraigschmidt/LiveLessons/tree/master/PalantiriManagerApplication)
A Brief History of Concurrency in Java

• Advanced concurrency support
  • Focus on course-grained “task parallelism”
  • Feature-rich & optimized, but also tedious & error-prone to program
A Brief History of Parallelism in Java
A Brief History of Parallelism in Java

• Foundational parallelism support

See en.wikipedia.org/wiki/Java_version_history#Java_SE_7
A Brief History of Parallelism in Java

- Foundational parallelism support
- Focus on finer-grained data parallelism

See en.wikipedia.org/wiki/Data_parallelism
A Brief History of Parallelism in Java

- Foundational parallelism support
- Focus on finer-grained data parallelism
  - e.g., runs the same task on different elements of data by using the “split-apply-combine” model

See [www.jstatsoft.org/article/view/v040i01/v40i01.pdf](http://www.jstatsoft.org/article/view/v040i01/v40i01.pdf)
A Brief History of Parallelism in Java

- Foundational parallelism support
- Focus on finer-grained data parallelism
  - e.g., runs the same task on different elements of data by using the “split-apply-combine” model

```java
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/douglasraigschmidt/LiveLessons/tree/master/SearchForkJoin
A Brief History of Parallelism in Java

- Foundational parallelism support
- Focus on data parallelism
- Powerful & scalable, but tedious to program directly
A Brief History of Parallelism in Java

- Advanced parallelism support

See en.wikipedia.org/wiki/Java_version_history#Java_SE_8
A Brief History of Parallelism in Java

- Advanced parallelism support
- Focus on fine-grained functional programming for data parallelism

See [en.wikipedia.org/wiki/Data_parallelism](en.wikipedia.org/wiki/Data_parallelism)
A Brief History of Parallelism in Java

- Advanced parallelism support
- Focus on fine-grained functional programming for data parallelism & reactive asynchrony

See gist.github.com/staltz/868e7e9bc2a7b8c1f754
A Brief History of Parallelism in Java

- Advanced parallelism support
- Focus on fine-grained functional programming for data parallelism & reactive asynchrony

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

*See [github.com/douglasraigschmidt/LiveLessons/tree/master/ImageStreamGang]*

*Synchronously download images that aren’t already cached from a list of URLs & process/store the images in parallel*
A Brief History of Parallelism in Java

- Advanced parallelism support
- Focus on fine-grained functional programming for data parallelism & reactive asynchrony

```java
CompletableFuture<Stream<Image>>
resultsFuture = urls
    .stream()
    .map(this::checkUrlCachedAsync)
    .map(this::downloadImageAsync)
    .flatMap(this::applyFiltersAsync)
    .collect(toFuture())
    .thenApply(stream ->
        log(stream.flatMap
            (Optional::stream),
            urls.size()))
    .join();
```

Combines streams & completable futures to asynchronously download images that aren’t already cached from a list of URLs & process/store the images in parallel.

A Brief History of Parallelism in Java

- Advanced parallelism support
  - Focus on fine-grained functional programming for data parallelism & reactive asynchrony
- Focus on pub/sub reactive streams frameworks

*e.g.*, Java reactive streams made available in Java 9 have enabled the RxJava & Project Reactor frameworks

See [en.wikipedia.org/wiki/Java_version_history#Java_SE_9](en.wikipedia.org/wiki/Java_version_history#Java_SE_9)
A Brief History of Parallelism in Java

- Advanced parallelism support
- Focus on fine-grained functional programming for data parallelism & reactive asynchrony
- Focus on pub/sub reactive streams frameworks

List<Image> filteredImages = ReactorUtils
  .fromIterableParallel(urls)
  .filter(url -> !urlCached(url))
  .map(this::blockingDownload)
  .flatMap(this::applyFilters)
  .sequential()
  .collectList()
  .block();

Applies RxJava & Project Reactor reactive streams to asynchronously 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 Parallelism in Java

- Advanced parallelism support
  - Focus on fine-grained functional programming for data parallelism & reactive asynchrony
  - Focus on pub/sub reactive streams frameworks
- Strikes an effective balance between productivity & performance
A Brief History of Parallelism in Java

- Advanced parallelism support
  - Focus on fine-grained functional programming for data parallelism & reactive asynchrony
- Focus on pub/sub reactive streams frameworks
- Strikes an effective balance between productivity & performance
- However, may be overly prescriptive
The Evolution of Java from Concurrency to Parallelism
The Evolution of Java from Concurrency to Parallelism

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

See [www.youtube.com/watch?v=NsDE7E8sIdQ](http://www.youtube.com/watch?v=NsDE7E8sIdQ)
The Evolution of Java from Concurrency to 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](http://www.infoq.com/presentations/parallel-java-se-8)
The Evolution of Java from Concurrency to Parallelism

• Rob Pike also has a good talk that explains the differences between concurrency & parallelism

His talk emphasizes that concurrency is about dealing with lots of things at once, whereas parallelism is about doing lots of things at once

See [www.youtube.com/watch?v=cN_DpYBzKso](http://www.youtube.com/watch?v=cN_DpYBzKso)
End of the History of Concurrency & Parallelism Support in Java