Learning Objectives in this Part of the Lesson

- Understand the purpose of the Java executor framework
- Know the types of thread pools supported by the framework
- Recognize a human known use of thread pools
- Learn the key interfaces the framework provides
Key Java Executor Framework Interfaces
The Java executor framework contains several key interfaces:

- `Executor` interface:
  - `execute(Runnable): void`

- `ExecutorService` interface:
  - `shutdown(): void`
  - `shutdownNow(): List<Runnable>`
  - `isShutdown(): boolean`
  - `isTerminated(): boolean`
  - `awaitTermination(long, TimeUnit): boolean`
  - `submit(Callable<T>): Future<T>`
  - `submit(Runnable, T)`
  - `submit(Runnable)`
  - `invokeAll(Collection<? extends Callable<T>>): List<Future<T>>` (or `invokeAll(Collection<? extends Callable<T>>, long, TimeUnit)`)
  - `invokeAny(Collection<? extends Callable<T>>)` (or `invokeAny(Collection<? extends Callable<T>>, long, TimeUnit)`)

- `ScheduledExecutorService` interface:
  - `schedule(Runnable, long, TimeUnit)`
  - `schedule(Callable<V>, long, TimeUnit)`
  - `scheduleAtFixedRate(Runnable, long, long, TimeUnit)`
  - `scheduleWithFixedDelay(Runnable, long, long, TimeUnit)`

- `CompletionService<V>` interface:
  - `submit(Callable<V>)`
  - `submit(Runnable, V)`
  - `take()`
  - `poll()`
  - `poll(long, TimeUnit)`

See [docs.oracle.com/javase/tutorial/essential/concurrency/executors.html](https://docs.oracle.com/javase/tutorial/essential/concurrency/executors.html)
The Java executor framework contains several key interfaces:

- **Executor**
  - Provides a means of submitting new runnable tasks for execution
  - Defines a simple API that decouples task submission from the mechanics of how each task will be run

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/Executors.html)
The Java executor framework contains several key interfaces:

- **Executor**
- **ExecutorService**
  - Extends Executor to manage task & executor lifecycles

```java
// Java Interface
interface ExecutorService

- shutdown():void
- shutdownNow():List<Runnable>
- isShutdown():boolean
- isTerminated():boolean
- awaitTermination(long, TimeUnit):boolean
- submit(Callable<T>):Future<T>
- submit(Runnable, T):Future<T>
- submit(Runnable):Future<?>
- invokeAll(Collection<? extends Callable<T>>):List<Future<T>>
- invokeAny(Collection<? extends Callable<T>>)
- invokeAny(Collection<? extends Callable<T>>),long, TimeUnit
```

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ExecutorService.html](https://docs.oracle.com/javase/8/docs/api/java/util/concurrent/ExecutorService.html)
The Java executor framework contains several key interfaces:

- **Executor**
- **ExecutorService**
- **ScheduledExecutorService**
  - Extends ExecutorService to support future and/or periodic execution of tasks

See [docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledExecutorService.html](docs.oracle.com/javase/8/docs/api/java/util/concurrent/ScheduledExecutorService.html)
The Java executor framework contains several key interfaces:

- Executor
- ExecutorService
- ScheduledExecutorService
- CompletionService
  - Decouples task invocation from processing of completed task results

```
<<Java Interface>>

CompletionService<V>

- submit(Callable<V>):Future<V>
- submit(Runnable,V):Future<V>
- take():Future<V>
- poll():Future<V>
- poll(long, TimeUnit):Future<V>
```
End of Overview of the Java Executor Framework (Part 2)