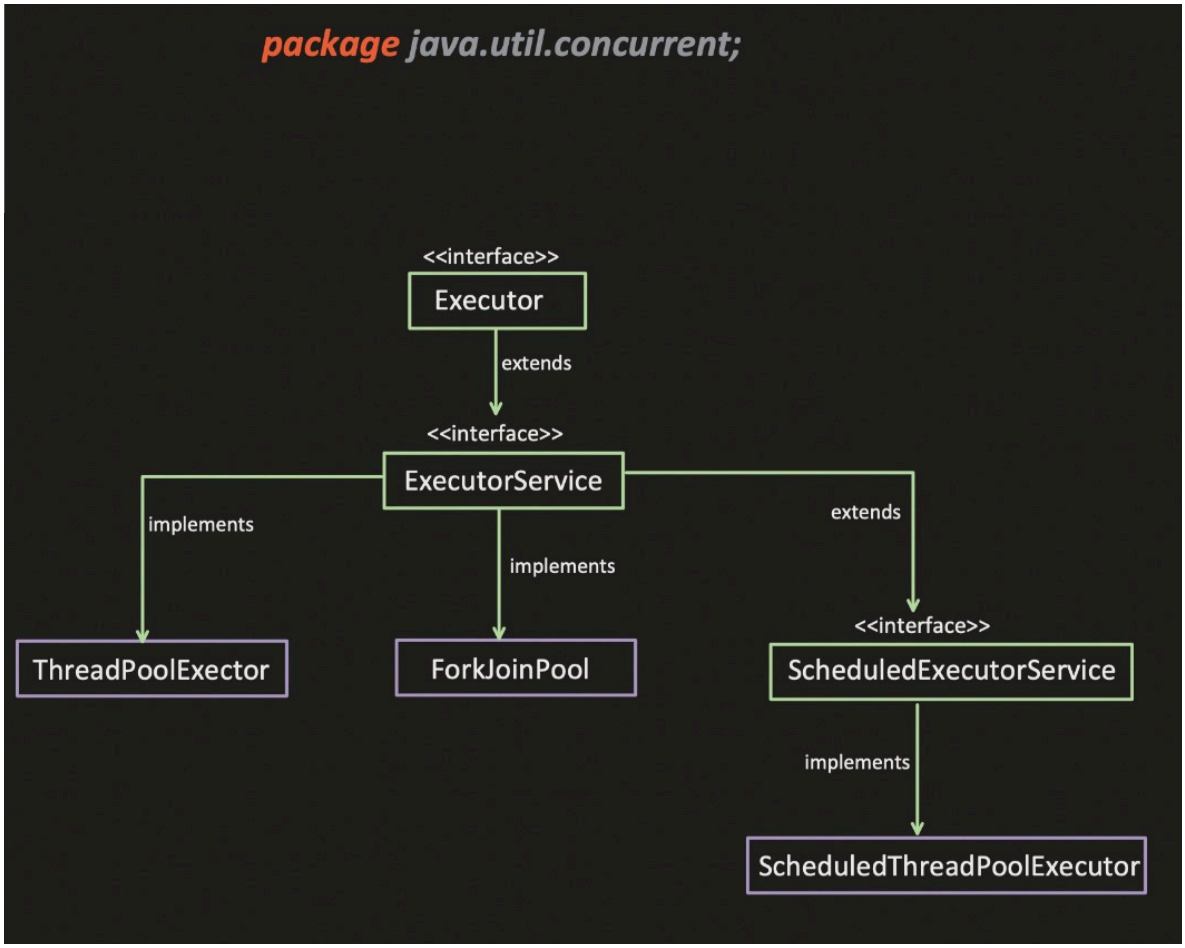

🔶 1. Executor (Top-most interface)

This is the *root* interface for running tasks asynchronously.

👉 What it has?

Only **one method**:

```
void execute(Runnable command);
```

-
- It just *submits* the task; no return value, no lifecycle management.

📌 Executor = Someone who can run a Runnable.

🔶 2. ExecutorService (extends Executor)

A more powerful version of `Executor`.

👉 Adds extra abilities:

- Submit tasks and get **Future**
- Stop the executor (`shutdown`, `shutdownNow`)
- Manage asynchronous tasks

Key methods:

```
Future<?> submit(Runnable task);
<T> Future<T> submit(Callable<T> task);
void shutdown();
```

📌 ExecutorService = Executor + lifecycle + submit + Future support.

🔶 3. Implementations of ExecutorService

The diagram shows two major implementations:

🟣 ThreadPoolExecutor

This is the **most commonly used thread pool**.

Used for:

- Fixed thread pool
- Cached thread pool
- Single thread executor

- Custom thread pool

Spring Boot internally also uses this.

Features:

- Configurable core pool size
- Queue type (LinkedBlockingQueue, SynchronousQueue etc.)
- Rejection handler

🟣 ForkJoinPool

Designed for:

- CPU-intensive parallel tasks
- Divide-and-conquer algorithms
- Work stealing

Used in:

- Java parallel streams
- RecursiveTask / RecursiveAction

🔶 4. ScheduledExecutorService (extends ExecutorService)

This interface adds **task scheduling** capability.

Supports:

- Run a task after a delay

- Run a task repeatedly at fixed rate
- Run a task at fixed delay

🟣 ScheduledThreadPoolExecutor

This is the implementation of `ScheduledExecutorService`.

Equivalent of:

- `Timer`
- `TimerTask`

But:

- More scalable
- Uses thread pools instead of a single thread

💡 Summary in Plain English

Concept	Meaning
Executor	Runs a Runnable
ExecutorService	Executor + submit tasks + Future + shutdown
ThreadPoolExecutor	Normal thread pool
ForkJoinPool	Parallel task executor (work stealing)
ScheduledExecutorService	Runs tasks on schedule
ScheduledThreadPoolExecutor	Thread pool version of scheduler

✔ Why this diagram is important in Spring Boot?

When you use:

- `@Async`
- `@Scheduled`
- Thread pools in WebFlux

➡ Spring Boot internally uses these classes:

- For async calls → `ThreadPoolExecutor`
- For scheduled jobs → `ScheduledThreadPoolExecutor`