Thread Pool

ExecutorService & FixedThreadPool

1. new一个实现Runnable

2. new Thread(runnable)

3. thread.start()　　// JVM创建新线程并运行

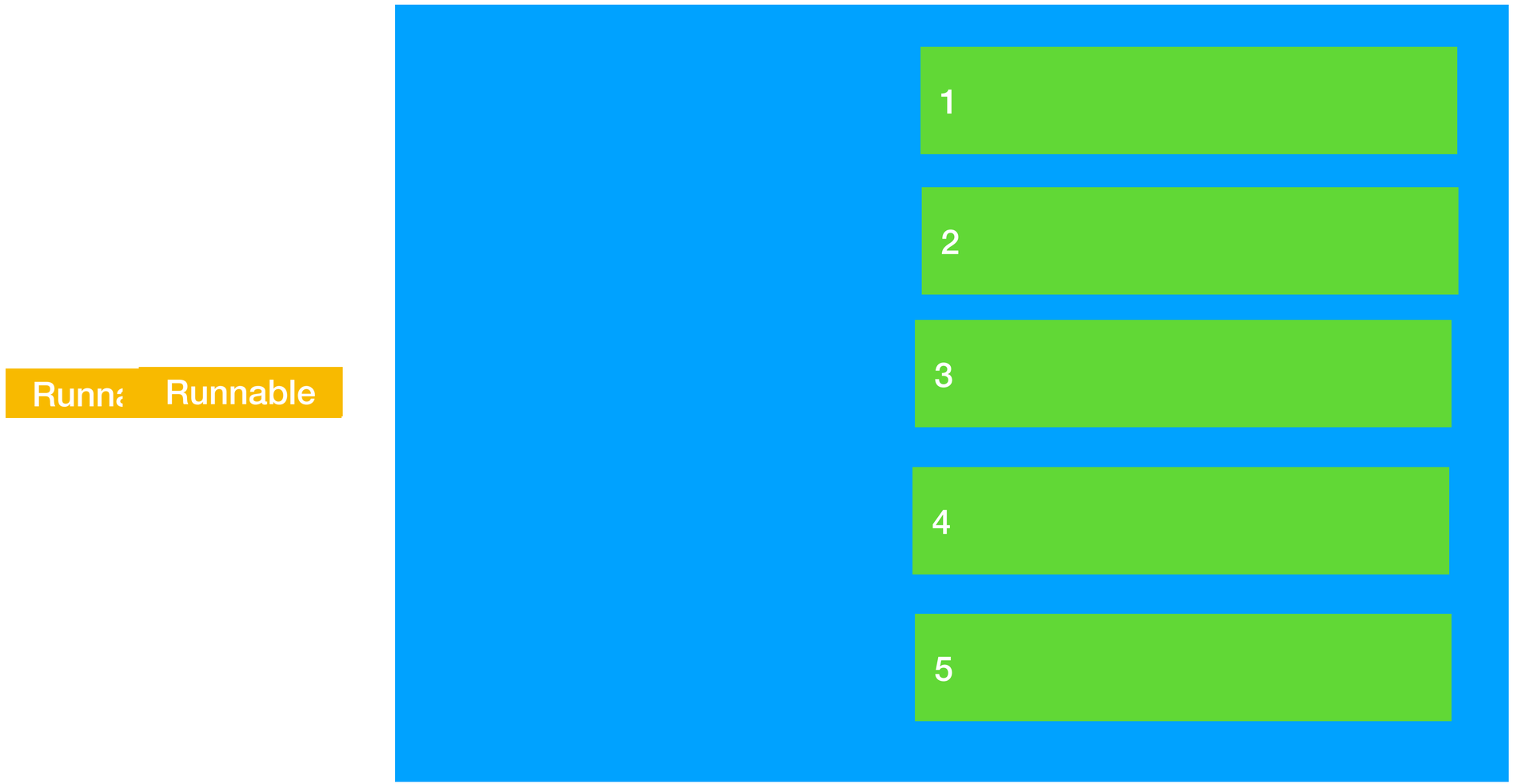
java.util.concurrent.ExecutorService & java.util.concurrent.Executors

```
ExecutorService service = Executors.newFixedThreadPool(5);

Runnable r = … ;

Future<?> f = service.submit(r);
```

ExecutorService提交任务

Future<?>	submit(Runnable task)
<T> Future<T>	submit(Runnable task, T result)
<T> Future<T>	submit(Callable<T> task)

FiexedThreadPool



关闭ExecutorSercice

- void shutdown() 不再接收新任务，已经提交的任务还是会继续

  shutdown后继续submit任务，会抛出RejectExecutionException

- boolean awaitTerminate(long timeout, TimeUnit time unit)

  - **全部任务执行完毕　　返回 true**
  - **超时　　返回 false**
  - **线程被中断 InterruptedException**

- List<Runnable> shutdownNow() 尝试结束正在执行的任务

  - **调用所有正在运行线程的interrupt方法**
  - **返回尚未执行的任务列表**

关闭ExecutorSercice（例子）

```
service.shutdown(); // 禁止service再接收新任务的submit
try {
    // service中还有正在运行的任务，等一段时间，让他们执行完毕
    if (!service.awaitTermination(60, TimeUnit.SECONDS)) {
        service.shutdownNow(); // shutdownNow会给正在运行的线程发送Interrupt中断；要求所有线程退出
        // 等待一段时间以便线程处理中断退出
        if (!service.awaitTermination(30, TimeUnit.SECONDS)) {
            //线程没有在30秒内相应中断并退出
            System.err.println("无法终止 " + service);
        }
    }
} catch (InterruptedException ie) {
    // executorservice的主线程收到中断；尝试强制任务线程退出
    service.shutdownNow();
    // 设置中断标识（有InterruptedException不会有中断标识，所以此处再次设置，以便其他地方使用）
    Thread.currentThread().interrupt();
}
```