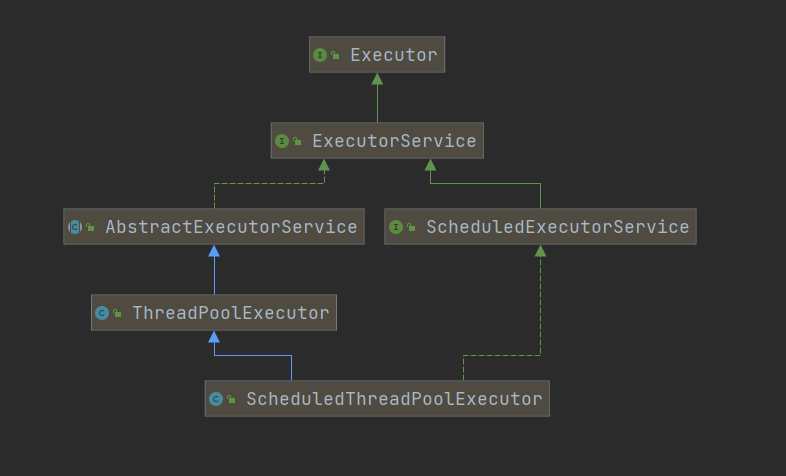
1. ThreadPool
   1. 线程池的优势
      1. 线程池的好处是减少在创建和销毁线程上所消耗的时间以及系统资源的开销，解决资源不足的问题。
      2. 如果不使用线程池，有可能造成系统创建大量同类线程而导致消耗完内存或者“过度切换”的问题。
   2. 分析线程池结构图



我们可以直到 ExecutorService是继承了Executor因此我们可以使用ExecutorService 来接收线程池。使用Executors线程池工作类来创建线程池；

* 1. Executors线程池工具类的核心方法
     + 1. 固定线程数：

ExecutorService threadPool =

Executors.newFixedThreadPool(5);

* + - 1. 一个线程数:

ExecutorService threadPool =

Executors.newSingleThreadExecutor();

* + - 1. 可扩展线程池(根据所需要的线程数来进行线程的创建)

ExecutorService threadPool =

Executors.newSingleThreadExecutor();

范例:

|  |
| --- |
| package cn.gsdx.JUC;  import java.util.concurrent.ExecutorService;  import java.util.concurrent.Executors;  public class ThreadPoolDemo {  public static void main(String[] args) {  ExecutorService threadPool = Executors.newCachedThreadPool();  try {  for (int i =0 ;i <10 ;i++){  threadPool.execute(  () -> {  System.out.println(Thread.currentThread().getName());  });  }  }catch (Exception e){  e.printStackTrace();  }finally {  threadPool.shutdown();  }  }  } |

* 1. 使用说明：Executors返回的线程池对象的弊端如下：

1）FixedThreadPool和SingleThreadPool:

  允许的请求队列长度为Integer.MAX\_VALUE，可能会堆积大量的请求，从而导致OOM。

2）CachedThreadPool:

  允许的创建线程数量为Integer.MAX\_VALUE，可能会创建大量的线程，从而导致OOM