并发基础篇（2）：Thread 类的 API 总结

Thread 类是 java 中的线程类，提供给用户用于创建、操作线程、获取线程的信息的类。是 java 线程一切的基础，掌握这个类是非常必须的，先来看一下它的 API.

1、字段摘要

static int MAX_PRIORITY	线程可以具有的最高优先级
static int MIN_PRIORITY	线程可以具有的最低优先级
static int NORM_PRIORITY	分配给线程的默认优先级

2、构造方法摘要

Thread() ：	分配新的 Thread 对象。
Thread(Runnable target)	分配新的 Thread 对象。
Thread(Runnable target, String name)	分配新的 Thread 对象。
Thread(String name)	分配新的 Thread 对象。参数是线程名称
Thread(ThreadGroup group, Runnable target)	分配新的 Thread 对象。
Thread(ThreadGroup group, Runnable target, String name)	分配新的 Thread 对象，以便将 target 作为其运行对象，将指定的 name 作为其名称，并作为 group 所引用的线程组的一员。
Thread(ThreadGroup group, Runnable target, String name, long stackSize)	分配新的 Thread 对象，以便将 target 作为其运行对象，将指定的 name 作为其名称，作为 group 所引用的线程组的一员，并具有指定的堆栈大小。

Thread(ThreadGroup group, String name)	分配新的 Thread 对象。

注意：

线程组（ThreadGroup ）已经算是过时的，被抛弃的了，所以不需要去研究线程组，仅仅知道有这个存在就可以了。按照《java 编程思想》的说法，线程组一次不成功的尝试。

3、方法摘要

3.1、静态方法

static Thread currentThread()	返回对当前正在执行的线程对象的引用。
static int activeCount()	返回当前线程的线程组中活动线程的数目。
static boolean interrupted()	测试当前线程是否已经中断。
static void sleep(long millis)	在指定的毫秒数内让当前正在执行的线程休眠（暂停执行），此操作受到系统计时器和调度程序精度和准确性的影响。
static void sleep(long millis, int nanos)	在指定的毫秒数加指定的纳秒数内让当前正在执行的线程休眠（暂停执行），此操

	作受到系统计时器和调度程序精度和准确性的影响。
static void yield()	暂停当前正在执行的线程对象，并执行其他线程。
static boolean holdsLock(Object obj)	当且仅当当前线程在指定的对象上保持监视器锁时，才返回true。
static void dumpStack()	将当前线程的堆栈跟踪打印至标准错误流。
static int enumerate(Thread[] tarray)	将当前线程的线程组及其子组中的每一个活动线程复制到指定的数组中。
static Map< Thread,StackTraceElement[]> getAllStackTraces()	返回所有活动线程的堆栈跟踪的一个映射。
static Thread.UncaughtExceptionHandler getDefaultUncaughtExceptionHandler()	返回线程由于未捕获到异常而突然终止时调用的默认处理程序。

static void setDefaultUncaughtExceptionHandler(Thread.UncaughtExceptionHandler eh)	设置当线程由于未捕获到异常而突然终止，并且没有为该线程定义其他处理程序时所调用的默认处理程序。

3.2、获取线程的信息

long getId()	返回该线程的唯一标识符。
String getName()	返回该线程的名称。
int getPriority()	返回线程的优先级。
Thread.State getState()	返回该线程的状态。
ThreadGroup getThreadGroup()	返回该线程所属的线程组。
ClassLoader getContextClassLoader()	返回该线程的上下文 ClassLoader。
StackTraceElement[] getStackTrace()	返回一个表示该线程堆栈转储的堆栈跟踪元素数组。
Thread.UncaughtExceptionHandler getUncaughtExceptionHandler()	返回该线程由于未捕获到异常而突然终止时调用的处理程序。

3.3、线程的其他操作

void checkAccess()	判定当前运行的线程是否有权修改该线程。
void interrupt()	中断线程。
boolean isInterrupted()	测试线程是否已经中断。

boolean isAlive()	测试线程是否处于活动状态。
boolean isDaemon()	测试该线程是否为守护线程。即后台线程
void setName(String name)	改变线程名称，使之与参数 name 相同。
void setPriority(int newPriority)	更改线程的优先级。
void setDaemon(boolean on)	将该线程标记为守护线程或用户线程。又叫后台线程（是后台提供一种通用的服务线程）
void setContextClassLoader(ClassLoader cl)	设置该线程的上下文 ClassLoader。
void setUncaughtExceptionHandler(Thread.UncaughtExceptionHandler eh)	设置该线程由于未捕获到异常而突然终止时调用的处理程序。
void join()	等待该线程终止。
void join(long millis)	等待该线程终止的时间最长为 millis 毫秒。
void join(long millis, int nanos)	等待该线程终止的时间最长为 millis 毫秒 + nanos 纳秒。
void start()	使该线程开始执行；Java 虚拟机调用该线程的 run 方法。
String toString()	返回该线程的字符串表示形式，包括线程名称、优先级和线程组。