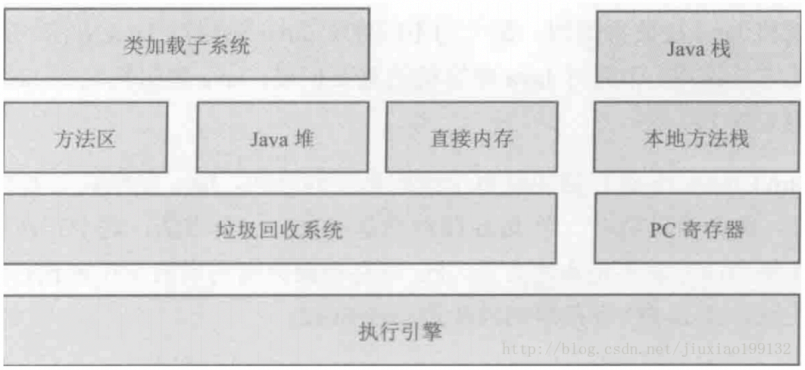
概念

一个应用程序在操作系统中对应一个进程
一个进程里面可以包含多个线程
线程比进程更加轻量，因此创建一个线程会比创建一个进程更加快速，消耗的资源更少
在操作系统中实际上还有一种叫做协程，比线程更加轻量级，没有cpu上线程的切换，并行执行任务，这里不做展开，python中有说明

jvm模型简述



如上图所示，Java虚拟机由9个部分组成，下面2个部分区域与线程有直接的关系

- 每一个Java虚拟机线程都有一个私有的Java栈。一个线程的Java栈在线程创建的时候被创建。Java保存着帧信息，Java栈中保存着局部变用、返回密切相关。

- PC寄存器(程序计数器)也是每个线程私有的空间，Java虚拟机会为每一个Java线程创建PC寄存器。在任意时刻，一个Java线程总是在执行

为当前方法。如果当前方法不是本地方法，PC寄存器就会指向当前正在被执行的指令。如果当前方法是本地方法，那么PC寄存的值就是un

线程的生命周期

线程的状态定义在 java.lang.Thread.State 中，一共有五个状态

- NEW：创建状态并未执行，通过start()方法开始线程执行，进入RUNNABLE
- RUNNABLE：运行状态
- BLOCKED：阻塞状态，如果在执行的过程中遇到了synchronize 关键字标注的同步块，就会进入阻塞状态
- WAITING：等待状态，表示线程进入了等待状态，直到notify()方法执行通知后，线程继续进入运行状态
- TERMINATED：终止状态

线程的基本操作

新建线程

1 继承Thread类
2 实现Runable接口

线程终止(Stop)

正常情况下，线程执行完毕就会终止，不用手动关闭
但是有时候也会有一些情况下是有例外的，比如线程本身就在一个无限循环里面执行
关闭线程在jdk中提供了一个stop方法，但是这个方法已经过时了，因为这个方法太暴力，直接关闭线程，不管线程是否执行完毕

线程中断(interrupt)

线程直接终止是非危险的操作，那么可以通过中断操作来进行线程的停止

```
public static void main(String[] args) {
    Thread t = new Thread(new T1());
    t.start();
    t.interrupt();
}

static class T1 implements Runnable{
    @Override
    public void run() {
        while (true){
            System.out.println("thread running");
            if (Thread.currentThread().isInterrupted()){
                System.out.println("thread interrupted");
                break;
            }
        }
    }
}
```

挂起（suspend）和继续执行（resume）线程

这两个方法官方已经不推荐使用了

等待线程结束（join）和谦让(yeild)

join表示在两个线程执行的过程中A线程执行某一项工作，B线程执行一项工作依赖A线程执行初始化完毕之后才能开始执行，则可以用的joir

yield表示当前线程让出cpu执行，但是让出线程之后不代表这个线程就不执行了，而是由当前线程与其他线程进行资源的抢夺，当前线程能

等待(wait)与通知(notify)

在调用wait的时候，线程自动释放其占有的对象锁，同时不会去申请对象锁。当线程被唤醒的时候，它才再次获得了去获得对象锁的权利。

notify唤醒在等待该对象同步锁的线程(只唤醒一个,如果有多个在等待),注意的是在调用此方法的时候，并不 能确切的唤醒某一个等待状态的而且不是按优先级 notifyAll唤醒所有等待的线程

线程优先级

```
/**
 * The minimum priority that a thread can have.
 */
public final static int MIN_PRIORITY = 1;

/**
 * The default priority that is assigned to a thread.
 */
public final static int NORM_PRIORITY = 5;

/**
 * The maximum priority that a thread can have.
 */
public final static int MAX_PRIORITY = 10;
```

守护线程和用户线程

Java 中的 线程分为两类，分别为 daemon 线程(守护线程）和 user 线程(用户线程)。 在JVM启动时会调用main函数， main函数所在的钱程时-还启动了 好多守护线程， 比如垃圾回收 线程。

那么守护线程 和用户线 程有 什么 区 别呢? 区别之一是当最后一个非守护线程结束时， JVM 会 正常退出，而不管当前是否有守护线程 ，也JVM 的退出。言外之意，只要有一个用户线程还没结束， 正常情况下 JVM 就不会退出。

创建一个守护线程

```
Thread t=new Thread ();
t.setDaemon(true);
```

ThreadLocal

ThreadLocal是JDK提供的，用来创建线程本地变量。如果创建了一个ThreadLocal变量 ，那么访问这个变量的每个线程都会有这个变量的一量时，实际操作的是自己本地内存里面的变量 ，从而避免了线程安全问题。

InheritableThreadLocal

是ThreadLocal的子类，在ThreadLocal中每个类都是变量独享，而这个类可以在继承关系上达到共享