JAVA编程高级

——多线程

Programming Your Future

Programming Your Future

火龙果•整理
uml.org.cn

多线程

目标：

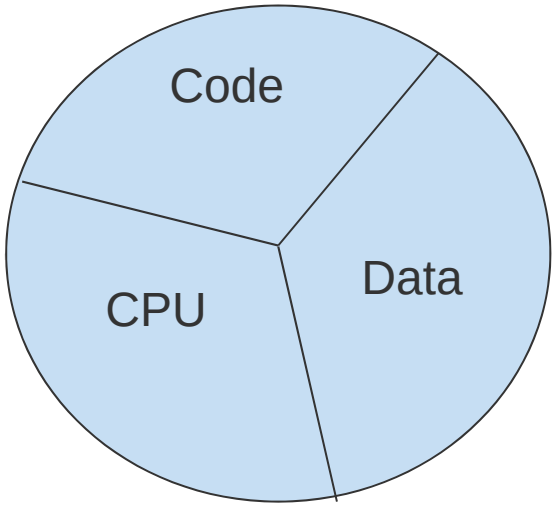
- Java线程机制及线程与进程的区别
- Java线程模型
- Java多线程实现的方式
- 线程的状态及其生命周期
- 线程相关类的常用方法的使用
- 线程的休眠，中断及优先级
- 线程的同步和死锁

教学方法：讲授ppt＋上机练习

火龙果•整理
uml.org.cn

进程

- 每个独立的应用程序就是一个进程，进程的主要作用在于资源的分配，进程与进程之间是不同的应用程序。
- 进程的数据空间只针对该进程共享，不同的进程间则不能共享。每个进程都有独立的代码和数据空间，进程间切换会有较大的开销。
- 一个进程由三个部分来组成：
  - 一个是CPU
  - 一个是执行的程序代码(code)
  - 一个是程序执行所处理的数据(data)



火龙果•整理
uml.org.cn

线程

- 线程是进程中的实体，一个进程可以拥有多个线程，一个线程必须有一个父进程。线程与父进程的其它线程共享该进程所拥有的全部资源。
- 线程可以看成是轻量级的进程，同一类线程共享代码和数据空间，每个线程有独立的运行栈，线程的切换开销小。
- 线程是程序内部的一个顺序控制流
- 多进程：在操作系统上能同时运行多个任务（程序）
- 多线程：在同一个程序中有多个顺序流同时执行

火龙果•整理
uml.org.cn

创建线程

- 建立Threads
  - Java程序中负责Threads这个功能的是java.lang.Thread类。
- 创建Thread类对象有两种方式:
  - 继承Thread类，复写“ run”方法;
  - 实现Runnable接口，然后复写“ run”方法。
- run（）方法：线程的运行体，要将一段代码（线程体）在一个新的线程上运行，该代码应该在一个线程类的run（）方法中。

示例：**ThreadDemo.java、ThreadRunnableDemo.java**

火龙果•整理
uml.org.cn

Threads操作

- 四种操作Threads的基本方法：
  - start
  - sleep
  - join
  - yield

火龙果•整理
uml.org.cn

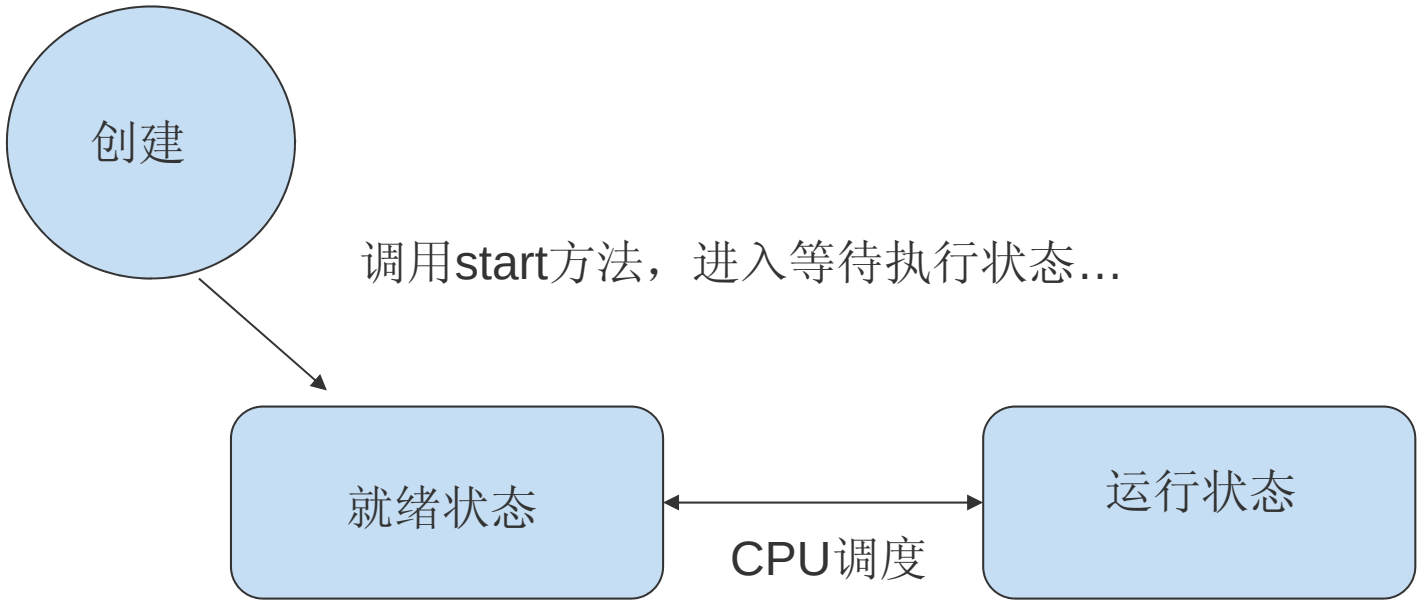
启动Threads

- 启动Thread对象的方法是 ”start”，功能是通知Thread对象可以准备开始执行run方法中的程序了。
- 一个Thread对象只能调用start方法一次，如果对一个已经启动的Thread对象，再次调用start方法，会产生“ IllegalThreadStateException”异常。

火龙果•整理
uml.org.cn

启动Threads后状态转移图

- Threads状态转移图



启动Threads

- 由于系统中同一时刻不止一个Thread对象在执行，而且每种操作系统都有不同的方式，至于什么时候系统会从等待执行的Threads中取出一个来执行，我们无法预测它的执行时间和顺序。
- 当系统选定了一个等待执行中的Thread对象后，它就会从等待执行状态进入执行状态，系统挑选的动作我们称之为 ”CPU调度“ (schedule)。
- 被挑选到的Threads执行一阵子后，就会被系统给换下来回到等待状态，然后又会再挑选下一个Threads执行。

火龙果•整理
uml.org.cn

创建Thread两种方式比较

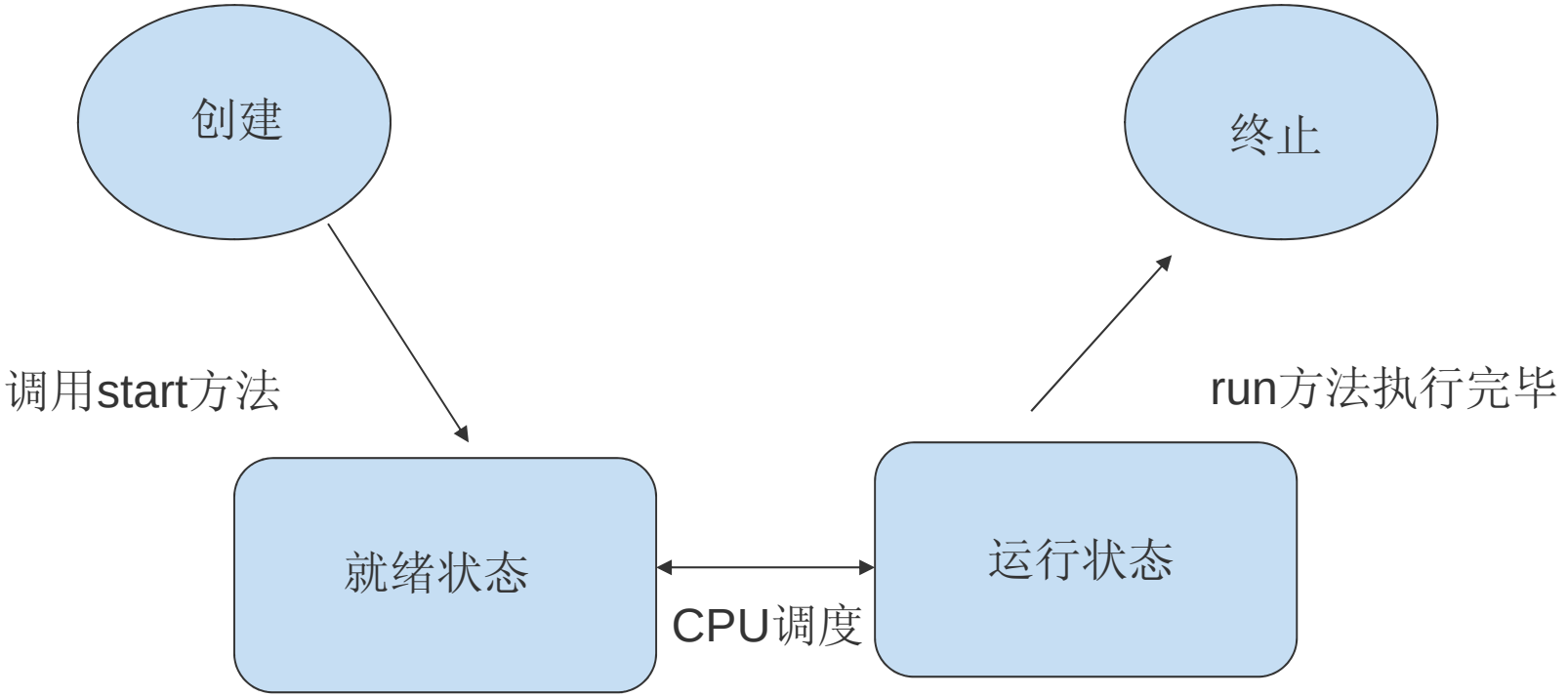
- 无论是通过继承Thread类还是通过实现Runnable接口，实际上最终都需要通过Thread类构造器来创建线程对象，然后调用线程对象的start方法运行线程。
- 使用继承Thread的类，可以直接构造Thread对象；
- 实现Runnable接口的类，是线程对象执行的目标（相当于先建立一个空线程，然后该线程去执行实现了Runnable接口的类对象的相关方法）。

停止Threads

- 让一个线程停止，就是当它把run方法中的程序代码执行完后，就进入”死亡”(dead)状态了。
- 当一个Thread对象进入死亡状态后，就没有任何方法回到其他状态了.
- 要想让Thread对象再重新执行一次，唯一的方法，就是重新再产生一个Thread对象。

停止Threads状态转移

- Threads状态转移图



示例：StopThreads.java

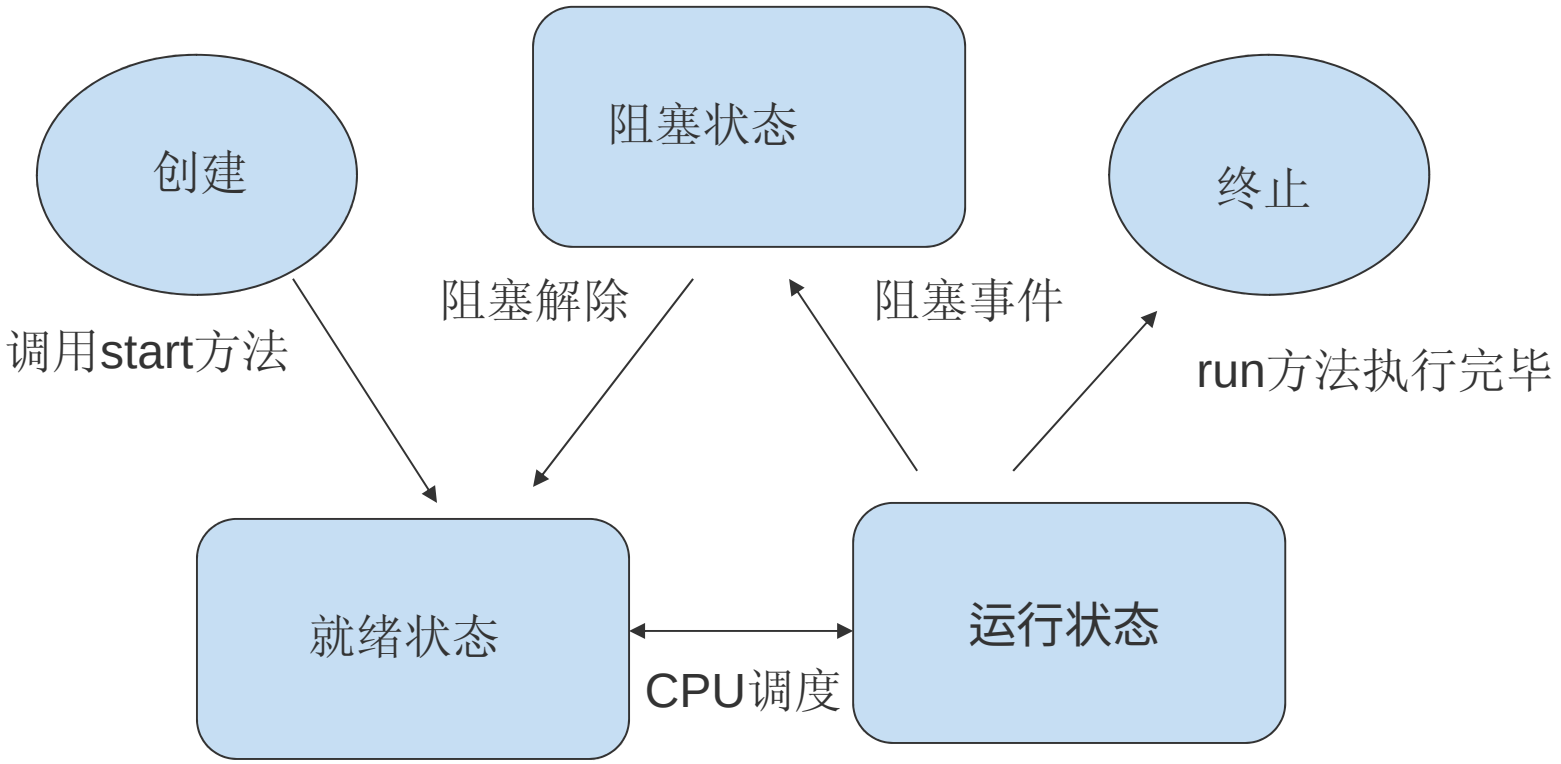
Sleep方法

- Sleep就是让Thread对象“睡”一下，然后再设定一个闹钟，时间一到，它又会恢复执行。
- 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）
- 当一个Thread对象睡醒后，不是立刻进入执行状态来执行，而是进入等待执行状态，在那里等待被系统挑选到，才会被执行！
- sleep方法所传入的时间，只是保证这个Thread对象“至少”会停止这么长的时间。
- 在哪个线程里调用sleep方法，就让哪个线程睡眠。

火龙果•整理
uml.org.cn

线程sleep的状态转移

- Threads状态转移图



示例：SleepDemo.java

Yield方法

- Yield方法使线程直接从执行状态转换到等待执行状态，为其他可运行的线程提供执行机会。

- 示例：yieldDemo.java

Join方法

- 当一个系统中有多个Threads在执行时，有时候某个Threads甲需要等待另一个Threads乙完成时，甲才能继续执行下去，这时甲在乙还未执行完成之前，就必须处于等待状态。
- 要让某个Threads等待另一个Threads执行完后，它才能继续执行，这时我们需要用到 ” join”方法。

- 示例：JoinDemo.java

其它Threads方法

取得Threads信息的相关方法：

- boolean **isAlive**()
  - 测试线程是否处于活动状态
- void **setName**(String name)
  - 改变线程名称，使之与参数 name 相同
- String **getName**()
  - 返回该线程的名称
- static Thread **currentThread**()
  - 返回对当前正在执行的线程对象的引用。

线程的优先级

- Java提供一个线程调度器来监控程序中启动后进入就绪状态的所有线程，线程调度器按照线程的优先级决定应调度哪个线程来执行。
- 线程的优先级用数字表示，范围从1到10，一个线程的缺省优先级是5。
  - **Thread.MIN_PRIORITY** =1
  - **Thread.NORM_PRIORITY** =5
  - **Thread.NAX_PRIORITY** =10
- int **getPriority**()
  - 返回线程的优先级。
- void **setPriority**(int newPriority)
  - 更改线程的优先级

示例：PriorityDemo.java

火龙果•整理
uml.org.cn

练习

- 写一个多线程程序，要求每隔1秒钟显示一次当先时间和日期，10秒钟后停止显示。

火龙果•整理
uml.org.cn

数据同步处理

- 不同的Threads之间，程序代码可以共用，数据也可以共用，只有CPU不能共用，每个Threads都以为自己拥有自己的CPU。
- 多个线程同时访问同一个资源，资源的共用会存在一些问题。
- 为了确保在任何时间点一个共享的资源只被一个线程使用，需要使用线程同步。
- 线程同步：当一个线程运行到需要同步的语句时，CPU不去执行其他线程需要运行的该同步语句，而是等到当前线程执行完同步的语句后，其他线程再执行该语句。

示例：NosynDemo.java

Threads同步

- Threads同步使用synchronized关键字。
- 同步的代码在一个线程执行过程中，不会被另外线程打断。
- 同步代码块语法：

```
synchronized(要取得锁的对象)
{
    要锁定的程序代码
}
```

- 受到synchronized保护的程序代码中，要访问的对象属性必须设定为private。

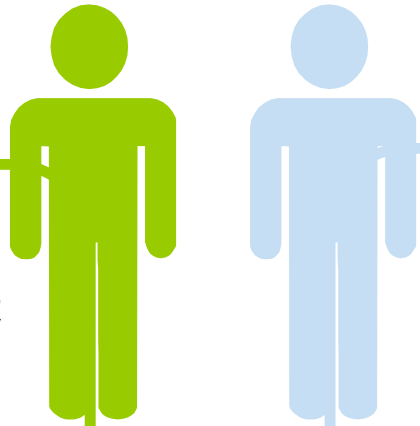
示例： synDemo.java

Threads同步

- synchronized指令另外一种用法：
  - 把Synchronized写在方法声明上，把整个方法锁定。

```
public synchronized void method()
{
    …
}
```

两种同步方式的比较



锁定方法

- 优点：
✓可以显示的知道哪些方法是被锁定的
- 缺点：
✓ 方法中有些程序是不需要保护的，如果该方法执行会花很长时间，那么其他人就要花较多时间等待锁被归还；
✓只能取得自己对象的锁

锁定代码块

- 优点：
✓ 可以针对某段程序代码锁定，不需要浪费时间在别的程序代码上；
✓可以取得不同对象的锁
- 缺点：
✓ 无法显示的得知哪些方法是被锁定的；

线程同步锁

- 乱用synchronized可能会造成系统死锁(Dead Lock)的状况！
- 锁归还的几种方式：
  - 1.执行完锁定的程序代码后，锁就会自动归还；
  - 2.用break语句跳出锁定的语句块，对于写在方法声明的synchronized没有作用；
  - 3.遇到return语句；
  - 4.遇到了异常；
- 避免死锁办法
  - 确定要获得锁的顺序，整个程序都要遵守该顺序。

示例：DeadlockDemo.java

Neusoft

Beyond Technology

Copyright © 2008 版权所有 东软集团

火龙果•整理
uml.org.cn