专题 OS与Java多线程编程 (2学时)

主讲教师：张春元

联系电话：13876004640

联系地址：信息学院220室

课程邮箱：haidaos@126.com

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

1 反思：我们从OS课程学习到了什么？

❖ OS课程的重要性

* 本专业的八门核心课程之一
* 本专业全国硕士研究生入学统考四门基础课之一
* 部分高校博士研究生入学考试两门基础课程之一
* 百度、阿里巴巴、腾讯校招笔试的两门课程之一

❖ OS定义回顾

* OS是计算机中最重要的系统软件，是一组控制和管理计算机软硬件资源、合理地对各类作业进行调度以及方便用户使用的程序集合。

1 反思：我们从OS课程学习到了什么？

❖ OS课程章节内容回顾

章节	重点内容
处理机管理(2-3章)	进程控制、进程同步、进程通信、处理机调度
存储器管理(4-5章)	内存分配、地址变换、 内存扩充
设备管理(6章)	I/O系统、设备控制、设备分配、缓冲管理、 磁盘调度
文件管理(7-8章)	文件系统、目录管理、文件共享、文件保护、磁盘管理
OS接口(9章)	用户接口、程序接口

1 反思：我们从OS课程学习到了什么？

- 我们到底学得怎样
  - 我们会做作业、会考试
  - 我们会模拟实现进程调度算法
  - 我们会模拟实现银行家算法、死锁检测算法
  - 我们会模拟实现多达6种页面置换算法
  - .....
  - ???

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

2 并发、进程与线程基本概念

❖ 并发

* 程序的并发执行：是指两个或两个以上的程序或程序段可在同一时间间隔内同时执行。
* 程序并发执行的特征
  - 1> 间断性：走走停停，一个程序可能走到中途停下来，失去原有的时序关系。
  - 2> 失去封闭性：程序并发执行时，系统中多道程序共享资源，资源的状态不是唯一地取决于某一个程序，因此，必然失去了程序的封闭性，而程序的执行结果因依赖于外部环境也失去了可再现性。
  - 3> 不可再现性： 失去封闭性[illegible]不可再现性；外界环境可能在程序的两次执[illegible]变化，失去原有的可重复特征。

2 并发、进程与线程基本概念

❖ 进程

* 研发多道OS系统(OS/360、MULTICS)时引入
* 进程的定义
  - 1> 进程是程序的一次执行。
  - 2> 进程是一个程序及其数据在处理机上顺序执行时所发生的活动。
  - 3> 进程是程序在一个数据集合上运行的过程，它是系统进行资源分配和调度的一个独立单位。
  - 4> 在引入了进程实体的概念后，传统OS中的进程还可定义为：进程是进程实体的运行过程，是系统进行资源分配和调度的一个独立单位。

2 并发、进程与线程基本概念

* 进程的状态切换

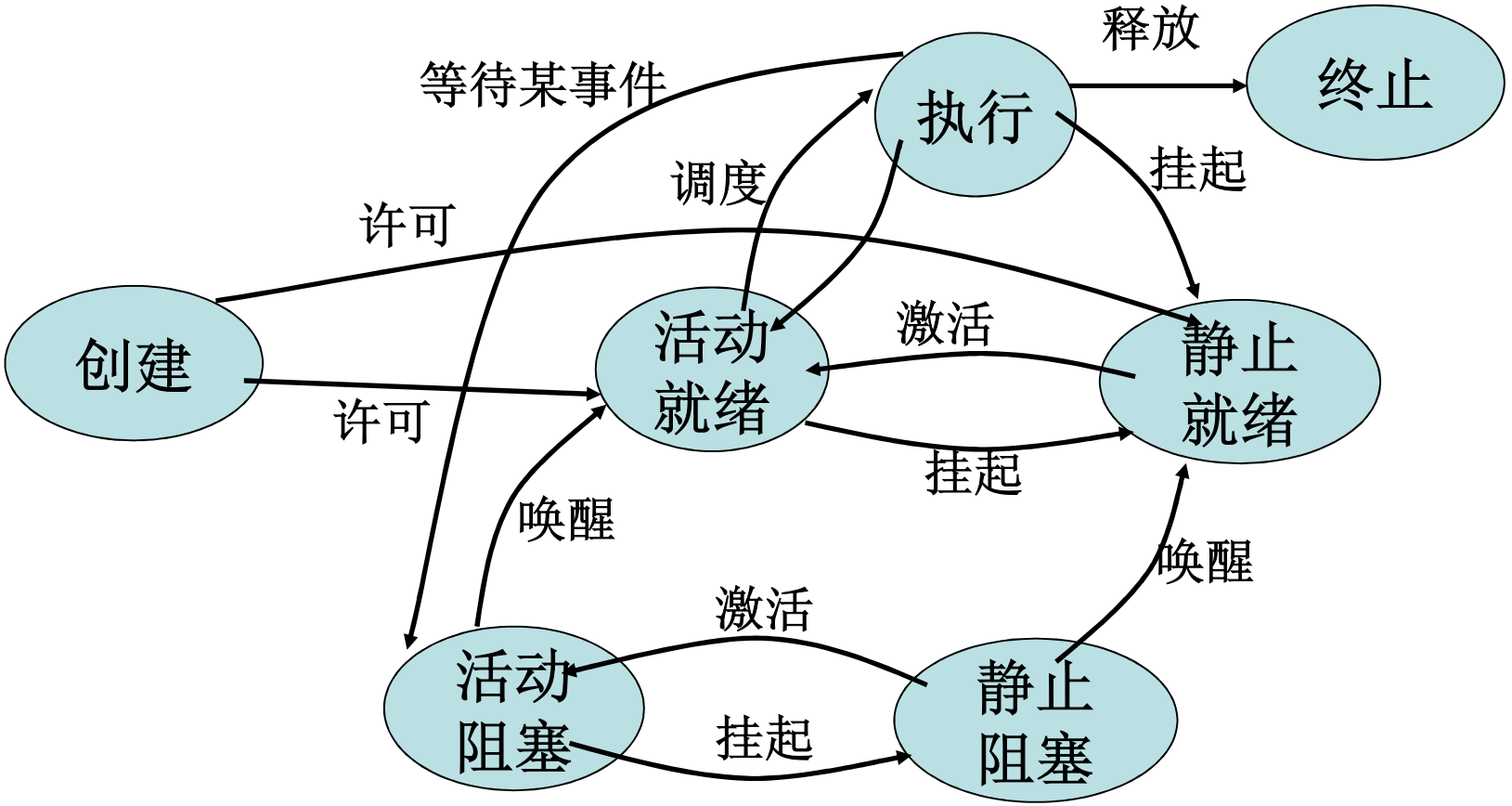


图　UNIX的进程状态转换

2 并发、进程与线程基本概念

* 进程同步
  - 进程同步的定义：对多个相关的进程在执行次序上进行协调，以使并发执行的诸进程之间能有效地共享资源和相互合作，使程序的执行具有可再现性。
  - 进程同步的实现
    - 硬件同步机制
    - 信号量机制
    - 管程机制

❖ 线程

* 引入缘由

- 传统OS中进程是一个可拥有资源的独立单位，同时又是一个可独立调度和分派的基本单位。由于进程是一个资源的拥有者，因而在创建、撤消和切换中，系统必须为之付出较大的时空开销。

* 线程的定义

- 在多线程OS中，线程是能独立运行的基本单位，基本上不拥有系统资源(只有一点必不可少的、能保证其独立运行的资源)，但共享其所属进程所拥有的全部资源。

2 并发、进程与线程基本概念

* 线程的实现方式

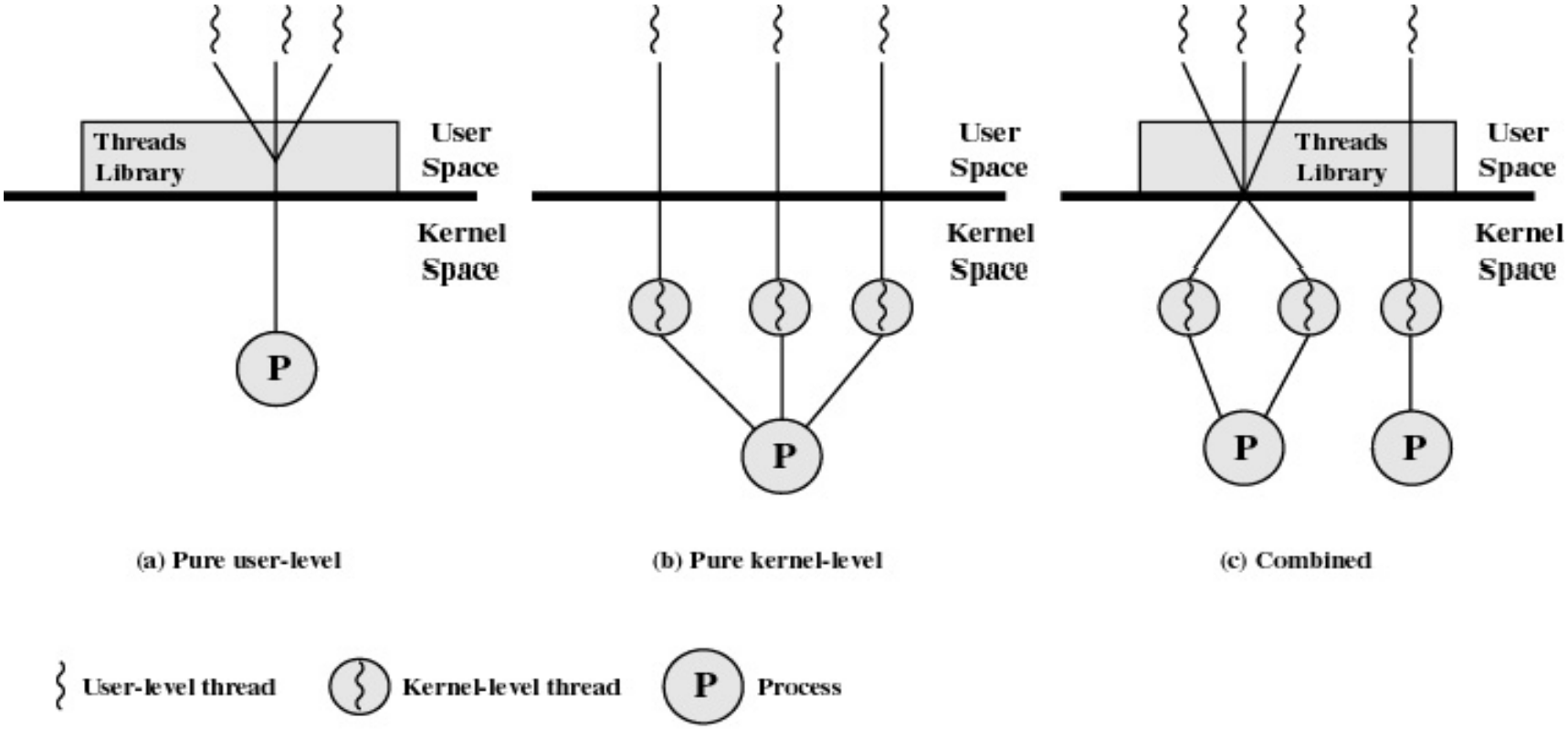


图 线程实现的三种方式

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

3 Java线程与拟运行任务对象的创建

❖ Java线程对象创建与启动

* 线程对象的创建
  - Thread 线程对象名 = new Thread(任务对象名);
* 线程对象的启动
  - 线程对象名.start( );

    //启动该线程对象进入就绪态，此后JVM将调度该线程对象来执行任务对象中的run( )方法

3 Java线程与拟运行任务对象的创建

❖ Java线程拟运行任务对象的创建

* 在Java中，每个任务对象都是 Runnable接口的一个实例，也称为可运行对象。一个任务对象必须在线程中执行。

* 任务类的定义

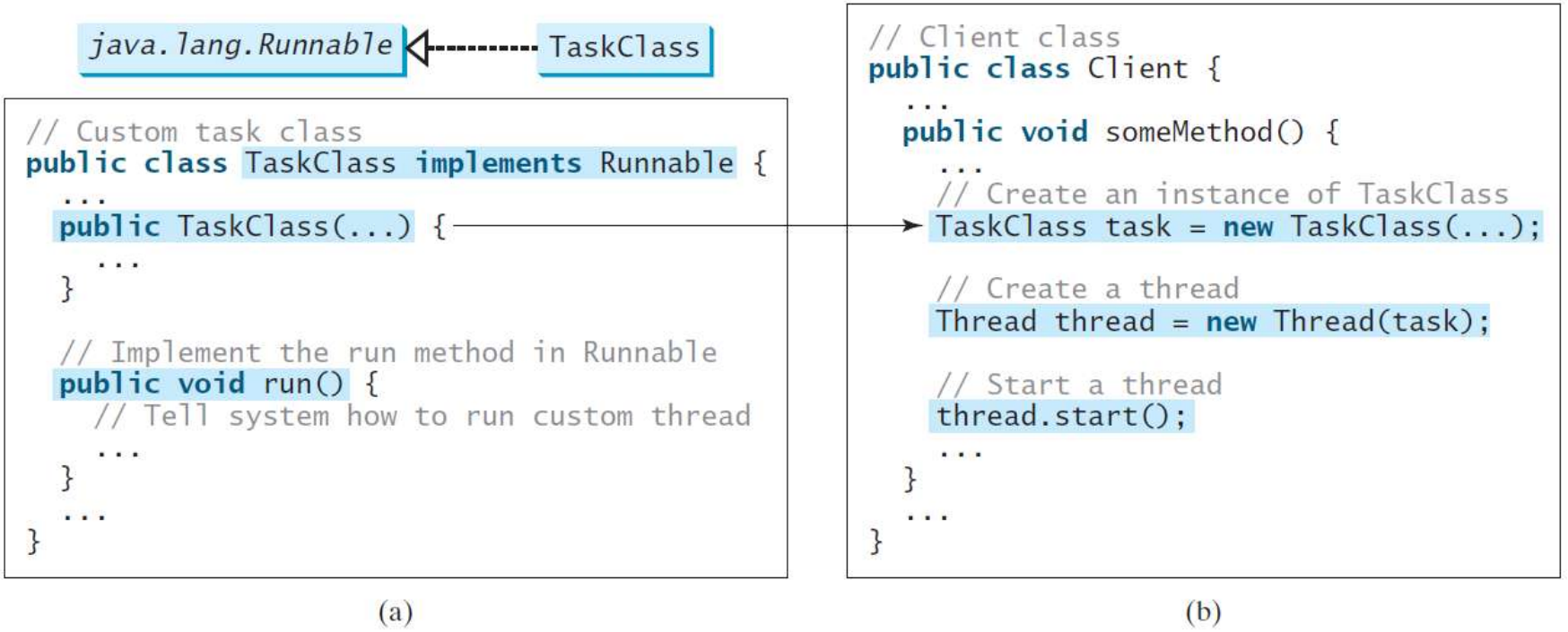
```
class 任务类名 implements Runnable{
    public void run( ){
        … 告诉线程如何完成此任务
    }
}
```

* 任务对象的创建

  - 任务类名 任务对象名 = new 任务类名(…);

3 Java线程与拟运行任务对象的创建

❖ 代码演示



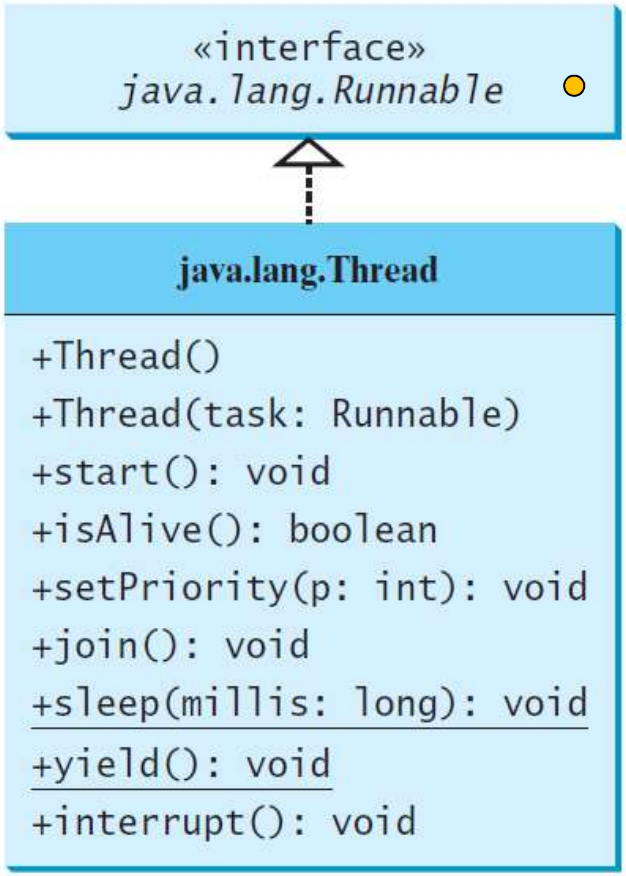
* 应用示例： TaskThreadDemo.java 创建三个线程，分别打印字母a、b各100次和数字1至100

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

4 Java线程类和线程对象的状态

❖ Thread的UML图



意味着也可以通过扩展Thread类来定义一个新的类，使其既是任务类又是线程类。

Creates an empty thread.	
Creates a thread for a specified task.	
Starts the thread that causes the `run()` method to be invoked by the JVM.	
确定该线程是否活着（当处于就绪、执行或阻塞态，均返回true）	
Sets priority p (ranging from 1 to 10) for this thread. **数值越大级别越高**	
Waits for this thread to finish.	
Puts a thread to sleep for a specified time in milliseconds.	
Causes a thread to pause temporarily and allow other threads to execute.	
Interrupts this thread.	

4 Java线程类和线程对象的状态

❖ Java线程对象的状态

* 和传统OS中进程具有多个状态一样，Java中的线程对象在整个生命周期也具有五个状态：
  - 新建态（只是创建，尚未启动）
  - 就绪态（已启动但尚未分配到CPU资源
  - 执行态（已获得CPU资源，真正执行）
  - 阻塞态（因等待某事件发生而主动交出CPU停止运行，在该事件发生前即使把处理机分配给该进程，线程也无法运行）
  - 完成态（整个任务执行完毕）

4 Java线程类和线程对象的状态

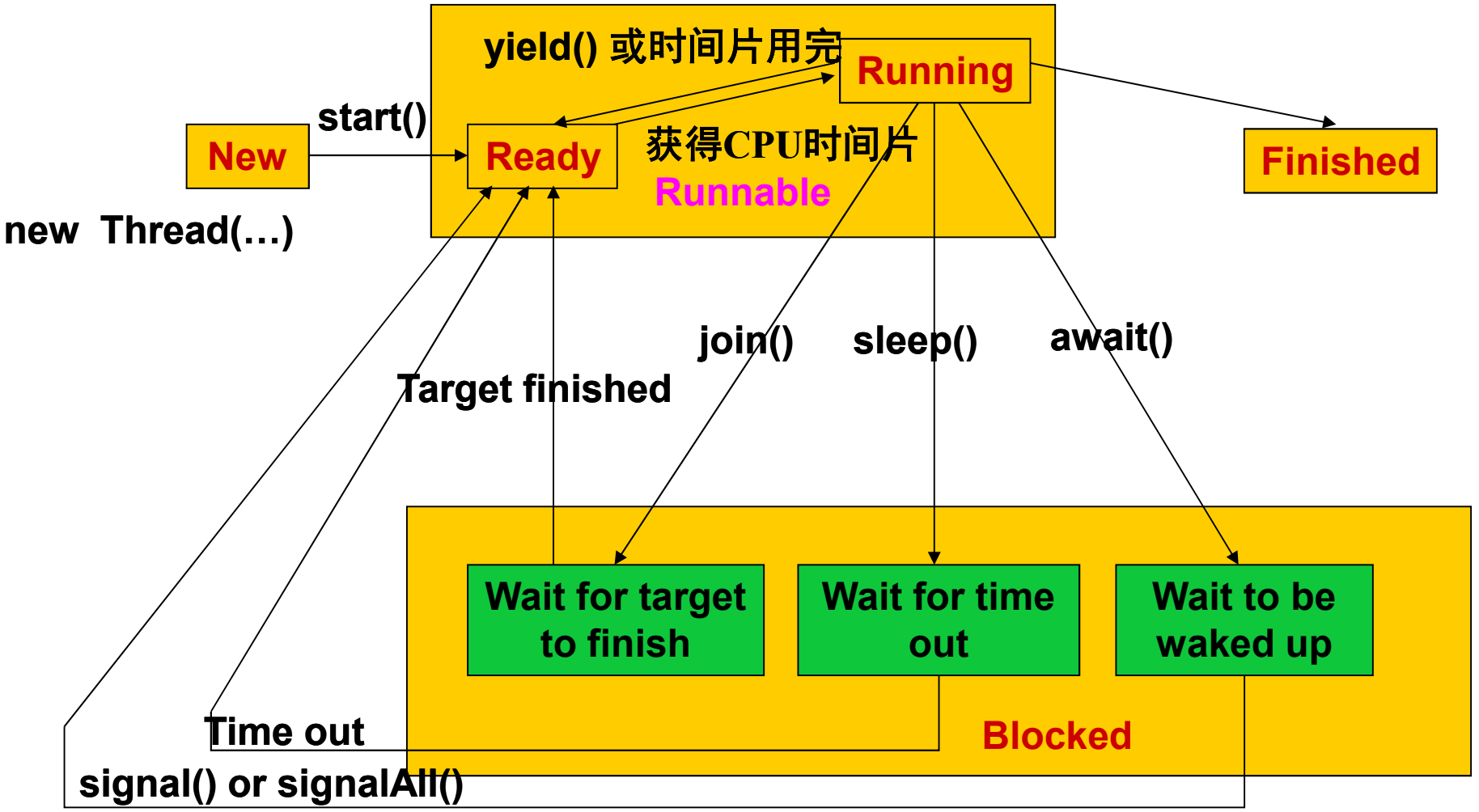


图 Java线程对象的五种状态切换

4 Java线程类和线程对象的状态

❖ Java线程对象的状态切换方法说明

* **yield( )**
  - 当前线程对象暂停执行，主动交出CPU让其它线对象执行，其状态由执行态转至就绪态。
* **sleep(…) throws InterruptedException**
  - 当前线程主动交出CPU让其它线程执行，其状态由执行态转至阻塞态，待指定休眠期满后再自动转至就绪态。
* **join( ) throws InterruptedException**
  - 如果当前线程对象a所执行的任务的run( )方法存在另一线程对象调用join()方法的语句，当a执行到此条语句时，a即进入阻塞态，直到b结束（进入完成态）a才重新就绪态。

4 Java线程类和线程对象的状态

- join( ) 应用示例

```
public void run() {
  Thread thread4 = new Thread(
    new PrintChar('c', 40));
  thread4.start();
  try {
    for (int i = 1; i <= lastNum; i++) {
      System.out.print(" " + i);
      if (i == 50) thread4.join();
    }
  }
  catch (InterruptedException ex) {
  }
```

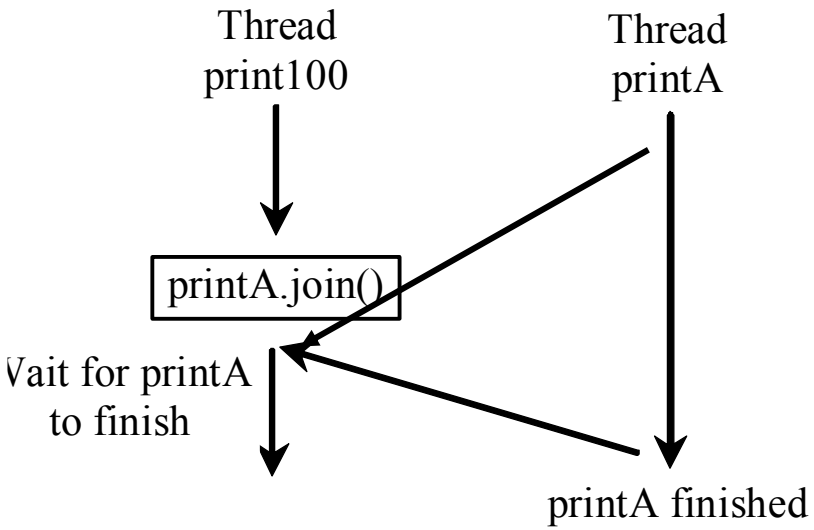


图　当前线程所执行的任务的run( )方法

4 Java线程类和线程对象的状态

* await( )、signal( )、signalAll( )
  - 通常和锁配合使用，本专题第6节再详细讲授
* stop( ) 、suspend( )、resume( )
  - 以上三个方法由于存在不安全因素，已经废弃，不再建议使用

* 应用示例： FlashText.java 创建一个线程，每隔200毫秒修改一下标签上的文字，以实现文本闪烁

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

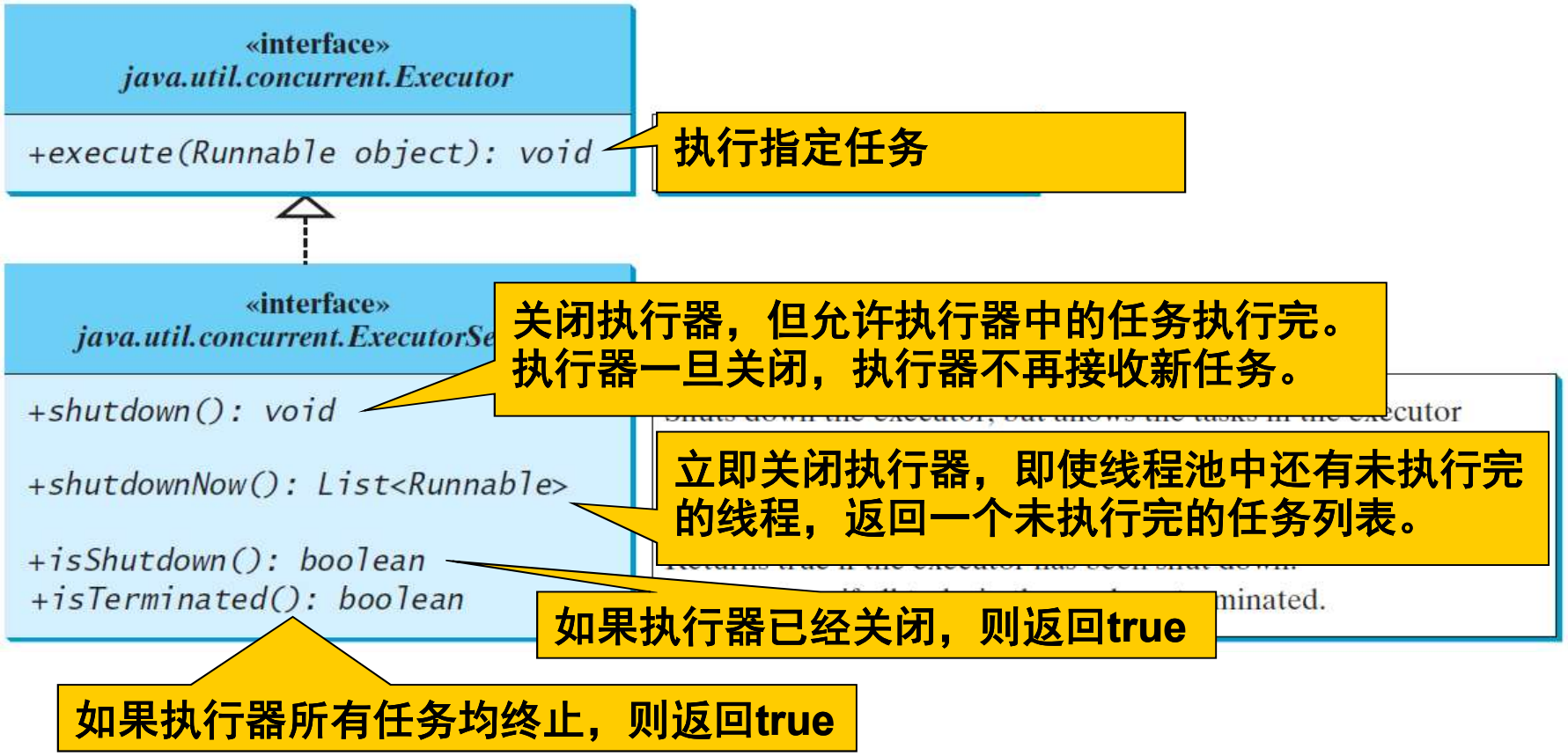
5 Java线程池与线程池类

❖ 为什么要引入线程池

* 在前面的例子中，我们为每一任务均创建一个线程，当任务数量非常多时，则需创建非常多的线程，会导致程序的性能下降。为此，可创建一个包含一定数量线程的线程缓存池，如果其中某个线程完成了一个任务的执行，可以给其再行分配一个新的任务让其接着执行，从而避免了创建过多的线程。
* 适用场合：通常用于执行相似的任务

5 Java线程池与线程池类

❖ 线程池类Executors所实现的接口



5 Java线程池与线程池类

❖ 线程池类Executors自定义的方法

java.util.concurrent.Executors

+newFixedThreadPool(numberOfThreads: int): ExecutorService

+newCachedThreadPool(): ExecutorService

创建一个可并行运行指定数目的线程池。当池中的一个线程执行完任务后，可接着用来执行另一个新的任务。

concurrently. A thread may be reused to execute another task after its current task is finished.

eeded, but ey are

创建一个线程池，它会在必要时创建新的线程以满足等待执行的任务，但如果之前创建的线程可用，则先重用之前创建的线程。

* 应用示例： ExecutorDemo.java 创建一个拥有两个线程的线程池，分别打印字母a、b各100次和数字1至100

本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

6 Java多线程互斥

❖一个例子

* AccountWithoutSync1.java 创建一个拥有100个线程的线程池，每个线程每次朝同一帐户存入一个便士，总共存100万次，最后一共有多少钱？
  - 问题原因：此程序中帐户的操作语句块是临界区（帐户是临界资源），不能让多个线程同时进入

❖多线程互斥

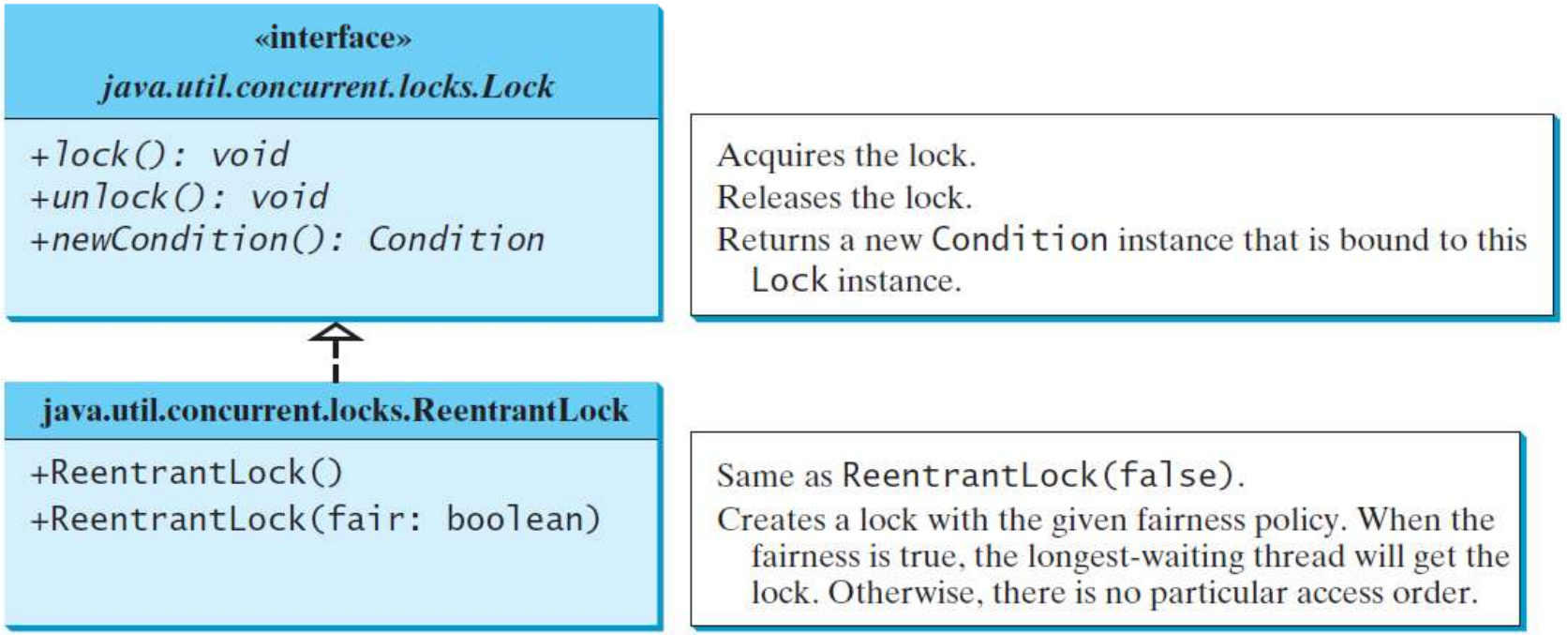
* 当程序中存在临界资源时，为了使程序的执行具有可再现性，任一时间只能让一个线程对该资源进行操作即多个线程必须互斥地进入临界区

6 Java多线程互斥

- 多线程互斥方法1：添加synchronized关键字
  - 添加办法1：在含有临界区语句块的方法头中加上synchronized关键字
    - 应用示例： AccountWithSync1.java 将原程序中的public void deposit(int account)方法头改成了public synchronized void deposit(int amount)
  - 添加办法2：对临界区语句块附加synchronized
    - 应用示例： AccountWithSync2.java 或 AccountWithSync3.java

❖ 多线程互斥方法2：加锁互斥

* 采用ReentrantLock类中lock( )和unlock( )方法对临界区语句块先加锁后解锁
* 应用示例：AccountWithSyncUsingLock.java



本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

7 Java多线程协作

❖ 线程间协作

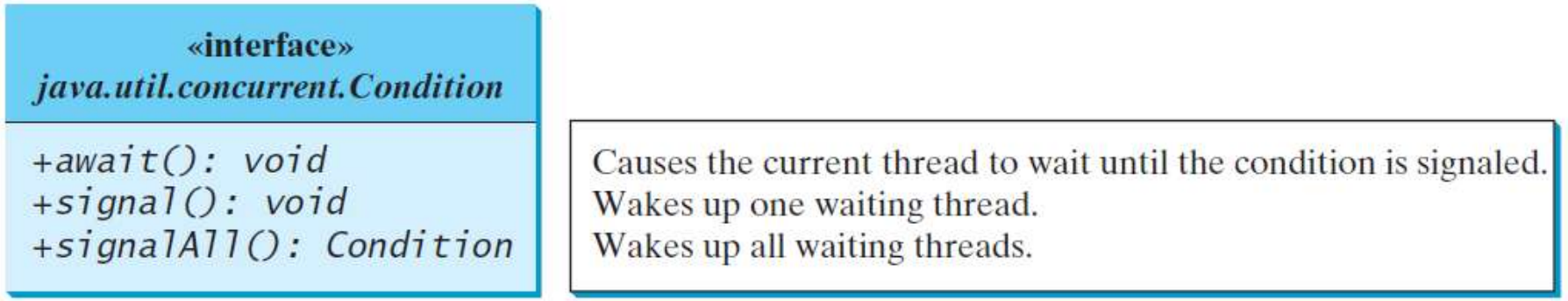
* 当一个程序中的多个线程存在依赖关系时，必须协调线程间的执行次序来让线程彼此协作来访问共享的资源，才能让程序得到正确的结果。

❖ 多线程协作实现办法

* 首先通过锁对象建立一个锁条件对象

  Condition 条件名 = 锁对象名.newCondition( );

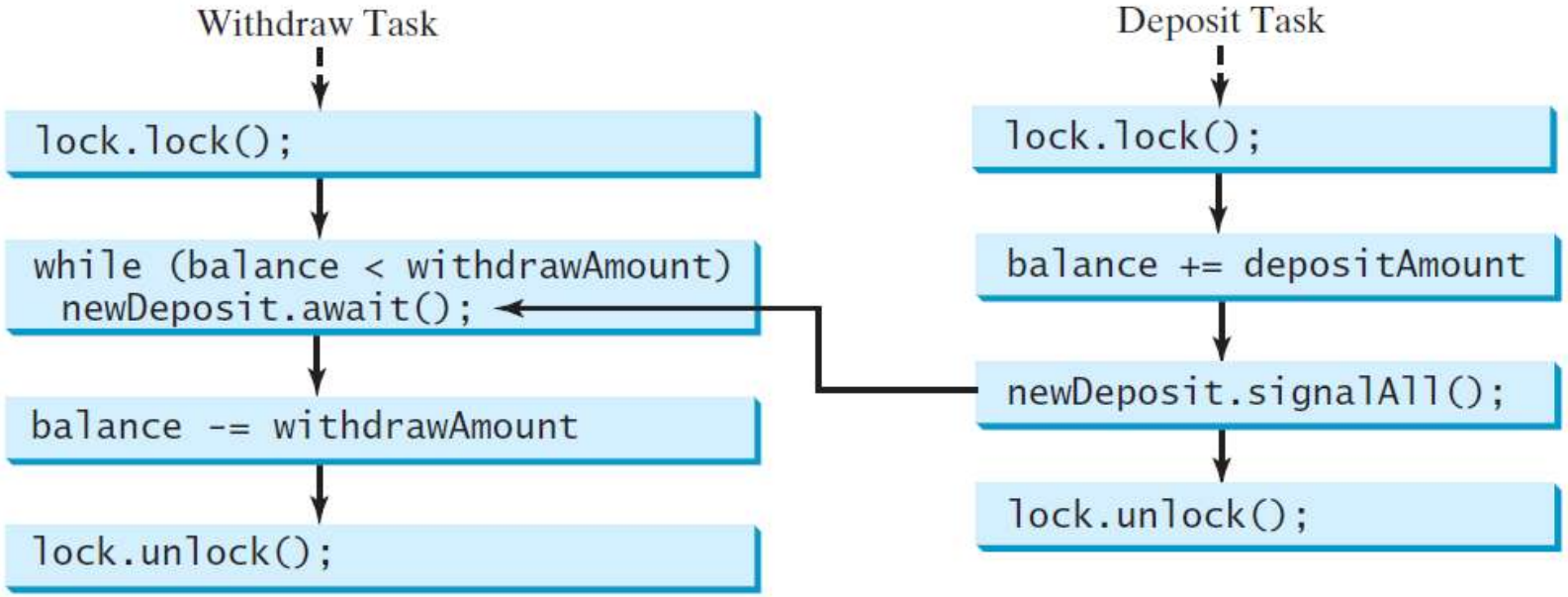
* 然后调用锁条件提供的await( )、signal( )和signalAll( )方法对多线程进行协调



- 这三个方法的使用前，都需要事先加锁。
- await( )方法：让当前线程对象进入阻塞态并且自动释放条件上的锁。
- signal( )方法：当前线程对象唤醒一个被阻塞在条件对象上的线程对象并让其重新加锁。

❖ 代码演示

* 应用示例：ThreadCooperation.java 创建一个拥有两个线程的线程池，这两个线程共享一个帐户，其中一个线程朝这个帐户里存钱，另一个线程从这个帐户里取钱。



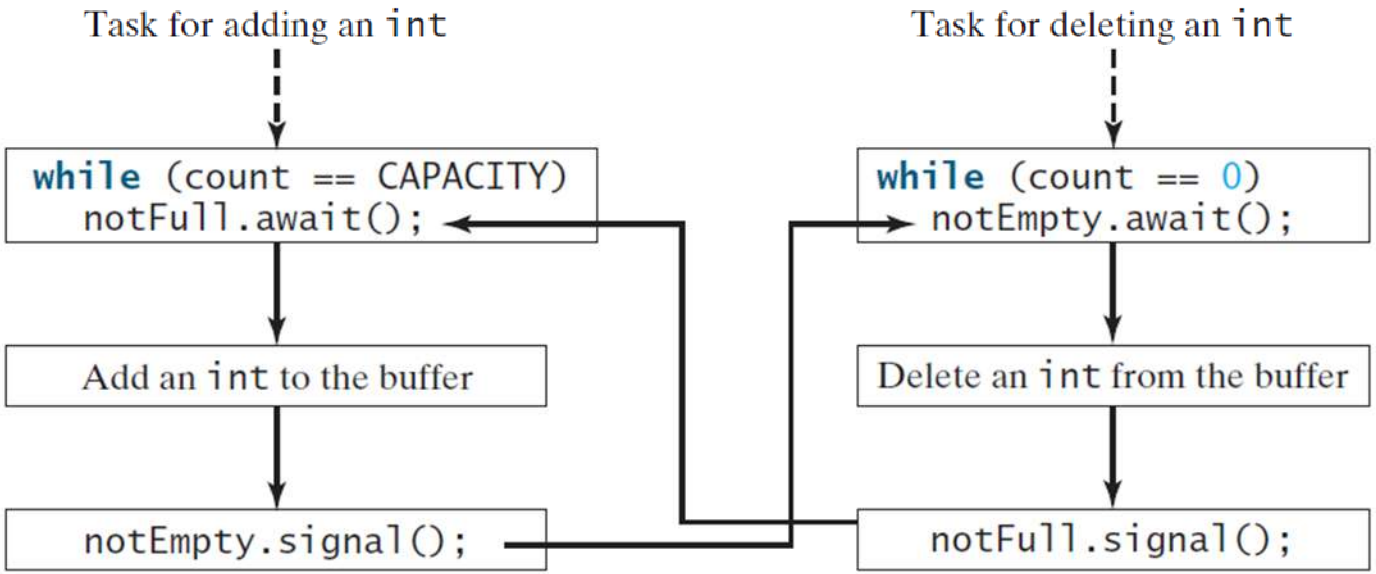
* 应用示例：生产者-消费者问题

```
var mutex，empty，full：semaphore：=1，n，0；
buffer：array[0，1，…，n-1] of item；
in，out：integer：=0，0；
```

```
producer (i)
    repeat
        生产一个产品=>nextp;
        wait(empty);
        wait(mutex);
        buffer(in):=nextp;
        in:=(in+1)mod n;
        signal(mutex);
        signal(full);
    until false;
```

```
consumer (j)
    repeat
        wait(full);
        wait(mutex);
        nextc: =buffer(out);
        out:=(out+1) mod n;
        signal(mutex);
        signal(empty);
        消费掉产品nextc;
    until false;
```

- 代码：ConsumerProducer.java 创建一个拥有两个线程的线程池，共享一个缓冲池（最大容量为5），其中一个线程朝这个缓冲池里写数据，另一个线程从这个缓冲池里取数据。

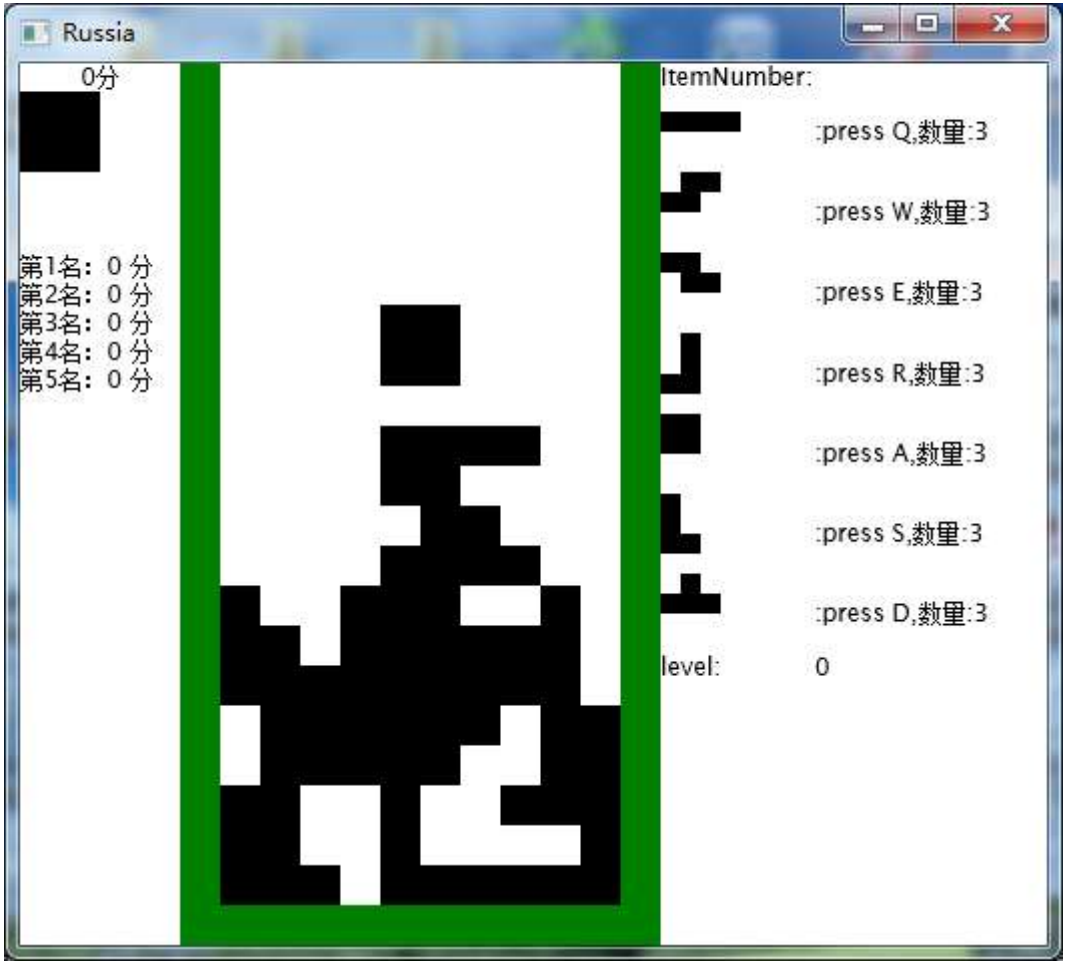


本专题主要内容

- 1 反思：我们从OS课程学习到了什么？
- 2 并发、进程与线程基本概念
- 3 Java线程与拟运行任务对象创建
- 4 Java线程类和线程对象的状态
- 5 Java线程池与线程池类
- 6 Java多线程互斥
- 7 Java多线程协作
- 8 实例分析：俄罗斯方块小游戏

8 实例分析：俄罗斯方块小游戏

- 李昊伦、李金波俄罗斯方块小游戏



本专题结束！祝大家期终考试顺利！

前程似锦！谢谢大家！