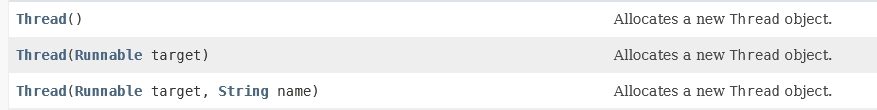
1. 买票程序进阶，
   1. 企业中我们常常为了保持类的干净和纯粹，并不会在资源类上实现Runnable，因此我们需要用到Thread（new Runnable ，String name）的有参构造。



|  |
| --- |
| public class ReviewTicket {  public static void main(String[] args) {  Tickets tickets = new Tickets() ;  new Thread(new Runnable() {  @Override  public void run() {  tickets.saleTickets();  }  },"售票员A").start();  new Thread(new Runnable() {  @Override  public void run() {  tickets.saleTickets();  }  },"售票员B").start();  new Thread(new Runnable() {  @Override  public void run() {  tickets.saleTickets();  }  },"售票员C") ;  }  } |

同时使用匿名内部类进行Runnable结构的实例化，我们知道接口是不能直接new的，因此需要依靠匿名内部类进行接口的new ；

资源类:

|  |
| --- |
| class Tickets{  private int num = 30;  public synchronized void saleTickets(){  for (int i = 0; i < 30; i++) {  System.out.println(Thread.currentThread().getName()+  "\t卖出第:"+num-- + "\t还剩:"+num);  }  }  } |

在资源类中，通常我们会在saleTickets方法上加上synchronized ，而在实际开发中，我们常常使用java.util.concurrent.locks包中的方法，来实现一段代码的同步问题；

如下:

|  |
| --- |
| class Tickets{  private int num = 30;  private Lock lock = new ReentrantLock();  public void saleTickets(){  lock.lock();  try {  if(num > 0) {  System.out.println(Thread.currentThread().getName()+  "\t卖出第:"+num-- + "\t还剩:"+num);  }  }finally {  lock.unlock();  }  }  } |

但是我们通过观察可以看到 在new Thread （）中有大量冗余代码，而Runnable是一个函数式接口:

@FunctionalInterface

public interface Runnable

因此我们可以使用Lambda表达式进行简化。

总体代码：

|  |
| --- |
| package com.company.多线程;  import java.util.concurrent.locks.Lock;  import java.util.concurrent.locks.ReentrantLock;  class Tickets{  private int num = 30;  private Lock lock = new ReentrantLock();  public void saleTickets(){  lock.lock();  try {  if(num > 0) {  System.out.println(Thread.currentThread().getName()+  "\t卖出第:"+num-- + "\t还剩:"+num);  }  }finally {  lock.unlock();  }  }  }  public class ReviewTicket {  public static void main(String[] args) {  Tickets tickets = new Tickets() ;  new Thread(() -> {  for (int i = 0; i < 30; i++) {  tickets.saleTickets();  }  },"售票员A").start();  new Thread(() -> {  for (int i = 0; i < 30; i++) {  tickets.saleTickets();  }  },"售票员B").start();  new Thread(() -> {  for (int i = 0; i < 30; i++) {  tickets.saleTickets();  }  },"售票员C").start();  }  } |