Java学习day11

1. 多线程（概述）：

进程：是一个正在执行的程序。

每一个进程都有一个执行顺序，该顺序是一个执行路径，或者叫一个控制单元。

线程： 就是进程中的一个独立的控制单元。 线程控制着进程的执行。

一个进程中至少有一个线程。

java VM 启动的时候会有一个进程java.exe。

该进程中至少一个线程负责java程序的运行。

而且这个线程运行的代码存在于main方法中。

该线程称之为主线程。

扩展：更细节的jvm，jvm启动不止一个线程，还有负责垃圾回收的线程



2.进程：是一个正在执行的程序。

每一个进程都有一个执行顺序，该顺序是一个执行路径，或者叫一个控制单元。

线程： 就是进程中的一个独立的控制单元。 线程控制着进程的执行。

一个进程中至少有一个线程。

java VM 启动的时候会有一个进程java.exe。

该进程中至少一个线程负责java程序的运行。

而且这个线程运行的代码存在于main方法中。

该线程称之为主线程。

扩展：更细节的jvm，jvm启动不止一个线程，还有负责垃圾回收的线程

创建线程的第一种方式：继承Thread类：

步骤： 1).继承Thread类；

2).重写run方法；

3).调用start方法

该方法有两个方法：1.启动线程；2.调用run()方法。

我们可以看到每次打印的结果都是不同的，因为多线程的关系

多个线程都能获取cpu的执行权，哪个线程在运行由cpu来决定。

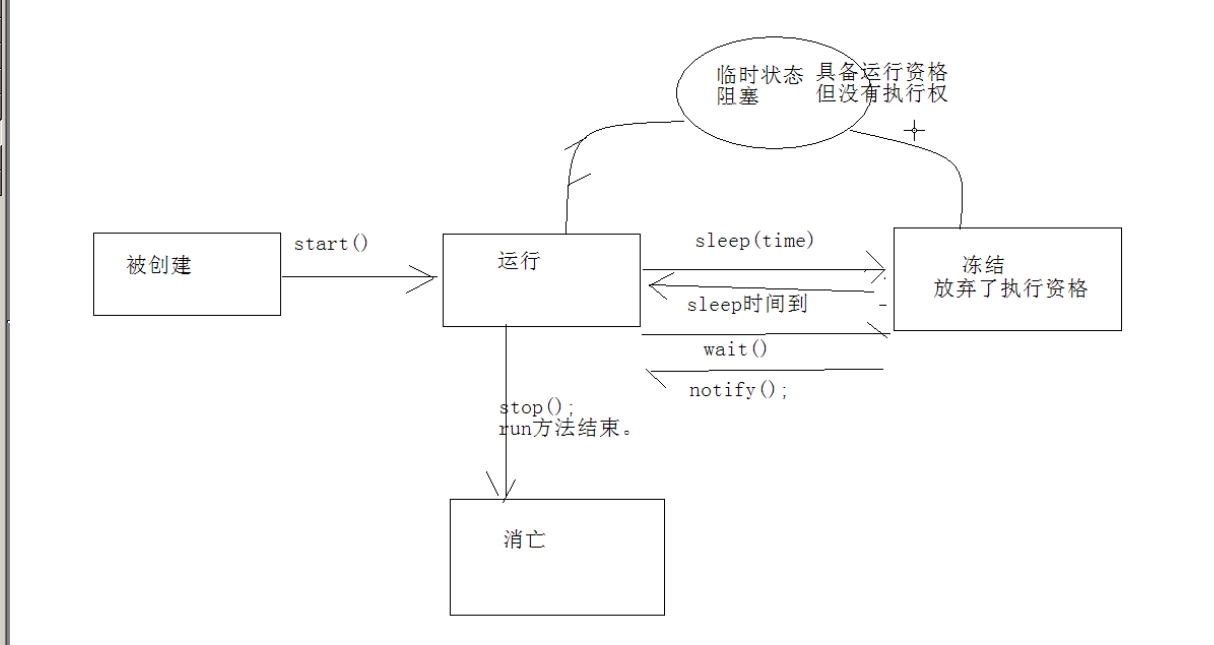
形象地理解cpu的运行其实是在抢夺cpu的使用权。

因此多线程的运行有一个特点：随机性。 谁抢到谁说的算，至于到底谁能抢到由cpu说的算。

3.多线程(创建线程-run和start的特点)。

1. 多线程的练习：创建两个线程，和主线程交替运行。
2. 线程(多线程运行状态)：

当一个线程被创建，并且使用cpu的控制权时，其他线程进入临时状态阻塞。被cpu选中的状态使用start函数能够运行，并且经过sleep或者wait进入冻结，notify()唤醒后进入可能进入运行又或者是临时状态阻塞。运行结束后，经过stop()，run方法结束，此线程可以消亡。

1. 获取线程对象以及名称

currentThread()--static Thread 静态线程。获取当前线程正在执行的引用,

Thread.currentThread() 相当于this。

因为是静态数据，所以用类名就可以直接访问。

getName() 返回该线程的名称.

1. 多线程（售票的例子）：学习到了创建线程，实现Runnable的例子。

理解synchronized这一个关键字！！！

多线程的第二种算法： 接口Runnable

步骤：

1.定义一个类名实现Runnable接口

2.重写其中的run方法

该线程要运行的代码存放在run方法之中

3.通过建立多个Thread类建立多个线程

4.通过将Runnable的子类实例化后的参数传给Thread类中构造函数

为什么要将Runnable建立的子类对象传递给Thread类的构造函数？

因为自定义的run方法所属的对象是Runnable接口的子类对象。

所以要让线程去执行指定对象的run方法，就必须明确该方法所述的对象

8.通过Thread类的start方法启动线程，并且调用的是Runnable中的run方法。

实现方式和继承方式有什么区别？

继承方式是有单继承的局限性，在定义线程的时候，用实现的方式比较合理。

两种方式的区别：

1.继承方式的代码存放于Thread子类run方法中；

2.实现方式存放在Runnable接口的子类的run方法中。

9.多线程的安全问题：

需求： 多线程所存在的安全问题。

通过分析：发现数据中会出现小于0的负数，那是因为在执行代码的时候，启动线程，但是当这个线程还没有运行完的时候，另一个线程开始执行语句，导致共享数据的错误。

Thread类中的sleep用来模拟休眠。因为要抛出异常，所以用

解决方法：通过synchronized关键字，实现数据之间的同步。

synchronized(对象)

{

要同步的代码;

}

1. 多线程-同步函数

需求：银行有一个金库，有两个储户分别存300元，每次存100元，有3次。

问 该程序是否有安全问题？如果有，该怎么解决？

如何找问题？

1.看一下哪些代码是多线程运行的代码;

2.看一下哪些数据是共享的数据;

3.看一下多线程运行的代码中哪些是在操作共享数据的。

发现存的钱都是乱的。

同步函数：同步这个关键字可以作为修饰符修饰函数。

11.用小程序来证明同步函数用的锁是this。

使用两个线程来买票。

一个线程在同步代码块中，另一个线程在同步函数中。

都在执行买票的动作。

同步的两个前提：

1. )两个或两个以上的线程；
2. )多线程用的是同一个锁。

用小程序来证明静态同步函数用的锁是class。

因为静态函数运行时，还没有创建本类的对象，所以不能使用this。

但是可以用到的是所属代码中的本类的字节码文件 类名.class

在使用synchronized(类名)后，发现同步，也不会造成安全问题。

12.多线程-单例设计模式

多线程-单例设计模式-懒汉式

什么是单例设计模式： 一个类保证只有一个实例; 创建的方法分为饿汉式与懒汉式。

饿汉式： 先把对象new出来实例化;

懒汉式： 把对象指为空，然后经过判断是否为空来new，具有延时加载的特性。

饿汉式：

class Single

{

private static final Single s = new Single();

private Single(){}

private void getSingle()

{

return s;

}

}

下面我写一个懒汉式：

因为懒汉式有点麻烦，所以搞开发的时候一般用饿汉式， 但是面试题目的时候还是懒汉式多。

问出来的问题：懒汉式和饿汉式有什么不同？懒汉式存在什么问题？懒汉式加同步的时候使用的锁是哪一个？

懒汉式具有延时加载的特性。懒汉式用双重否定的格式提高效率。 使用的锁是该类所属的字节码文件对象。

13.写一个死锁的例子 A中有B，B中有A。

存在一些问题不知道怎么解决，就是最后运行的时候不会发生死锁的情况