Java语言基础

【Day16】

线程

线程的基本概念

- Java语言的优势之一就是线程处理较为简单。
- 一般操作系统都支持同时运行多个任务，一个任务通常就是一个程序，每个运行中的程序被称为一个进程，当一个程序运行时，内部可能包含多个顺序执行流，每个顺序执行流就是一个线程。

Thread类的简介

- java.lang.Thread类代表线程，任何线程对象都是Thread类（子类）的实例
- Thread类是线程的模板，封装了复杂的线程开启等操作，封装了操作系统的差异性。

线程的创建方式一

- 创建一个具体线程, 需要继承于Thread类。
- 覆盖run 方法(就是更新运行过程), 实现用户指定任务的过程。
- 创建线程实例(一个线程)。
- 使用线程实例调用start方法启动线程, 该线程会尽快的去并发执行run方法。

线程的创建方式二

- 创建一个类实现Runnable接口并重写run方法。
- 创建实现Runnable接口类的实例对象作为参数创建Thread对象。
- 使用Thread类对象调用start方法启动线程, 该线程会尽快并发执行run方法

线程的编号和名称

- 线程编号和名称的相关方法如下：

long getId()	用于获取调用对象所表示线程的编号
String getName()	用于获取调用对象所表示线程的名称
void setName(String name)	用于设置线程的名称为参数指定的数值
static Thread currentThread()	获取当前正在执行线程的引用

Thread类的常用方法

- Thread类的常用方法如下：

static void yield()	当前线程让出处理器（离开Running状态），使当前线程进入Runnable状态等待
static void sleep(times)	使当前线程从 Running 放弃处理器进入Block状态, 休眠times毫秒, 再返回到Runnable如果其他线程打断当前线程的Block(sleep), 就会发生InterruptedException。

Thread类的常用方法

- Thread类的常用方法如下：

void join()	等待该线程终止
void join(long millis)	表示等待参数指定的毫秒数
boolean isDaemon()	用于判断是否为守护线程
void setDaemon(boolean on)	用于设置线程为守护线程

线程同步机制

线程同步的概念

- 多个线程并发读写同一个临界资源时会发生线程并发安全问题。
- 常见的临界资源:
  1 多线程共享实例变量
  2 多线程共享静态公共变量
- 若想解决线程安全问题，需要将异步的操作变为同步操作。

异步操作:多线程并发的操作，各自独立运行。

同步操作:多线程串行的操作，先后执行的顺序。

线程同步的实现

- Java提供了一种内置的锁机制来支持原子性，使用synchronized关键字。
- 第一种实现方式如下：

使用同步代码块的方式实现部分代码的锁定，格式如下：

```
synchronized(类类型的引用){
    编写所有需要锁定的代码;
}
```

线程同步的实现

- 第二种实现方式如下：

  使用同步方法的方式实现所有代码的锁定。

  直接使用synchronized关键字来修饰整个方法即可

  该方式等价于:

  synchronized(this){ 整个方法体的代码 }

实现的注意方式

- 使用synchronized保证线程同步应当注意:

  1 多个需要同步的线程在访问该同步块时，看到的应该是同一个所对象引用。

  2 在使用同步块时应当尽量减少同步范围以提高并发的执行效率。

线程安全类和线程不安全类

- StringBuffer 是同步的 synchronized append();
- StringBuilder 不是同步的 append();
- Vector 和 Hashtable 是同步的
- ArrayList 和 HashMap 不是同步的
- Collections.synchronizedList()
- Collections.synchronizedMap()

静态方法的锁定

- 当我们对一个静态方法加锁，如:

  public synchronized static void xxx(){….}

- 那么该方法锁的对象是类对象。每个类都有唯一的一个类对象。获取类对象的方式:类名.class。
- 静态方法与非静态方法同时声明了synchronized，他们之间是非互斥关系的。原因在于，静态方法锁的是类对象而非静态方法锁的是当前方法所属对象

死锁的概念

- 当两个线程或多个线程之间互相锁定时就形成了死锁。
- 避免死锁的原则：

  顺序上锁，反向解锁，不要回头

总结与答疑

技术资料

IT兄弟连

ITXDL.CN

变态严管　让学习成为一种习惯