目录

[1、 String 2](#_Toc14316)

[1.1、 String() 2](#_Toc1460)

[1.2、 String(byte bytes[], int offset, int length) 2](#_Toc9171)

[1.3、 int compareTo(String anotherString) 2](#_Toc25599)

[2、 StringBuilder 2](#_Toc270)

[2.1、 StringBuilder() 2](#_Toc18840)

[3、 Integer 3](#_Toc10121)

[4、 Thread 3](#_Toc25339)

[4.1、 Thread()、Thread(Runnable target) 3](#_Toc32005)

[4.2、 synchronized void start() 4](#_Toc9526)

[4.3、 final void join() 4](#_Toc30563)

[4.4、 4](#_Toc22410)

## String

内部维护了一个最终字符数组。

### String()

方法作用：无参构造，初始化新创建的 String对象，使其表示空字符序列。

### String(byte bytes[], int offset, int length)

方法作用：通过使用平台的默认字符集解码指定的字节子阵列来构造新的 String 。

①检查需要使用到的传入字节数组byte的字集是否超过了byte的范围，如果是，则抛出字符串索引越界异常；

②将传入的byte的指定子集转化为字符数组，并赋值给成员常量字符数组value。

### int compareTo(String anotherString)

方法作用：按字典顺序比较两个字符串。

①获取本字符串和传入字符串中字符数组长度中较小的值，依次比较同一索引处字符是否相等，若不想等，直接返回两字符的差值（本字符串中的字符减传入字符串的字符），如果所有索引处的字符均相同，则直接返回本字符串的长度减传入字符串的长度。

## StringBuilder

从父类继承了一个char数组变量；

### StringBuilder()

方法做用：无参构造，初始化字符数组长度为16

## Integer

①内部维护了一个基本int类型的value，用于存储此整型代表的整数值；

②Integer的hashcode是value本身

③自动装箱（Integer i = 100）编译后变成Integer i = Integer.valueOf(100);

④自动拆箱（ Integer a = new Integer(128);int m = a;）编译后变为（Integer a = new Integer(128);int m = a.intValue();）；

⑤Integer 内部维护了一个静态内部类IntegerCache，在Integer 被首次加载时，会创建一个Integer cache[]数组，里面存储了从-128到127的Integer 对象，在直接给Integer 赋值时（示例：Integer i = 100），如果赋值数值在-128 <= i <= 127，则直接返回cache[]中缓存的对象，不会新创建对象；

## Thread

①创建线程时，没有指定线程名称的话，会生成默认线程名称"Thread-" + nextThreadNum()；nextThreadNum()是同步方法，不会出现线程安全问题，nextThreadNum()被调用一次，维护的静态的int型成员变量threadInitNumber就会自增长1；

②sleep()让线程睡眠，交出CPU，让CPU去执行其他的任务。sleep方法不会释放锁，也就是说如果当前线程持有对某个对象的锁，则即使调用sleep方法，其他线程也无法访问这个对象。sleep方法相当于让线程进入阻塞状态。

③调用yield()方法会让当前线程交出CPU权限，让CPU去执行其他的线程。它跟sleep方法类似，同样不会释放锁。但是yield不能控制具体的交出CPU的时间，另外，yield方法只能让拥有相同优先级的线程有获取CPU执行时间的机会。注意，调用yield方法并不会让线程进入阻塞状态，而是让线程重回就绪状态，它只需要等待重新获取CPU执行时间，这一点是和sleep方法不一样的。

### Thread()、Thread(Runnable target)

构造方法

①按"Thread-" + nextThreadNum()；生成此线程的名称，nextThreadNum()是同步静态方法，获取静态的int型成员变量threadInitNumber的值；

②将第一步生成的名称赋值给成员变量name；

③获取当前正在执行线程的引用作为父线程；若是在main方法中创建的线程，则此时返回的就是主线程的引用；

④将父线程的线程组作为本线程的线程组，并将线程组的未启动线程数加一；

⑤如果父线程是守护线程则将本线程也是守护线程（将父线程的daemon布尔变量值赋值给本线程的daemon）；

⑥本线程的优先级也等于父线程的优先级；

⑦如果新建线程时传入了Runnable的实现，则将实现赋值到成员变量target中；

⑧将成员变量stackSize设置为0；

⑨调用nextThreadID()生成线程ID。

### synchronized void start()

方法做用：导致此线程开始执行; Java虚拟机调用此线程的run方法。

①检查线程状态threadStatus是否为0，不是，则抛出非法线程状态异常；

②通知组此线程即将start，以便可以将其添加到组的线程列表中并且可以减少组的unstarted线程的计数；

③调用本地方法start0()启动线程；

④如果线程启动报错，则将此线程从线程组中移出。

### final void join()

方法做用：假设在线程1中调用线程2的join()，线程1会被挂起，直到线程2死亡（执行完毕），线程1才会继续执行下去

①方法内部的原理就是一个while循环，执行条件是isAlive()方法（即线程2还活着），执行语句是wait()（及线程2的wait()方法）；其实就是在线程2死亡之前，线程1一直在执行这个while循环，直到线程2死亡，线程1 就跳出循环继续向下执行。

②如果在调用join()时传入了参数，单位为毫秒的时间值，则线程1只会被挂起这么长时间，无论此时线程2是否死亡，线程1都能跳出我还while循环，继续往下执行

### 4.4、