基本类知识点

Object类

1、什么是等待/通知机制?

一个形象的例子就是厨师和服务员之间就存在等待/通知机制。

1. 厨师做完一道菜的时间是不确定的，所以菜到服务员手中的时间是不确定的；
2. 服务员就需要去“等待（wait）”；
3. 厨师把菜做完之后，按一下铃，这里的按铃就是“通知（nofity）”；
4. 服务员听到铃声之后就知道菜做好了，他可以去端菜了。

用专业术语讲：

等待/通知机制，是指一个线程A调用了对象O的wait()方法进入等待状态，而另一个线程B调用了对象O的notify()/notifyAll()方法，线程A收到通知后退出等待队列，进入可运行状态，进而执行后续操作。上诉两个线程通过对象O来完成交互。

2、等待/通知机制的实现

Java为每个Object都实现了等待/通知（wait/notify）机制的相关方法，它们必须用在synchronized关键字同步的Object的临界区内。通过调用wait()方法可以使处于临界区内的线程进入等待状态，同时释放锁。而notify()方法可以唤醒一个因调用wait操作而处于阻塞状态中的线程，使其进入就绪状态。被重新唤醒的线程会试图重新获得临界区的锁，并继续执行wait方法之后的代码。如果发出notify操作时没有处于阻塞状态中的线程，那么该命令会被忽略。

`notify()`可以使调用对象上阻塞的某一个线程进入调度。该线程的选择决定于具体的调度方案，几乎相当于随机，不可对线程的选择做预先的猜测判断；而`notifyAll()`则是使当前对象的所有线程进入调度。

等待方：

1. 获得对象的锁
2. 如果条件不满足，则调用对象的wait()方法，被通知后仍然要检查是否满足条件。
3. 条件满足则执行对应的逻辑

```
synchronized(object){
  while(条件不满足){
    object.wait();
  }
  dosomething;
}
```

通知方：

1. 获得对象的锁

2. 改变条件
3. 通知等待在对象上的锁

```
synchronized(object){
  改变条件;
  object.notify();
}
```

3、为何调用wait或者notify一定要加synchronized，不加行不行?

如果你不加，你会得到下面的异常

```
Exception in thread "main" java.lang.IllegalMonitorStateException
```

> 在JVM源代码中首先会检查当前线程是否持有锁,如果没有持有则抛出异常

因为wait/notify是为了线程间通信的，为了这个通信过程不被打断，需要保证wait/notify这个整体代码块的原子性，所以需要通过synchronized来加锁。

4、notify()锁不释放

当方法wait()被执行后，锁自动被释放。

但当执行玩notify()方法后，锁不会自动释放。必须执行完notify()方法所在的synchronized代码块后才释放。

5、当interrupt方法遇到wait方法

当线程呈wait状态时，对线程对象调用interrupt方法会出现InterrupedException异常。

6、Object类有哪些方法并分别简单介绍

`getClass()`：这是一个`final`方法，亦即不可重写。其作用是获取对象的运行时`Class`。

`hashCode()`：获取对象的哈希值，一般情况下是根据对象的地址或者字符串或者数字计算。

`equals()`：判断两个对象是否相等。

`clone()`：这是一个`protected`方法。实现对象的浅复制，只有当对象实现了`Cloneable`接口才可以调用该方法，否则抛出`CloneNotSupportedException`异常。

`toString()`：返回一个能够表示该对象的字符串，一般来说该字符串应该是简明而有意义的，且尽量所有的子类都应该重写该方法。

`notify()`：唤醒在该对象上等待的某个线程，如果有多个线程在该对象上等待，那么按照一定的算法唤醒其中一个。

`notifyAll()`：唤醒在该对象上等待的所有线程

`wait()`：使当前线程在该对象上等待。

`finalize()`：这是一个`protected`方法。当对象被释放时，该方法则会被调用进而程序员可通过该方法释放JVM无法管理的内存以避免内存泄漏。

7、Object类中equal与hashCode以及==

hashCode() 的作用是获取哈希码，也称为散列码；它实际上是返回一个int整数。这个哈希码的作用是确定该对象在哈希表中的索引位置。

equals() : 它的作用也是判断两个对象是否相等。但它一般有两种使用情况：

- 情况1：类没有覆盖 equals() 方法。则通过 equals() 比较该类的两个对象时，等价于通过“==”比较这两个对象。
- 情况2：类覆盖了 equals() 方法。一般，我们都覆盖 equals() 方法来比较两个对象的内容是否相等；若它们的内容相等，则返回 true (即，认为这两个对象相等)。

== : 它的作用是判断两个对象的地址是不是相等。即，判断两个对象是不是同一个对象(基本数据类型==比较的是值，引用数据类型 ==比较的是内存地址)。

首先最重要的是`hashCode()`相等并不表示对象相等，`equals()`为`true`则对象一定相等，且Java要求此时的计算出的`hashCode()`必须一致。所以实际上的判断对象是否相等是先判断`hashCode()`是否相等，如果不相等则对象肯定不相等，如果相等则继续调用`equals()`判断对象是否相等。

hashCode（）与equals（）

- 如果两个对象相等，则hashcode一定也是相同的
- 两个对象相等,对两个对象分别调用equals方法都返回true
- 两个对象有相同的hashcode值，它们也不一定是相等的
- **因此，equals 方法被覆盖过，则 hashCode 方法也必须被覆盖**
- hashCode() 的默认行为是对堆上的对象产生独特值。如果没有重写 hashCode()，则该 class 的两个对象无论如何都不会相等（即使这两个对象指向相同的数据）

8、详述finalize()作用

当对象被释放时，该方法则会被调用，进而程序员可通过该方法释放JVM无法管理的内存以避免内存泄漏。

finalize()是Object的protected方法，子类可以覆盖该方法以实现资源清理工作，GC在回收对象之前调用该方法。

9、clone()方法的浅拷贝和深拷贝

我们在Java开发时，有时会涉及到对象拷贝复制，就是将一个对象的所有属性（成员变量）复制到另一个对象中。

【浅拷贝】

若成员变量为基本数据类型，拷贝时为【值传递】，是两份不同的数据，改变A类中的该变量，B类不会变化

若成员变量为引用类型，拷贝时为【引用传递】，拷贝的是引用变量，实际上两个引用变量指向的是一个实例对象，若改变A类中的该变量，B会变化，即为【浅拷贝】

【深拷贝】

深拷贝就是对引用类型的成员变量拷贝时，拷贝整个对象，而不是只拷贝引用

clone()方法

- Object提供了clone()方法帮助我们进行拷贝，但它只是浅拷贝，如果需要深拷贝，需要重写clone()方法的逻辑
- 由源码protected native Object clone() throws CloneNotSupportedException可知，我们无法直接调用clone()方法，我们需要重写该方法并使用super.clone()方法去调用Object的clone方法来实现
- 使用clone方法的类必须实现Cloneable接口，否则会抛出异常CloneNotSupportedException

String类

1、什么是String，它是什么数据类型?

String是定义在 java.lang 包下的一个类。它不是基本数据类型。 String是不可变的，JVM使用字符串池来存储所有的字符串对象。

2、创建String对象的不同方式有哪些?

和使用其他类一样通过new关键字来创建。

使用这种方式时，JVM创建字符串对象但不存储于字符串池。我们可以调用intern()方法将该字符串对象存储在字符串池，如果字符串池已经有了同样值的字符串，则返回引用。

使用双引号直接创建。

使用这种方式时，JVM去字符串池找有没有值相等字符串，如果有，则返回找到的字符串引用。否则创建一个新的字符串对象并存储在字符串池。

```
String str = new String("abc");
String str1 = "abc";
```

3、如何判断两个String是否相等?

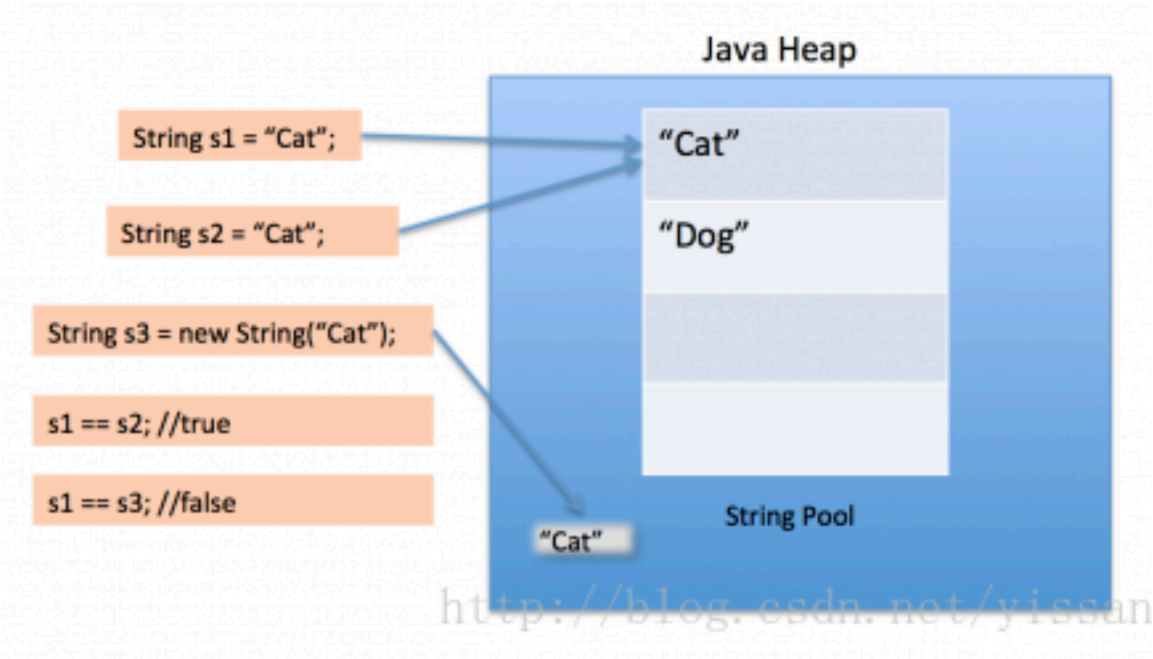
有两种方式判断字符串是否相等，使用”== ”或者使用equals方法。当使用”==”操作符时，不仅比较字符串的值，还会比较引用的内存地址。大多数情况下，我们只需要判断值是否相等，此时用equals方法比较即可。

```
        String s1 = "abc";
        String s2 = "abc";
        String s3= new String("abc");
        System.out.println("s1 == s2 ? "+(s1==s2)); //true
        System.out.println("s1 == s3 ? "+(s1==s3)); //false
        System.out.println("s1 equals s3 ? "+(s1.equals(s3))); //true
```

4、什么是字符串池?

字符串常量池就是用来存储字符串的。它存在于Java 堆内存。

下图解释了字符串池在java堆空间如何存在以及当我们使用不同方式创建字符串时的情况。



5、String、StringBuffer 和 StringBuilder 的区别是什么? String 为什么是不可变的?

String 类中使用 final 关键字修饰字符数组来保存字符串，`private final char value[]`，所以 String 对象是不可变的。而StringBuilder 与 StringBuffer 都继承自 AbstractStringBuilder 类，在 AbstractStringBuilder 中也是使用字符数组保存字符串`char[]value` 但是没有用 final 关键字修饰，所以这两种对象都是可变的。

StringBuilder 与 StringBuffer 的构造方法都是调用父类构造方法也就是 AbstractStringBuilder 实现的。

线程安全性:

String 中的对象是不可变的，也就可以理解为常量，线程安全。

StringBuffer 对方法加了同步锁或者对调用的方法加了同步锁，所以是线程安全的。

StringBuilder 并没有对方法进行加同步锁，所以是非线程安全的。

性能:

每次对 String 类型进行改变的时候，都会生成一个新的 String 对象，然后将指针指向新的 String 对象。StringBuffer 每次都会对 StringBuffer 对象本身进行操作，而不是生成新的对象并改变对象引用。相同情况下使用 StringBuilder 相比使用 StringBuffer 仅能获得 10%~15% 左右的性能提升，但却要冒多线程不安全的风险。

对于三者使用的总结:

1. 操作少量的数据: 适用String
2. 单线程操作字符串缓冲区下操作大量数据: 适用StringBuilder
3. 多线程操作字符串缓冲区下操作大量数据: 适用StringBuffer

6、在Java中String是不可变的和无法改变的，有什么好处?

1、可以使用字符串池来存储字符串，提高存储效率。

2、增加安全性，在存储一些敏感信息，如数据库用户名，密码等是，黑客不能改变它的值。java的类加载器加载类时，字符串的不变性可以确保正确的类被加装。

3、由于String是不可变的，它是安全的,在多线程环境下，我们不需要任何同步。