**什么是IO流？**

它是一种数据的流从源头流到目的地。比如文件拷贝，输入流和输出流都包括了。输入流从文件中读取数据存储到进程(process)中，输出流从进程中读取数据然后写入到目标文件。

字节字符区别

字节是存储容量的基本单位，字符是数子，字母，汉子以及其他语言的各种符号。

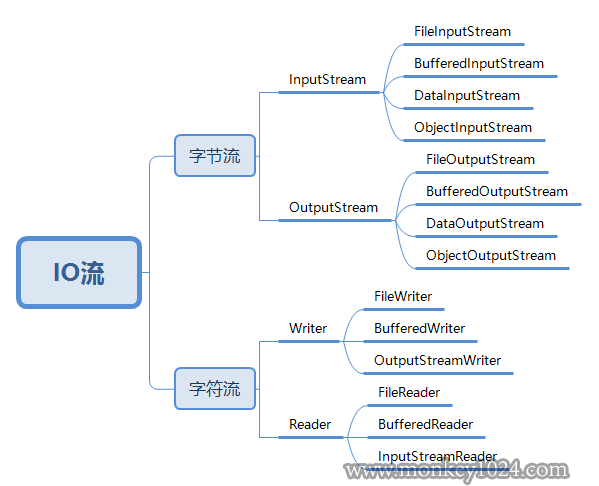
1 字节=8 个二进制单位：一个一个字符由一个字节或多个字节的二进制单位组成

**字节流和字符流的区别。**

字节流在JDK1.0中就被引进了，用于操作包含ASCII字符的文件。JAVA也支持其他的字符如Unicode，为了读取包含Unicode字符的文件，JAVA语言设计者在JDK1.1中引入了字符流。ASCII作为Unicode的子集，对于英语字符的文件，可以使用字节流也可以使用字符流。

**FileInputStream和FileOutputStream是什么？**

这是在拷贝文件操作的时候，经常用到的两个类。在处理小文件的时候，它们性能表现还不错，在大文件的时候，最好使用BufferedInputStream (或 BufferedReader) 和 BufferedOutputStream (或 BufferedWriter)



**IO流的分类**

**按照流向分类：**

输入流（InputStream和Reader）：从硬盘或者别的地方读到内存里面

输出流（OutputStream和Writer）：从内存里面向硬盘或者别的地方输出

**按照操作类型分类：**

字节流（InputStream和OutputStream） : 字节流可以操作任何数据,因为在计算机中的数据都是以字节的形式存储的。

字符流（Reader和Writer） : 字符流只能操作纯字符数据，防止乱码。

**InputStream,OutputStream,Reader和Writer都是抽象类**

**使用File.separator解决不同系统的路径问题**

**缓冲区**

缓冲区可以简单地理解为一段内存区域。可以简单地把缓冲区理解为一段特殊的内存。某些情况下，如果一个程序频繁地操作一个资源（如文件或数据库），则性能会很低，此时为了提升性能，就可以将一部分数据暂时读入到内存的一块区域之中，以后直接从此区域中读取数据即可，因为读取内存速度会比较快，这样可以提升程序的性能。在字符流的操作中，所有的字符都是在内存中形成的，在输出前会将所有的内容暂时保存在内存之中，所以使用了缓冲区暂存数据。如果想在不关闭时也可以将字符流的内容全部输出，则可以使用Writer类中的flush()方法完成。

**乱码**

计算机中存储的数据都是二进制的数字，我们看到的文字信息是将二进制转换之后显示的，两者之间存在编码与解码的过程，其互相转换必须遵循某种规则，即编码和解码都遵循同一种规则才能将文字信息正常显示，如果编码跟解码使用了不同的规则，就会出现乱码的情况

**File类常用方法**

boolean isDirectory()判断是否是目录

boolean isFile()判断是否是文件

boolean exists()判断是否存在

boolean isHidden()判断是否隐藏

String getAbsolutePath()获取绝对路径

long length()获取文件大小。字节数

long lastModified()获取最后一次的修改时间，毫秒值

String[] list()获取文件路径下所有文件名（同一层级目录）

File[] listFiles()获取文件路径下所有File对象数组

**序列化和反序列化的作用**

在工作中有可能遇到多台机器远程通信的情况，如果要将机器A中的某个java对象传输到机器B上面，需要将这个java对象转换为字节序列然后进行传输。将对象转换为字节序列的过程叫做序列化，反之叫做反序列化。

如果不希望将某个属性序列化，可以使用transient声明该属性，在序列化时将忽略这个属性。

一个对象如果支持序列化，需要实现Serializable的接口，这个接口中没有任何方法，实现该接口后，JVM会给这个对象做特殊待遇

**什么是设计模式？**  
设计模式是可以重复利用的解决方案。软件开发的先驱或者前辈们将之前在开发中遇到的问题进行总结并给出了解决方案，后辈在遇到这些问题之后直接使用这些方案即可解决问题