# JavaIo

## 讲讲IO里面的常见类，字节流、字符流、接口、实现类、方法阻塞。

1.File类

java 处理文件的类 File,java提供了十分详细的文件处理方法，举了其中几个例子:

public static void creatFile(){  
 //File类主要用于文件和目录的创建、文件的查找和文件的删除等  
 File f = new File("E:/iotest/create123");  
 try {  
 //当f中的文件路径不存在时，创建一个新的空文件,可以多级创建  
 // f.createNewFile();  
 //返回由此抽象路径名表示的文件或目录的名称，此例中输出的是E盘的容量  
 System.*out*.println("该分区大小"+f.getTotalSpace()/(1024\*1024\*1024)+"G");  
 //创建目录  
 */\*\*  
 \* mkdirs()可以建立多级文件夹， mkdir()只会建立一级的文件夹  
 \*/* f.mkdirs();  
 System.*out*.println("文件名 "+f.getName());//返回文件名即 create123  
 //返回父目录即： E:\iotest  
 System.*out*.println("文件父目录字符串 "+f.getParent());  
 }  
 catch (Exception e)  
 {  
 e.printStackTrace();  
 }  
}

1. 字节流

字节流主要是操作byte类型数据，以byte数组为准，主要操作类就是OutputStream、InputStream。两者都是抽象类，是字节输入、输出流的父类。

BufferedInputStream和InputStream类都是字节流，而更确切地说，BufferedInputStream是缓冲字节流。

InputStream是不带缓冲的操作，每读一个字节就要写入一个字节，由于涉及磁盘的IO操作相比内存的操作要慢很多，所以不带缓冲的流效率很低。BufferedInputStream带缓冲的流，可以一次读很多字节，但不向磁盘中写入，只是先放到内存里。等凑够了缓冲区大小的时候一次性写入磁盘，这种方式可以减少磁盘操作次数，速度就会提高很多！这是两者的区别。

1. 阻塞IO模型

最传统的一种IO模型，即在读写数据过程中会发生阻塞现象。当用户线程发出IO请求之后，内核会去查看数据是否就绪，如果没有就绪就会等待数据就绪，而用户线程就会处于阻塞状态，用户线程交出CPU。当数据就绪之后，内核会将数据拷贝到用户线程，并返回结果给用户线程，用户线程才解除block状态。

2.非阻塞IO模型

## 2.讲讲NIO

答：

NIO最重要的组成部分

通道 Channels

缓冲区 Buffers

选择器 Selectors

传统的IO流是阻塞式的，会一直监听一个ServerSocket，在调用read等方法时，他会一直等到数据到来或者缓冲区已满时才返回。调用accept也是一直阻塞到有客户端连接才会返回。每个客户端连接过来后，服务端都会启动一个线程去处理该客户端的请求。并且多线程处理多个连接。每个线程拥有自己的栈空间并且占用一些 CPU 时间。每个线程遇到外部未准备好的时候，都会阻塞掉。阻塞的结果就是会带来大量的进程上下文切换。

对于NIO，它是非阻塞式，核心类：

1.Buffer为所有的原始类型提供 (Buffer)缓存支持。

2.Charset字符集编码解码解决方案

3.Channel一个新的原始 I/O抽象，用于读写Buffer类型，通道可以认为是一种连接，可以是到特定设备，程序或者是网络的连接。

Java NIO是一个用来替代标准Java IO API的新型数据传递方式，像现在分布式架构中会经常存在他的身影。其比传统的IO更加高效，非阻塞，异步，双向

所有的 IO 在NIO 中都从一个Channel 开始。Channel 有点象流

Channel的实现

FileChannel:从文件中读写数据

DatagramChannel:通过UDP读写网络中的数据

SocketChannel:通过TCP读写网络中的数据

ServerSocketChannel:监听新进来的TCP连接，像Web服务器那样。对每一个新进来的连接都会创建一个SocketChannel

Scatter/Gather

分散（scatter）从Channel中读取是指在读操作时将读取的数据写入多个buffer中。因此，Channel将从Channel中读取的数据“分散（scatter）”到多个Buffer中

聚集（gather）写入Channel是指在写操作时将多个buffer的数据写入同一个Channel，因此，Channel 将多个Buffer中的数据“聚集（gather）”后发送到Channel

## String 编码UTF-8 和GBK的区别?

答：UTF-8是国际通用的编码，通用性比较好，GBK是国家编码，支持中文，但是相比UTF-8的通用性差，不过UTF-8占用的数据库比较大，所以相对反应速度要慢

## 4.什么时候使用字节流，什么时候使用字符流

答：InputStream 和OutputStream,两个是为字节流设计的,主要用来处理字节或二进制对象,

Reader和 Writer.两个是为字符流（一个字符占两个字节）设计的,主要用来处理字符或字符串.

字符流处理的单元为2个字节的Unicode字符，操作字符、字符数组或字符串，

字节流处理单元为1个字节，操作字节和字节数组。

所以字符流是由Java虚拟机将字节转化为2个字节的Unicode字符为单位的字符而成的，

所以它对多国语言支持性比较好！

如果是音频文件、图片、歌曲，就用字节流好点，

如果是关系到中文（文本）的，用字符流好点

所有的输入都是转换成字节流之后，然后在内存中变成字符流。所以一般建议使用字符流。但是遇到中文汉字，出现乱码的情况下，可以使用字节流。

在所有的硬盘上保存文件或进行传输的时候都是以字节的方法进行的，包括图片也是按字节完成，而字符是只有在内存中才会形成的，所以使用字节的操作是最多的。我们建议尽量尝试使用字符流，一旦程序无法成功编译，就不得不使用面向字节的类库，即字节流。

## 5．递归读取文件夹的文件，代码怎么实现

答：利用File中的listFiles方法，将所有文件保存访问，然后递归调用。代码如下：

/ public void visitFile(String path){

File file = new File(path);

File[] files = file.listFiles();

if(files==null){

return;

}

for (File file1:files)

{

if (file1.isDirectory())

{

System.out.println("Directory");

visitFile(file1.getPath());

}else if (file1.isFile())

{

System.out.println(file.getPath()+"\\"+file1.getName());

}

}

}