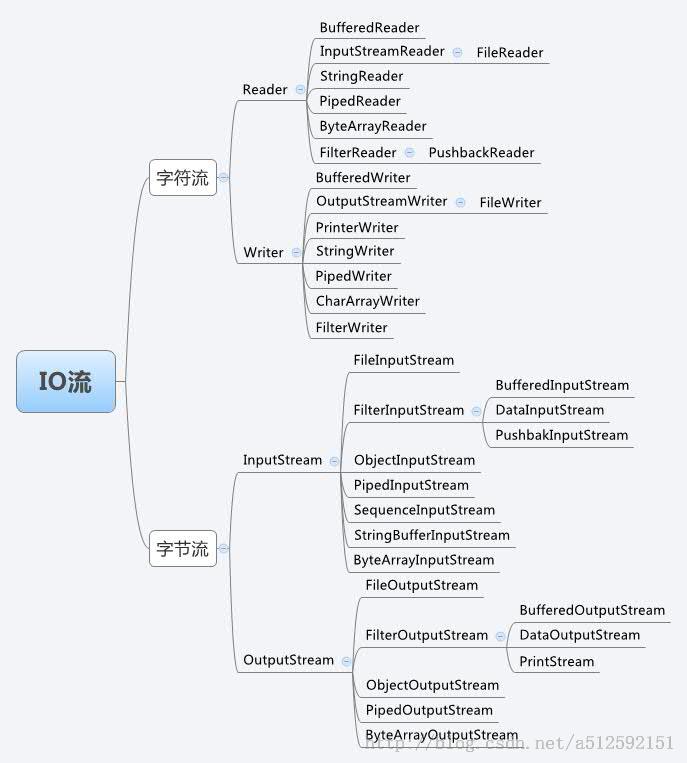
IO流的面试题



一、字节流      （用字节流处理字符数据可能会有编码问题，因为字节流是以字节为单位，没有编码，而字符流是以字符为单位传送数据，字符流即以字节流+编码）

  ·两个顶层父类 （抽象类）及实现类

        ·InputStream（读入内存）  ：所有字节输入流相关类的父类

              ··FileInputStream ：obtain input bytes from a file in a file system,for reading streams of raw bytes(原始字节) such as image data..For writing streams of characters,consider using FileReader

                       初始化时（构造函数）要和文件关联，读取的对象，要首先判断文件是否存在

                     ——read():read a byte of data from this inputStream.

                     ——read(byte [] b):read up to b.length bytes of data from this inputStream into an array of bytes.

                     ——read(byte [] b,int off,int length)

                     ——close()

import java.io.\*;

public class Demo1 {

public static void main(String [] args){

File file = new File("d:/helloWorld.txt");

InputStream in = null;

try {

if (!file.exists()){ //文件不存在则创建

file.createNewFile();

}

in = new FileInputStream(file);

byte [] buf = new byte[1024]; //先写到缓存中，然后再一起写到其他外设中

int length = 0;

while ((length=in.read(buf))!=-1){ //-1 represent the end of the file is reached ，

//字节一个一个地读入到内存

System.out.println(new String(buf,0,length)); //需要将int转为字节，如果为中文的话输出乱码字符 ，

//此处其实是写出到了外设（控制台）上，System.out返回的是PrintStream对象

}

} catch (IOException e) {

e.printStackTrace();

}finally {

if (in != null){

try {

in.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

               ··ByteArrayInputStream:包含一个内置的缓冲器存储字节

                      构造函数要和字节数组相关联:byte [] buff

                      ——read():从输入流中读取下一个字节

                      ——read(byte [] buff,int off,int len):

                      ——close():关闭后并没有影响，类中的方法仍然可以调用而不抛出IO异常

        ·OutputStream(写出内存)：所有和输出字节流相关类的父类

               ··FileOutputStream：for writing data to a file or a FileDescriptor,for writing streams of raw data(原始字节)such as image data.For writing streams of characters,consider using FileWriter.

                    初始化时要和文件关联，写出的目的地。没有该文件时会自动创建。

                    ——write(int b):write the specified(指定的) byte to this file output stream.

                    ——write(byte [] b):

                    ——write(byte [] b,int off,int len)

                    ——close()

import java.io.\*;

public class Demo2 {

public static void main(String [] args){

File file = new File("d:/helloWorld3.txt"); //没有会自动创建

OutputStream out = null;

try {

out = new FileOutputStream(file);

out.write(69); //文件中产生ASC码对应的字符

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

       ★ FileInputStream  & FileOutputStream   协同完成文件复制(不会乱码)

import java.io.\*;

public class Demo3 {

/\*\*

\* 从一个文件复制到另一个文件

\* @param args

\*/

public static void main(String [] args){

File origin = new File("d:/helloWorld.txt");//原始文件

if (!origin.exists()){

try {

origin.createNewFile();

} catch (IOException e) {

e.printStackTrace();

}

}

File destination = new File("d:/helloWorld4.txt");//目的文件

InputStream in = null;

OutputStream out = null;

try {

in = new FileInputStream(origin);

out = new FileOutputStream(destination);

byte [] buff = new byte[1024];

int len = 0;

while ((len=in.read(buff))!=-1){

out.write(buff,0,len);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

try {

if (in != null){

in.close();

}

if (out != null){

out.close();

}

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

二、字符流

    ·两个顶层父抽象类及其实现类

         ·Reader：for reading character streams

              ··InputStreamReader:从字节流到字符流的桥梁：读取字节流然后按指定的编码方式进行解码（看不懂→能看懂）

                         构造函数要和输入流InputStream/编码方式Charset相关联:System.in/FileInputStream传入

                          ——read()    :读入一个字符

                          ——read(char [] cbuf,int offset,int length)

                          ——close()

                    ···FileReader:读字符文件的方便类，本质是InputStreamReader在构造时 指定了默认的编码方式，用于读取字符流

                             构造函数要和文件相关联

              ★InputStreamReader 接收键盘上输入的数据，写入文件中（中文会乱码）

import java.io.\*;

public class Demo4 {

/\*\*

\* 控制台输入，写到文件夹中

\* @param args

\*/

public static void main(String [] args){

File file = new File("d:/helloWorld.txt");//会覆盖之前的数据

OutputStream out = null;

InputStreamReader reader = null;

try {

reader = new InputStreamReader(System.in);

out = new FileOutputStream(file);

int len = 0;

while ((len = reader.read())!= -1){

out.write(len);

}

} catch (IOException e) {

e.printStackTrace();

} finally {

if (out!=null){

try {

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if (reader!=null){

try {

reader.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

          ··BufferedReader：从一个字符输入（character-input）流中读取文本（text），并进行缓冲字符,默认缓存8192（8M），行最长80

                 构造函数要和Reader in/int size 关联：InputStreamReader

                 ——in read（）

                 ——in read(char [] cbuf,int off,int len)

                 ——String readLine()

                 ——close()

                   ★键盘录入，使用缓存

BufferedReader buffr = new BufferedReader(new InputStreamReader(System.in))

      ·Writer:for writing to character streams （字符流的写操作基本上后面都需要进行flush()操作）

             ··OutputStreamWriter :从字符流到字节流的桥梁：写出的字符被用指定的编码方式进行编码。

                   构造函数要和OutputStream out/charset 关联：System.out/FileOutputStream

                   ——write(int c):写一个单独的字符

                  ——write(char [] cbuf,int off,int len)

                   ——write(String str,int off,int len)

                   ——flush():刷新流

                   ——close()

                   ···FileWriter:写字符文件的方便类，实质是：OutputStreamWriter指定了默认的本机编码方式，且可以处理文件

              ··BufferedWriter:写文本到一个字符输出（character-out）流，并对传入的字符进行缓存

                    构造函数要和 Writer out/int size 相关联

                     ——write(int c):写一个单独的字符

                     ——write(char [] cbuf,int off,int len)

                     ——write(String str,int off,int len)

                     ——newLine():换行

                     ——flush():刷新流

                     ——close()

                    ★控制台输出，使用缓存

BufferedWriter buffw= new BufferedWriter(new OutputStreamWriter(System.out,"utf-8"));

                   ★键盘输入，控制台输出功能

import java.io.\*;

public class Demo5 {

/\*\*

\* 键盘输入，控制台输出

\* @param args

\*/

public static void main(String[]args){

BufferedReader buff = null;

BufferedWriter bufferedWriter = null;

String line = null;

try {

buff = new BufferedReader(new InputStreamReader(System.in,"utf-8"));

bufferedWriter = new BufferedWriter(new OutputStreamWriter(System.out,"utf-8"));

while ((line=buff.readLine())!=null){

bufferedWriter.write(line);

}

  bufferedWriter.flush(); //一定要刷新

} catch (IOException e) {

e.printStackTrace();

} finally {

if (buff!=null){

try {

buff.close();

} catch (IOException e) {

e.printStackTrace();

}

}

if (out!=null){

try {

out.close();

} catch (IOException e) {

e.printStackTrace();

}

}

}

}

}

面试题

**1.java中有几种类型的流？jdk为每种类型的流提供了一些抽象类以供继承，请说出它们分别是什么？**

解题思路：**了解io流的体系（重要）**就可以了

从大的方面来分可以分为字节流和字符流.字符流提供了提供了reader和writer;字节流提供了outputstream 和inputstream.

**2.字符流和字节流有什么区别？（重要）**

解题思路：从读写的原理上来进行区分会好记一点（字符流处理的单元为 2 个字节的 Unicode 字符，分别操作字符、字符数组或字符串；而字节流处理单元为 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，如果是音频文件、图片、歌曲，就用字节流好点（避免数据丢失）；如果是关系到中文（文本）的，用字符流好点）

字符流以字符或者字符数组的形式读写数据,只能读写二进制文件;字节流能读写各种类型的数据.

**3.什么是java序列化，如何实现java序列化？（重要）**

解题思路：像这样的题，一般先做名词解释，然后阐述作用和使用方法

序列化就是一种用来处理对象流的机制，所谓对象流也就是将对象的内容进行流化（将对象转换成二进制）。可以对流化后的对象进行读写操作，也可将流化后的对象传输于网络之间。序列化是为了解决在对对象流进行读写操作时所引发的问题。

注解：当两个进程在进行远程通信时，彼此可以发送各种类型的数据。无论是何种类型的数据，都会以二进制序列的形式在网络上传送。发送方需要把这个Java对象转换为字节序列，才能在网络上传送；接收方则需要把字节序列再恢复为Java对象。

　　将需要序化的类实现Serializable接口就可以了，该接口没有任何方法，可以理解为一个标记，即表明这个类可以序列化。注意的是被关键字static、transient修饰的变量不能被序列化。在被序列化后，transient修饰的变量会被设为初始值。如int型的是0、对象型的是null.

1.概念

　　序列化：把Java对象转换为字节序列的过程。  
　　反序列化：把字节序列恢复为Java对象的过程。

2.用途

对象的序列化主要有两种用途：  
1） 把对象的字节序列永久地保存到硬盘上，通常存放在一个文件中；  
2） 在网络上传送对象的字节序列。

4.**什么是比特(Bit),什么是字节(Byte),什么是字符(Char),它们长度是多少,各有什么区别（一般是笔试题的选择题里面出的多一点）**

Bit是最小的传输单位,byte是最小的存储单位,1byte=8bit,char 是一种基本数据类型,1char=2byte.

**5.BufferedReader属于哪种流,它主要是用来做什么的,它里面有那些经典的方法**

解题思路：望文知意，Reader是字符流，而buffer是缓冲的作用，缓冲区是基于内存的，起到读写高效的作用；所以BufferedReader是高效字符流

BufferedReader是字符流,也是一种包装流,用来增强reader流.主要用来读取数据的,最经典的方法是readline,可以一次读一行,是reader不具备的.

**6.什么是节点流,什么是处理流,它们各有什么用处,处理流的创建有什么特征**

节点流: 直接与数据源相连，用于输入或者输出  
处理流：在节点流的基础上对之进行加工，进行一些功能的扩展  
处理流的构造器必须要传入节点流的子类

**7.如果我要对字节流进行大量的从硬盘读取,要用那个流,为什么？**

解题思路：因为明确说了是对字节流的读取，所以肯定是inputstream或者他的子类，又因为要大量读取，肯定要考虑到高效的问题，自然想到缓冲流。

用BufferedInputStream,原因：BufferedInputStream是InputStream的缓冲流，使用它可以防止每次读取数据时进行实际的写操作，代表着使用缓冲区。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。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！并且也可以减少对磁盘的损伤。

**8.如果我要打印出不同类型的数据到数据源,那么最适合的流是那个流,为什么？**

解题思路：要考虑到打印的问题，就要用到打印输出流（printstream：操作字节流；或者是PrintWriter操作字符流）

用printstream,因为只有字节流才能读写各种类型的数据.

**9.怎么样把输出字节流转换成输出字符流,说出它的步骤？**

解题思路：把字节流转成字符流，这里肯定就要用到适配器模式，很自然的要想到outputstreamwriter。它继承了Writer接口，但要创建它必须在构造函数中传入一个outputstream的实例，outputstreamwriter的作用也就是将outputstream适配到Writer。，它实现了Reader接口，并且持有了InputStream的引用。（关于适配器的作用，可以想一下电脑转接头的例子）

利用转换流outputstreamwriter.创建一个字节流对象,将其作为参数传入转换流outputstreamwriter中,得到字符流对象.

**10.什么叫对象序列化，什么是反序列化，实现对象序列化需要做哪些工作？**

对象序列化:将对象以二进制的形式保存到硬盘上;

反序列化:将二进制文件转化为对象读取.

将需要序化的类实现Serializable接口

**11.说说你对io流的理解**

解题思路：个人觉得，可以讲io流的作用和体系

io流就相当于读写数据的一个管道.主要分为输入流和输出流,分别对应读数据和写数据.

**12递归读取文件夹的文件**

package test;

import java.io.File;

/\*\*

\*

\* 递归读取文件夹的文件

\*/

public class ListFileDemo {

public static void listFile(String path) {

if (path == null) {

return;// 因为下面的new File如果path为空，回报异常

}

File[] files = new File(path).listFiles();

if (files == null) {

return;

}

for(File file : files) {

if (file.isFile()) {

System.out.println(file.getName());

} else if (file.isDirectory()) {

System.out.println("Directory:"+file.getName());

listFile(file.getPath());

} else {

System.out.println("Error");

}

}

}

public static void main(String[] args) {

ListFileDemo.listFile("D:\\data");

}

}

**13Java IO体系中有太多的对象，到底用哪个呢？**   
明确操作的数据设备。   
数据source对应的设备：**硬盘(File)**，**内存(数组)**，**键盘(System.in)**   
数据destination对应的设备：**硬盘(File)，内存(数组)，控制台(System.out)**。

记住，只要一读取键盘录入，就用这句话。   
BufferedReader bufr = new BufferedReader(new InputStreamReader(System.in));   
BufferedWriter bufw = new BufferedWriter(new OutputStreamWriter(System.out));

https://blog.csdn.net/baidu\_37107022/article/details/76890019