1.io:输入输出

2.流：是一种抽象概念，是对数据传输的总称

3.输入流：读数据
输出流：写数据

4.按照数据类型来分：
字节流：
字节输入流，字节输出流//读不懂
字符流：
字符输入流，字符输出流//读得懂

5.InputStream：表示输入字节流的所有超类
OutputStream：表示输出字节流的所有超类
子类名称都是以其父类作为子类名的后缀

6.FileOutputStream：文件输出流用于将数据写入File
FileOutputStream(String name)：创建文件输出流以指定的名称写入文件
a.创建字节输出流对象：调用系统功能创建了文件，创建了字节输出流对象，让字节输出流对象指向创建好的文件
b.用这个对象里的方法写入数据：
eg：void write(int b);b是ASCII码
void write(byte[] b):将b.length字节从指定的字节数组写入此文件输出流，一次写一个字 节数组数据
void write(byte[] b,int off,int len)：将len字节从指定的字节数组开始，从偏移量off开始写入文件输出流，一次写一个字节数组的部分数据
c.所有和IO操作有关的都要释放资源：关闭此文件输出流并释放与此流相关联的任何系统资源

7.“字符串”.getByte()；就可以得到一个字节数组
eg：fileoutput("hello".getByte());//就可以实现把hello这个字符串写入文件的操作

8.a.字节流数据如何换行 and b.字节流写数据如何实现追加写入

a.可以在文件里写入数据之后加“\r\n”来实现换行

window:\r\n

linux:\n

mac:\r 不同的操作系统识别不一样

b.在创建输出对象时，加一个Boolean型，true就是在文件末尾加。

9.字节流写数据加异常处理

用try{}catch{}finally{}方法抛出IOException异常；在finally中要执行所有释放资源，都会执行，要让finally知道资源的位置，在定义对象和初始化的时候要在try外面定义，在finally中还要进行判断，就是在对象为空的时候就不会去释放资源，不然会空指针报错。也可以直接抛出异常

10.字节流读数据

FileInputStream：从文件系统中的文件获取输入字节

FileInputStream（String name）：通过打开与实际文件相连接来创建一个FileInputStream，该文件由文件系统的路径名name命名

使用字节输入流读数据的步骤：

a.创建字节输入流对象

b.调用字节输入流对象的读数据方法

c.释放资源

11.一次读一个数据：

eg：第一次读取数据： int by=fis.read();//输出的是ASCII码，如果想看到字符，就强制转换

第二次读取数据：by=fis.read();

也可以用循环读取数据

如果文件里面没有数据了，则函数返回-1，所有可以根据读取数值是不是-1来判断是否到达文件末尾,可以读取换行。

步骤：a.在循环外面先读取一次，再在循环外面读取一次；这样才会往下读取

eg:

```
int by =  fis.read();
while(by != -1)
{
    System.out.print((char)by);
```

```
        by = fis.read();
    }
    //代码优化
    int by;
    while(by=fis.read() != -1)
    {
        System.out.print((char)by);
    }
```

12.案例：复制文本文件

分析：

a.就是把文本文件的内容从一个文件中读取出来，然后写入另一个文件中去。

b.数据源：“文件地址”----读数据----InputStream（抽象类）---FileInputStream

c.目的地：“文件地址”----写数据----OutputStream（抽象类）---FileOutputStream

思路：

a.根据数据源创建字节输入流对象

b.根据目的地创建字节输出流对象

c.读写数据，复制文本文件

d.释放资源

13.一次读一个字节数组数据

步骤：

a.创建字节输入流对象

b.调用方法

c.释放资\

eg：

```
byte[] bys = new byte[1024];//1024及其整数倍
    int len;
    while((len=fis.read(bys)) != -1)
    {
        System.out.print(new String(bys,0,len));
    }
```

14.字节缓冲流

BufferedOutputStream：该类实现缓冲输出流，通过设置这样的输出流，应用程序可以向底层输出流写入字节，而不必为写入的每个字节导致底层系统的调用。

BufferedInputStream：创建BufferedInputStream将创建一个内部缓冲区数组。当从流中读取或跳过字节时，内部缓冲区将根据需要从所包含的输入流中重新填充，一次很多字节。

构造方法：

BufferOutputStream(OutputStream out)

BufferedInputStream(InputStream in)

字节缓冲区仅仅提供缓冲区，而真正的读写数据还得依靠基本的字节流对象进行操作

15.一个汉字的存储:

如果是GBK编码，占用2个字节

如果是UTF-8编码，占用3个字节

但无论在哪种编码中 第一个字节都是负数

16.字符流

Reader：

FileReader(String filename)

FileReader(File file)

(char)fr.read(file);

sout(new String(char))

Writer：

fw.write(String a);

必须得要close();操作，如果没有就得写一个fw.flush()操作，这样才会有字符得写入//因为有缓冲流

fw.write()中的格式和sout中差不多

17.缓冲输入流

BufferedReader(Reader in,[int sz]):创建一个使用指定大小输入缓冲区的缓冲字符输入流

可以一次只读一行，然后循环。

18.转换流

InputStreamReader：将字节输入流转换为字符输入流，用于将一个字节流中的字节解码成字符

eg：BufferedReader br = new BufferedReader(new InputSreamReader(System.in));

Sout(br.readLine());//输出从键盘里读取的内容

OutputStreamWriter：将字节输出流转换为字符输出流，用于将写入的字符编码成字节后写入一个字节流

19.对象流

把对象转换为字节序列的过程称为对象序列化

用途：

把对象字节序列永久保存到硬盘上

在网络上传送对象的字节序列

通过序列化在进程间传递对象

只有实现了Serializable和Externalizable接口的类的对象才能被序列化。

Serializable是一个标记接口，里面没有方法

Externalizable接口继承自Serializable接口：

该接口定义了readExternal()和writeExternal()方法

用户可以选择对象具体的属性进行序列化

ObjectOutputStream类用于对象的序列化，即对象输出。

把字节序列恢复为对象的过程称为对象的反序列化

ObjectInputStream类用于对象的反序列化，返回的object对象，所有要进行向下转型

eg：Person person = (person)Object();

transient关键字：用来表示一域不是对象序列化的一部分,比如密码账号这些不愿意在网络上传输的属性值，就可以加上这个关键字

Java对象参与序列化的内容：类名、属性等

transient和Static这两个关键字不会参与序列化

20.数据流

可以实现基本类型的传送问题

DataInputStream：读取Java标准数据类型的输入流

实现多个类的读取

DataOutputStream：写入Java标准数据类型的输出流