## File类

File类是对于文件和文件夹的抽象类，它的对象能够对存在的物理文件进行一些操作。但是，对于不存在的文件，即使创建了对应的File类，也无法进行操作，那么就是一个不存在的一个虚拟文件，目录。

这个时候不存在，既不是文件，也不是文件夹。就是没有和File对象对应的物理文件和文件夹

构造方法

File(“路径”)

File(“父路径”，“子路径”)

File（file, “子路径”）

创建File对象对应的物理文件，删除对应的物理文件和文件夹

File.mkdir() 创建指定目录

File.mkdirs() 创建多级目录

File.createNewFile() 创建对应的文件，而不是文件夹。

File.delete() 删除对象对应的物理文件，文件夹。

获取访问对象

String getName()

String getPath()

String getAbsolutePath()

判断是否存在

Boolean exists（）

Boolean isFile() 在存在的前提下，是一个文件，返回true。否则返回false

Boolean isDirectory() 在存在的前提下，是个目录，返回true，否则返回false

创建文件，是不区分大小写的。但是区分文件的后缀

## Java流输入输出流

File类不能对文件进行读取（从磁盘但内存），和写入（从内存到磁盘）

要想实现文件的读写，就要通过 流 ：

流就是流动的字符，对于内存来说，分为输入流和输出流。输入流基类：Reader和InputStream

输出流基类：Writer和OutputStream

字符流：Reader和Write：基本单位是16个字节的Unicode字符

字节流：InputStream和OutputStream 8位通用字节流

流对象往往和文件关联起来。来实现将文件的内容进行输入和输出

缓冲流：外部设备直接和内存中的输入输出流进行直接交互，传输的效率太低。这是侯，在外设备和内存中加了一层，缓冲流，当外部设备输入到内存中的输出流中，先输入到缓冲流中，当缓冲流到了一定的数量之后，再输入到磁盘中。

当内存中的输入流输出的显示器中，先将信息输入到缓冲流中，当缓冲流到一定的数量后，再一起输出到外部显示设备中。类似于计算机中的多级缓存机制，提高了外部设备和内存之间数据的传输效率

FileInputStream输入流的创建

New FileInputStream(“e://text.txt”) 可以是一个字符串，字符串表示文件的位置

New FileInputStream(file) 可以是一个文件，文件关联了对应的物理文件。

FileOutputStream输出流的创建

New FileOutputStream(“e://text.txt”)字符串表示关联文件的位置

New FileOutputStream(file) 文件，关联了对应的物理文件

New FileOutputStream(file, true) 默认是false，将重写文件内容，改为true，将追加文件末尾

对于输入输出字节流来说，只能是只是写入或者是只是读取。

出现了RandomAccessFile类，随机访问文件类

能够对一个文件同时进行读取和写入操作

能够在文件的任意位置进行读取和写入操作

New RandomAccessFile(“e://text.txt”，“rw”)；表示只读取

New RandomAccessFile（file， string model）model ：r只读取，w只写入，rw读取和写入都可以

字节输入输出流，基本操作单位是字节，

字符输入输出流，基本操作单位是字符。Unicode字符，一个字符占16个字节

字符输入流Reader抽象类，其子类FileRead

New FileRead(file);

New FileRead(“e://text.txt”)

Close()

字符输出流：

New FileWriter(String filename, boolean true/false)

New FileWriter(File file, boolean appender) 默认是false，直接将文件重写

读取文件常用的方法：

Read read = Resource.getResourceAsReader(“类路径下的文件”)

对read进行操作，new SqlSessionFactory.builder(read);

Read.close(); // 关闭流对象

## 字符缓冲流：对其他字符输入输出流的包装

字符缓冲流：提高内存和外部设备之间的文件读取和写入的效率，是内存和外部设备的一个中转站。

字符缓冲输入流：

BufferReader（Read in, int sz）

BufferWriter(Writer out, int sz)

字符缓冲输出流，相当于是在writer上加了一层，能够代替writer对字符进行操作

同时也实现了缓冲功能，到一定量才进行写入操作，提高了效率。

Flush()强制将缓冲区的内容刷新提交到磁盘中

## 序列化和反序列化：对象字节流对其他字节流的包装

I/O操作是以字符字节位基本单位进行操作的，处理的都是字符串类型和基本数据类型，无法对java对象进行处理。

以对象位基本单位进行数据的存储和传输：就是将对象序列化，转换成字节序列，借助对象输入流和对象输出流。

就是将Java对象，Java对象的状态，是保存再内存中的，断电就没有了。序列化就是将Java对象和他的状态转换成字节序列，转换成字节序列之后，就可以将这个字节序列保存到磁盘中，数据库中，实现持久化保存。也能将这个字节序列进行网络传输，实现分享。

同时因为保存的是字节，是平台无关性的。可以将windows系统上的字节序列传输到liunx系统上，再进行反序列化之后，都能得到相同的Java对象，以及相同的Java对象的状态

反序列化：就是将字节序列，读取到内存中，转换成Java对象。

序列化步骤：Java对象，实现系列化接口，这是一个标记接口，表名java对象是可以序列化的。创建一个输出流对象，同时输出流对象关联文件。再使用输出流对象的writeObject（）方法，将java转化成字节流，写入到关联的文件中，最后再关闭字节流

反序列化步骤：创建输入流，同时关联文件（文件里面含有Java对象的字节序列），使用这个输入流，就可以将文件中的内容读取到内存中

使用输入流方法，readObject() 将Java对象的字节流读取出来，实现反序列化，转换成Java对象。这个时候要使用强制类型转换，最后再关闭输入流。

借助对象输入输出流来实现：

对象输入输出流和缓冲字节输入输出流类似，缓冲字符输入输出流是在字符输入输出流上加一层。实现对文件的操作，

对象输入输出流是在字节输入输出流上加一层，实现对对象的操作

OutPutStream os = new FileOutputStream(file);

ObjectOutputStream oos = new ObjectOutputStream(os);

Oos.writer(new Person(“zhangkuan”, 20));

InputStream is = new FileInputStream(file);

ObjectInputStream ois = new ObjectInputStream(is);

Person p = (Person)Ois.readObject();

概括

序列化：创建一个对象输出流，它可以包装一个其他类型的字节输出流。其他类型的字节输出流对应相关的file文件：new ObjectOutputStream(FileOutputStream(file)), oos.writerObject(Object obj);写入对象

反序列化：创建一个对象输入流，它可以包装一个其他的字节输入流，其他类型的字节输入流对应相关的file文件：new ObjectInputStream（FileInputStream(file)）;

ois.readObject(Object obj) 读取对象

对象要实现序列化，就要实现serializable接口，其目的是起一个标记作用，也是出于安全考虑，一些重要敏感的信息，不实现这个接口，就无法序列化，也就无法实现存储和传输，起到了保护数据的作用。如果一个对象中，只对部分信息保密，其他不报名，那就使用transient来修饰，这样，这个属性就不会被序列化，实现了保护作用，防止信息泄露

对象中的属性，如果是引用，被引用的对象也要实现序列化

## 读写二进制文件

DataInputStream和DataOutputStream

相当于是将FileInputStream和FileOutputStream进行包装。

可以进行二进制数据的读写操作

New DataInputStream(new FileInputStream(file))

读取方法：dis.readInt()读取来的是整数 ; dis.readByte() 读取Byte数据

New DataOutputStream(new FileOutputStream(file))

写入方法：dis.write(1) 写入二进制数据

## 标准数据流

Java通过System类来实现标准输入输出功能（对于电脑和外面的人来说）

System类不能创建对象，而是直接使用，一共有3个成员

In，out，err

System.in

System.out

System.err

Public static final InputStream in;

Public static final printStream out;

Public static final printStream err;

In 是一个System对象，也是一个输入流InputStream。是将我们的键盘输入的数据，输入到内存当中。

System.in.read(b) 输入到内存b中 read方法会等待我们输入，当按下回车键表示输入完成，并将输入内容读取

Out 是一个system对象，是一个打印流，输出流OutputStream，是将内存中的数据输出到外部设备中，控制台。有两个方法print，println() 在末尾加上一个换行符。括号里面可以是任何类型

Err是一个system对象，是一个打印流PrintStream，输出流OutputStream，将内存中的数据输出到外不设备中，看你看股指太，有有两个方法print，println() 在末尾加上一个换行符

## 重定向标准I/O

对于in，是从键盘读取数据,是InputStream，对于out，err，是OutputStream的包装类，PrintStream,是将我们的数据输出到控制台中

重定向，使得不仅仅是从键盘读取数据，也不仅仅是从控制台输出数据。（从键盘读取和控制台输出是标准的输入输出流）

利用

System.setIn(InputStream in)

System.setOut(PrintStream out)

System.setErr(PrintStream out)

System.setIn(InputStream in)其中inputStream 是从哪里读取，

Int System.in.read(new b[215]) 读取后放到内存中的数组中。返回读取的字节数

## 总结

字节流

对象流：序列化

二进制流：输入输出二进制数据

打印流：重定向

字符流：

缓冲读写流