1. 二进制数据读写
2. 概述

文本格式的数据对于测试和调试非常方便，但是不像二进制格式那样高效，下面介绍二进制格式数据的传输。

1. DataOutput 与 DataInput接口

这两个接口分别用于二进制格式写数组，字符，boolean值和字符串。如writeInt总是将一个整数写出为4字节的二进制数量值，而不管它有多少位，这种对于给定类型的值使用的空间都是相同的，读写速度比解析文本要更快。

DataInputStream类实现了DataInput接口，可以使用该接口中的方法，为了从文件中读入二进制数据，可以将DataInputStream于某个字节源相结合，如下所示：

var in = new DataInputStream(new FileInputStream(“a.txt”));

1. 随机访问文件

RandomAccessFile类可以在文件中的任何位置查找或者写入文件，磁盘文件都是随机访问的，通过使用“r”表示读入，用“rw”表示读写作为构造器的第二个参数来制定读写方式，如下所示：

var in = new RandomAccessFile(“a.txt”, “r”);

当文件已RandomAccessFile打开时，这个文件不会被删除，且随机访问文件有一个表示下一个将被读入或写出的字节所处位置的文件指针，seek()方法可以用来设置该指针的位置，getFilePointer()方法返回该指针的当前位置。该类同时实现了DataInput和DataOutput接口。

1. ZIP文档

ZIP文档通常以压缩格式存储了一个或多个文件，每个ZIP文档都有一个头，包含诸如每个文件名字和所使用的压缩方法等信息。在Java中，使用ZipInputStream来读入ZIP文档，若需要浏览压缩文档中的每个单独项，使用getNextEntry()方法返回一个描述这些项的ZipEntry类型的对象，每次返回后需要调用closeEntry()方法来读入下一项，在读入最后一项之前不要关闭zin。如下所示：

var zin = new ZipInputStream(new FileInputStream(“a.zip”));

ZipEntry entry;

while((entry = zin.getNextEntry()) != null){

zin.closeEntry();

}

zin.close();

当要写入到ZIP文件中时，使用ZipOutputStream，对于希望放入ZIP文件中的每一项，都要创建一个ZipEntry对象，并使用putNextEntry()方法来写出新文件，并将文件数据发送到输出流中。如下所示：

var zout = new ZipOutputStream(new FileOutputStream(“test.zip”));

for all files{

var ze = new ZipENtry(fileName);

zout.putNextEntry(ze);

zout.closeEntry();

}

zout.close();

JAR文件只是一个带有清单项的ZIP文件，可以使用JarInputStream 和 JarOutputStream来读写清单项。