IO流与二进制

http://www.cnblogs.com/oubo/archive/2012/01/06/2394638.html
https://my.oschina.net/HerrySun/blog/652227
IO流学习总结博客
从Eclipse上我本来备注的时候复制过来，所以会有不足，缺陷
java.io.File类用于表示文件（目录）
File类只用于表示文件（目录）的信息（名称，大小等），不能用于文件内容的访问
RandomAccessFile java提供对 文件内容的访问，既可以读文件，也可以写文件
RandomAccessFile 可以随机访问文件，可以访问文件的任意位置
（1）java文件的模型
在硬盘是上的文件是byte byte byte是存储的，是数据的集合
（2）打开文件
有俩种模式"rw"（读写）"r"（只读）
RandomAccessFile raf=new RandomAccessFile(file,"rw")
文件指针，打开文件时指针在开头 pointer=0；
（3）写方法
raf.write(int)-->只写一个字节（后八位），同时指针指向下一个位置，准备再次写入
（4）读方法
int b=raf.read()-->//读一个字节
（5）文件读写完之后一定要关闭（Oracle官方说明）
IO流（输入，输出
字节流 字符流
1 inputStrem outputStrem

inputStrem抽象了应用程序了读取数据的方式
outputStrem抽象了应用程序了写出数据的方式
EOF=End 读到-1就读到了结尾
输入流基本方法
int b=in.read();读取一个字节无符号填充到int的低八位 -1是EOF
in.read(byte[] buf)读取数据填充到字节数组buf
in.read(byte[] buf,int start ,int size)读取数据到字节数组buf 从buf的start位置开始存放size长度的数据
输出流基本方法
out.write(int b)写入的是byte到流，b的低八位
out.write(byte[] buf)将buf字节数组写入到流
out.write(byte[] buf,int start,int size)

FileInputStream--具体实现了在文件上读取数据
FileOutputStream--实现了向文件中写入byte数据的方法
DataOutputStream /DataInputstream
对“流”一个扩展 可以更加方面的读取int,long,字符等类型数据
DataOutputStream

writeInt()/writedouble()/writeUTF()

BufferedInputStream&bufferedOutputStream
这俩个流类位IO提供了带缓冲区的操作，一般打开文件进行写入或读取操作时
都会带上缓冲，这种流模式提高了IO的性能
从应用程序中把输入放入文件，相当于将一缸水到入到另一个缸中：
FileOutputStream --》write()方法相当于一滴一滴的把水转移过去
DataOutputStream--》writeXxx()方法会方便一些，相当于一瓢一瓢的转移
BufferedOutputStream--》write方法更方便，相当于一瓢一瓢水，先放入桶中，再从桶中倒入到另一个当中

BufferedInputStream 缓冲的字节输入流
FileOutputStream 缓冲的自己输出流

Java中的流，可以从不同的角度进行分类。
按照数据流的方向不同可以分为：输入流和输出流。
按照处理数据单位不同可以分为：字节流和字符流。
按照实现功能不同可以分为：节点流和处理流。

字符流

1 编码问题
2 认识文本和文本文件
java的文本 （char）是16位无符号整数，是字符的unicode编码（双字节编码）
文件是byte、 byte byte...的数据序列
文本文件是文本（char）序列按照某种编码方案（utf-8 utf-16 gbk）序列化为byte
3字符流reader Writer --操作的是文本文件
字符的处理 一次处理一个字符
字符的底层依然是基本的字节序列
字符流的基本实现
InputStreamReader 完成byte流解析为char流 按照编码解析
OutputStreamWriter 提供char流到byte流，按照编码处理
FileReader/FileWriter 读/写
字节流加过滤 使其更加强大的功能
字符流的过滤器
BufferedReader ---> readline一次读一行
BufferedWriter/printWriter --->写一行
字节可以处理任何数据类型，字符要比字节能处理的类型要少。
通常在处理文本时优先使用字符流，其他的用字节流

字符流的底层就是字节流。而字符流主要是读取文本文件内容的，可以一个字符一个字符的读取，
也可以一行一行的读取文本文件内容。而字节流读取单位为byte.byte作为计算机存储最基本单位，
可以用字节流来读取很多其他格式的文件，比如图片视频等等。基于B/S和C/S的文件传输都可以采用字节流的形式。

stream结尾都是字节流,reader和writer结尾都是字符流
InputStream是所有字节输入流的祖先，而OutputStream是所有字节输出流的祖先。

Reader是所有读取字符串输入流的祖先，而writer是所有输出字符串的祖先。
InputStream，OutputStream,Reader,writer都是抽象类。所以不能直接new

3对象的序列化，反序列化
（1）对象序列化，进将Object转换成byte系列，反之叫对象的反序列化
（2）序列化流（ObjectOutputStream），是过滤流--writeObject
反序列化流（ObjectIntputStream）---readObject
（3）序列化接口（Serializable）
对象必须实现序列化接口，才能进行序列化，否则出现异常
这个接口，没有任何方法，只是一个标准
（4）序列化transient关键字
分析ArrayList序列化与反序列化的问题
（5）
序列化中子类和父类构造函数的调用问题
序列化是将对象状态转换为可保持或传输的格式的过程。
与序列化相对的是反序列化，它将流转换为对象。这两个过程结合起来，可以轻松地存储和传输数据。

下面是二进制方面的

按位与&
俩位全为1，结果才为1
位运算的特殊用法
（1）清零，如果想将一个单位清零，即使其全部二级制为0，只要与一个各位都为零的数值相等，结果为零
（2）取一个数为中间位 方法招一个数，对应的x要取的位，该数的对应位为1，其余为为零，此数与x进行“与运算”可以得到x中的指定位
之2：按位或|
只要有一个为1，结果就为1
或运算的特殊用法
常用来对一个数据的某些位置1
方法 ：找到一个数，对应的x要置1的位，该数的对应为1；其余为0；此数与x相或可使x的某些位置1
之3：异或运算^
两个相应位为”异“（值不同），则该位结果为1，否则为0
异或运算的特殊用途
（1）使特定为翻转 找一个数 ，对应的x要翻转的各位 ，该数的对应为1；其余位为0，此数与x对应位异或即可
（2）与0相或，保留原值
两个变量交换值的方法

1 借助第三个变量来实现
C=A;A=B;B=C
2 利用加减法实现俩个变量的交换
A=A+B;B=A-B;A=A-B;
3 用异或运算来实现，也是效率最高
原理：利用一个数异或本身等于0和异或运算符合交换率
如：A=A^B;B=A^B;A=A^B
之4：取反运算~
对一个二进制数按位取反，即将0变1.1变0
~1=0;~0=1;
之5：左移运算<<
将一个运算对象的各二进制全部左移若干位（左边的二级制丢弃，右边补零）
2<<1=4；
若左移时舍弃的高位不包含1，则每左移一位，相当于该数乘以2
-14（既二级制11110010）<<2=（11001000）
11（1011）<<=44
11（00000000 00000000 00000000 1011）（32bit）
之6 ：右移运算>>
将一个数的各二级制全部右移若干位，正数左补0，负数左补一，右边丢弃。操作数每右移一位，相当于该数除以2.
左补0 OR补1 得看被移数是正还是负
列一：1=4>>2
列2 ： -14（11110010）>>2
=-4（11111100）
之7：无符号右移>>>
各个位向右移 指定的个数。右移后左边空出的位用零来填充。移除右边的位被丢弃
列 -14>>>2
即 -14（111111111 11111111 111111111 11110010）>>>2
=（00111111 11111111 111111111 11111100）
= 103741820
负数以其正直的补码形式表示