一.IO流的概述

I：input,输入（读取）

O：output,输出（写入）

流：数据（字符，字节）

1.顶级父类们

|  |  |  |
| --- | --- | --- |
|  | 输入流 | 输出流 |
| 字节流 | 字节输入流  InputStream | 字节输出流  OutputStream |
| 字符流 | 字符输入流  Reader | 字符输出流  Writer |



二.

1.一切皆为字节

一切文件数据（文本，图片，视频等）在存储时，都是以二进制数字的形式保存，都是一个一个的字节，那么传输时一样如此。所以，**字节流可以传输任意文件数据**。在操作流的时候，我们要时刻明确，无论使用什么样的流对象，**底层**传输的始终为**二进制数据**。

2.字节输出流（OutputStream）

-java.io.OutputStream:此**抽象类**是表示**输出字节流**的所有类的**超类**。

定义了一些子类共享的成员方法：

-public void close():关闭输出流并释放与此流相关联的任何系统资源。

-public void flush():刷新此输出流并强制任何缓冲的输出字节被写出。

-public void write(byte[] b):将b.length字节从指定的字节数组写入此输出流。

-public void write(byte[] b, int off, int len):从指定的字节数组写入len字节，从偏移量off开始输出到此输出流。

-public abstract void write(int b):将指定的字节输出流。

-java.io.FileOutputStream extends OutputStream

FileOutputStream:文件字节输出流

作用：把内存中的数据写入到硬盘中的文件中。

构造方法：

- FileOutputStream(String name)创建一个像具有指定名称的文件中写入数据的输出文件流。

- FileOutputStream（File file）创建一个向指定File对象表示的文件中写入数据的文件输出流。

参数：**写入数据的目的地**

-String name：目的地是一个文件的路径

-File file：目的地是一个文件

构造方法的作用：

1.创建一个FileOutputStream对象。

2.会根据构造方法中传递的文件/文件路径，创建一个空的文件。

3.会把FileOutputStream对象指向创建好的文件。

写入数据的原理（内存🡪硬盘）

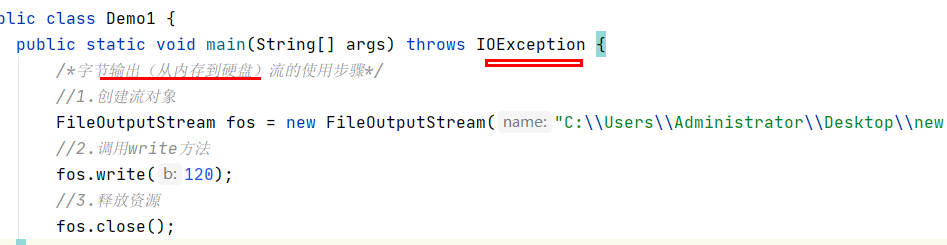
Java程序🡪JVM(Java虚拟机)🡪OS(操作系统)🡪OS调用写数据的方法🡪把输入写入文件中

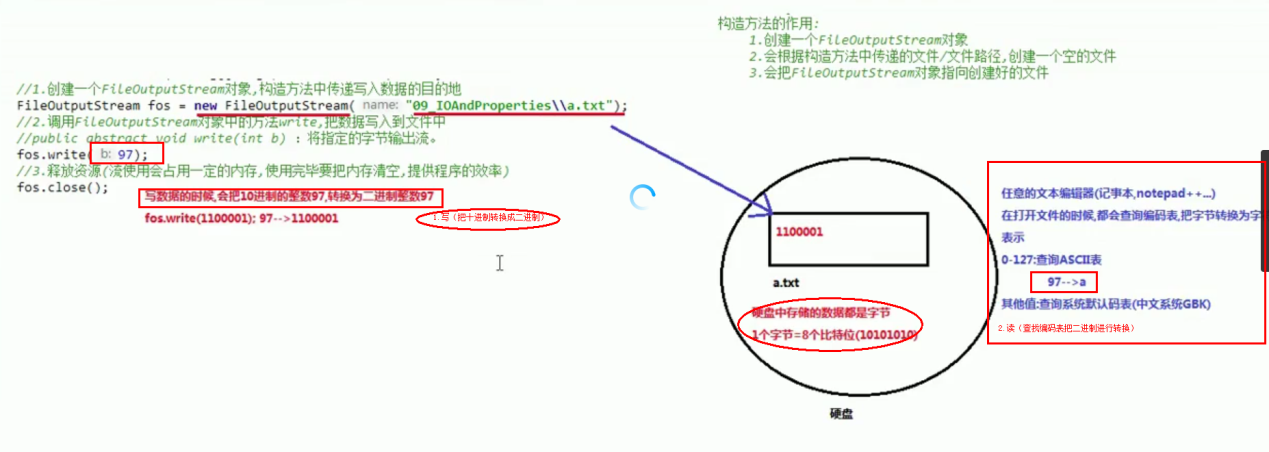
字节输出流的使用步骤（**重点**）：

1.创建FileOutputStream对象，构造方法中**传递写入数据的目的地**

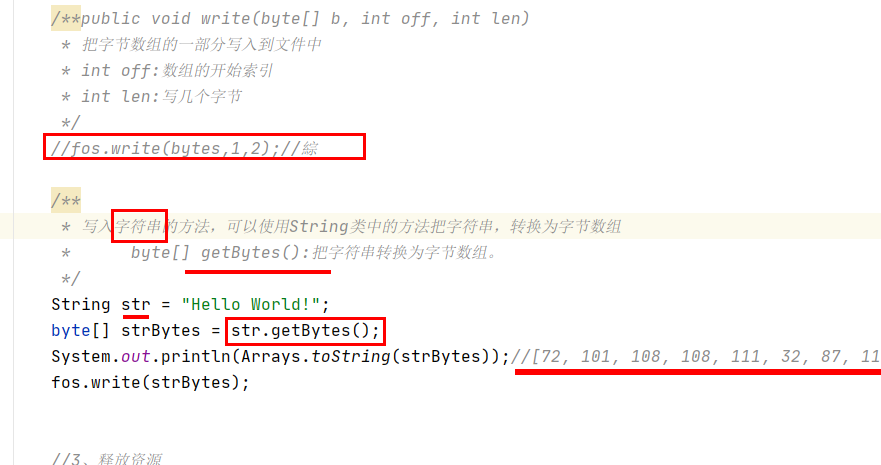
2.调用FileOutputStream对象中的**方法write**,把数据写入到文件中

3.**释放资源**（流使用会占用一定的内存，使用完毕要把内存清空，提高程序的效率）







****

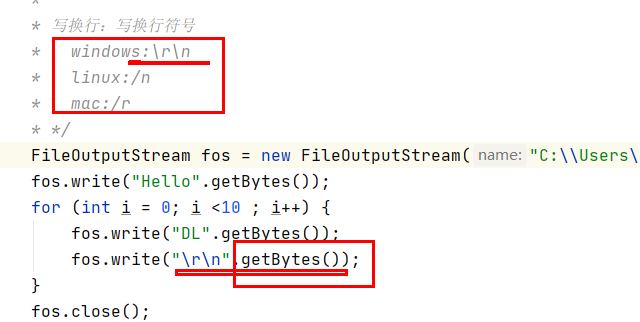
**GBK中两个字节是一个中文**

**UTF-8中三个字节是一个中文**

数据的追加写和换行写

*/\*追加写，续写：使用两个参数的构造方法就可以了  
\** ***FileOutputStream(String name,boolean append):*** *\* 创建一个像具有指定name的文件中写入数据的输出文件流。  
\** ***FileOutputStream(File file,boolean append):*** *\* 创建一个向指定File对象表示的文件中写入数据的文件输出流。  
\* 参数：String name,File file:写入数据的目的地  
\** **boolean append:追加写开关** *\* true：创建对象不会覆盖源文件，继续在文件的****末尾追加写数据*** *\* false：创建一个新文件，覆盖源文件  
\*  
\** ***写换行：写换行符号****\* windows:\r\n  
\* linux:/n  
\* mac:/r  
\* \*/*





3.字节输入流（InputStream）

此抽象类是表示字节输入流的所有类的超类。

定义了所有子类共性的方法：

-int read()从输入流中读取数据的下一个字节。

-int read(byte[] b)从输入流中读取一定数量的字节，并将其存储在缓冲区数组b中。

-void close()关闭此输入流并释放与该流没关联的所有系统资源。

-java.io.FileInputStream extends InputStream

**FileInputStream**:文件字节输入流

作用：**把硬盘文件中的数据，读取到内存中使用**

构造方法：

FileInputStream(String name)

FileInputStream(File file)

参数：读取文件的数据源

String name:文件的路径

File file：文件

构造方法的作用：

1.会创建一个FileInputStream对象

2.会把FileInputStream对象指向构造方法中要读取的文件

读取数据的原理（硬盘🡪内存）

Java程序🡪JVM—>OS—>OS读取数据的方法🡪读取文件

字节输入流的使用步骤（重点）：

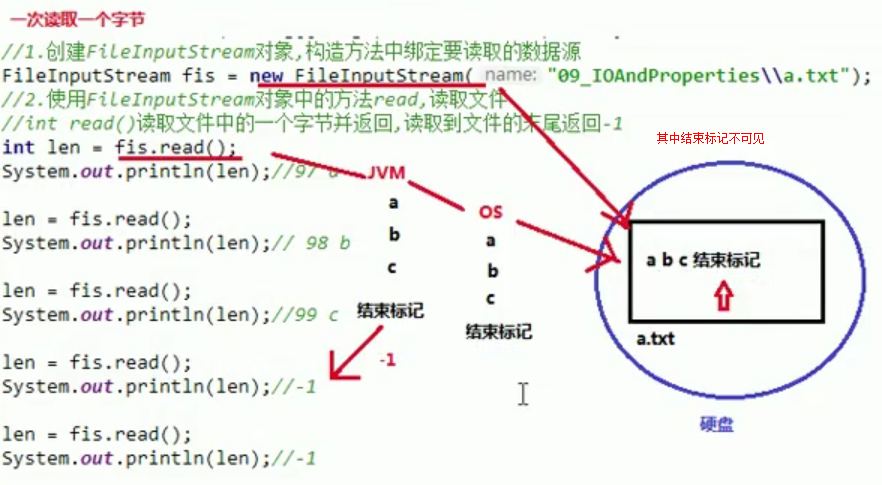
1.创建FileInputStream对象，构造方法中绑定要读取的数据源。

2.使用FileInputStream对象中的方法read，读取文件

3.使用FileInputStream对象中的close方法，关闭并释放资源。

****

字节输入流一次读取一个字节的原理



字节输入流一次读取多个字节的方法

-int **read**（**byte[] b**）从输入流中读取一定数量的字节，**并将其存储在缓冲区数组b中。**

明确两件事：

1.方法的**参数**byte[]的**作用**是什么？

**起到缓冲作用，存储每次读取到的多个字节**

**数组的长度一般定义为1024(1kb)或其整数倍**

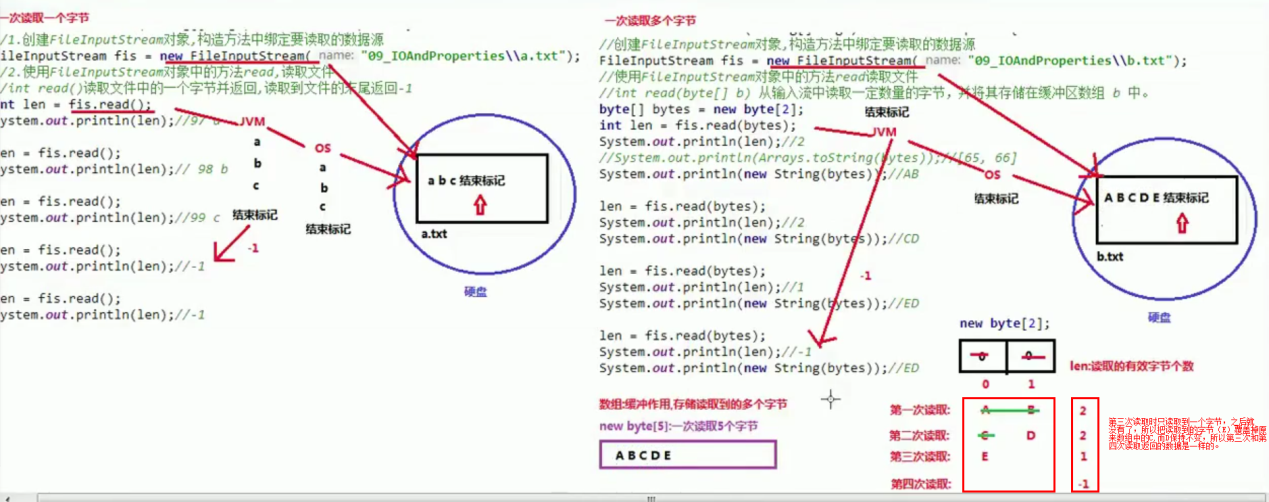
2.方法的**返回值int**是什么？

**每次读取的有效字节个数**

String类的**构造方法**：

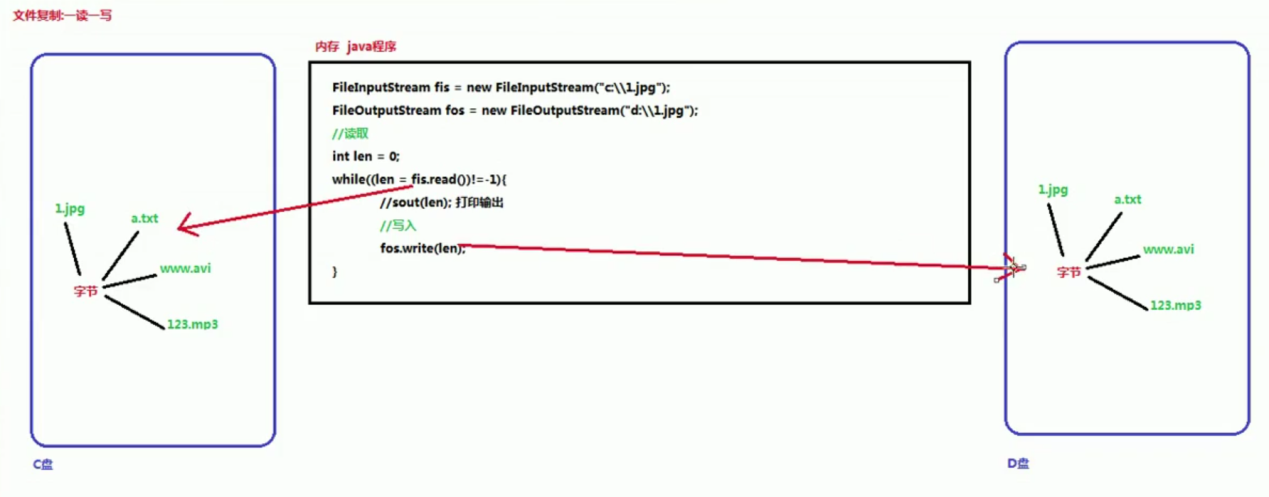
String（byte[] bytes）:把字节数组转换为字符串

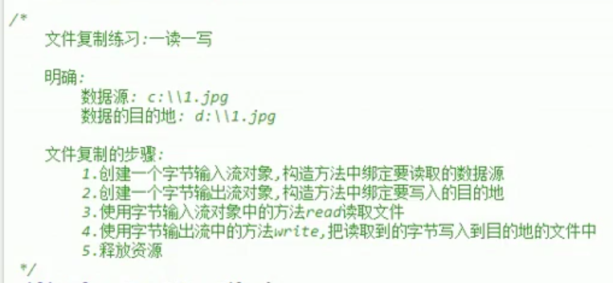
String（byte[] bytes,int offset,int length）:把字节数组的一部分转换为字符串，offset是数组的开始索引，length是转换的字节个数。





练习：文件复制







只要是文件，我们都可以使用字节流复制

使用字节流读取中文的问题

使用字节流读取中文文件：

1个中文：

GBK:占用2个字节

UTF-8:占用3个字节

所以会产生乱码问题，这时候就要用字符流读取了。

字符输入流（Reader）

-java-io.Reader：字符输入流。是**字符输入流**的最顶层的父类，定义了一些共性的成员方法。

-int read():读取当个字符并返回。

-int read(char[] cbuf):一次读取多个字符，**将字符读入数组**。

-void close()关闭该流并释放与之关联的所有资源。

-java.io.FileReader extends InputStreamReader extends Reader

FileReader:文件字符输入流

作用：把硬盘文件中的数据以字符方式读取到内存中。

构造方法：

-FileReader(String fileName)

-FileReader(File file)

参数：读取文件的数据源

String fileName:文件的路径

File file:一个文件

FileReader**构造方法的作用**：

1.创建一个FileReader对象

2.会把FileReader对象指向要读取的文件

字符输入流的使用步骤：

1.创建FileReader对象，构造方法中绑定要读取的数据源

2.使用FileReader对象中的方法read读取文件

3.释放资源



字符输出流（Writer）

-java.io.Writer：**字符输出流。**是所有字符输出流的最顶层父类，是一个抽象类。

共性的成员方法：

-void write(int c):写入单个字符。

-void write(char[] cbuf)写入**字符数组**。

-abstract void write(char[] cbuf,int off,int len)写入指定字符数组的部分。

-void write(String str):写入**字符串。**

-void write(String str,int off,int len)写入指定字符串的部分。

-void flush()刷新该流的缓冲。

-void close()关闭该流，但要先刷新它。

-java.io.FileWriter extends OutputStreamWriter extends Writer

FileWriter:文件字符输出流

作用：把内存中的字符数据写入到文件中

构造方法：

FileWriter（File file）:根据给定的File对象构造一个FileWriter对象。

FileWriter(String fileName):根据给定的文件名构造一个FileWriter对象。

参数：写入数据的目的地

String fileName:文件的路径

File file:是一个文件

构造方法的作用：

1.创建一个FileWriter对象

2.会根据构造方法中传递的文件/文件的路径，创建文件

3.会把FileWriter对象指向创建好的文件

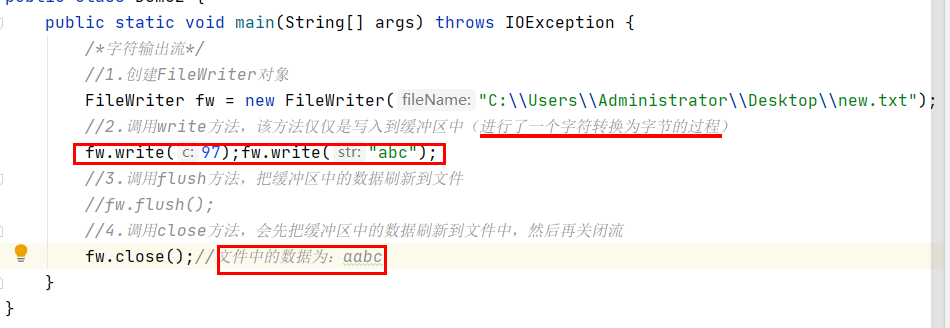
字符输出流的使用步骤：

1.创建一个FileWriter对象，构造方法中绑定要写入数据的目的地。

2.使用FileWriter中的方法write(),把数据写入到内存缓冲区中（字符转换为字节的过程）

3.使用FileWriter中的flush，把内存缓冲区中的数据，刷新到文件中。

4.释放资源（会先把内存缓冲区中的数据刷新到文件中（所以第三步可以省略））



-flush方法和close方法的区别：

-flush:刷新缓冲区，流对象可以继续使用。

-close:先刷新缓冲区，然后通知系统释放资源，该流对象不可以再被使用了。



字符输出流写数据的其他方法

-void write(char[] cbuf)写入字符数组。

-abstract void write(char[] cbuf,int off,int len)写入**指定字符数组的部分**。

-void write(String str):写入字符串。

-void write(String str,int off,int len)写入指定**字符串**的部分。



字符输出流的续写和换行（与字节流一样）

续写，追加写：使用两个参数的构造方法

-FileWriter(String fileName,boolean append)

-FileWriter(File file,boolean append)

参数：

参数一：写入数据的目的地

参数二：续写开关

-true：不会创建新的文件覆盖源文件 –false创建新的文件覆盖源文件

换行：

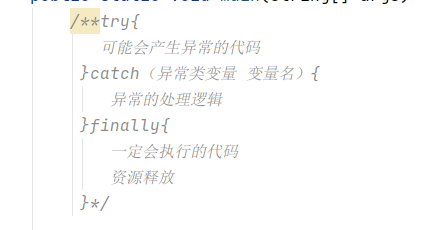
Windows：\r\n

Linux:/n

Mac:/r



使用try\_catch\_finally处理流中的异常





JDK7的新特性

在try的后边可以增加一个（），在括号中可以定义流对象

那么这个流对象的作用域就在try中有效

try中的代码执行完毕，会自动把流对象释放，不用写finally

格式：

/\*\*try(定义流对象){

可能会产生异常的代码

}catch（异常类变量 变量名）{

异常的处理逻辑\*/



JDK9的新特性

-try的前边可以定义流对象

在try后边的（）中可以直接引入流对象的名称（变量名）

在try代码执行完毕后，流对象也可以释放掉，不用写finally

A a = new A(); B b = new B();

try(a,b){

//可能出现异常的代码

}catch(异常类变量 变量名){

//异常处理逻辑

}



五、属性集（集合）

-java.util.Properties集合 extends Hashtable<K,V> implements Map<K,V>

Properties类表示一个持久的属性集。Properties可保存在流中或从流中加载。

Properties集合是一个**唯一和IO流相结合的集合**

可以使用Properties集合中的方法store，把集合中的临时数据，持久化**写入**到硬盘中存储。

可以使用Properties集合中的方法load，把硬盘中保存的文件（键值对），**读取**到集合中使用。

属性列表中每个**键**及其对应**值**都是一个**字符串。**

Properties集合是一个双列集合，**key和value默认都是字符串**

所以，Properties集合有一些操作字符串的特有方法

-Object **setProperty**(String key,String value)调用Hashtable的方法put。

-String **getProperty(**String key)通过key找到value值，此方法相当于Map集合中的get（key）方法。

-Set<**String**> **stringPropertyNames**()返回此属性列表中的**键集**，其中该键及其对应值是字符串，此方法相当于Map集合中的keySet方法。



Properties集合中的方法store

可以使用Properties集合中的方法store，把集合中的临时数据，持久化写入到硬盘中存储。

-void store(OutputStream out,String comments)

-void store(Writer writer,String comments)

参数：

OutputStream out：字节输出流，**不能**写入中文

Writer writer：字符输出流，**可以**写中文

String comments：注释：用来解释说明保存的文件是做什么用的，**一般不能使用中文**，会产生乱码，默认是Unicode编码，一般使用“”空字符串

使用步骤：

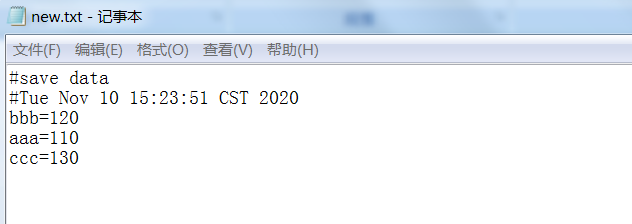
1.创建Properties集合对象，添加数据

2.创建字节输出流/字符输出流对象，构造方法中绑定要输出的目的地

3.使用Properties集合中的方法store,把集合中的临时数据，持久化写入到硬盘中存储

4.释放资源





Properties集合中的load方法

可以使用Properties集合中的方法load，把硬盘中保存的文件（键值对），读取到集合中使用。

-void load(InputStream inStream)

-void store(Reader reader)

参数：

InputStream inStream：字节输入流，**不能**读取含有中文的键值对

Reader reader：字符输入流，**能**读取含有中文的键值对

使用步骤：

1.创建Properties集合对象

2.使用Properties集合对象中的方法load读取保存键值对的文件

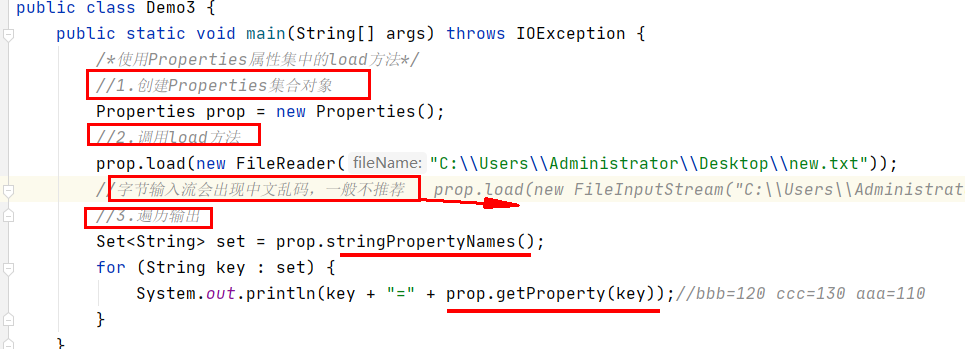
3.遍历Properties集合

注意：

1.存储键值对的文件中，键与值默认的连接符号可以是用=,空格（其他符号）

2.存储键值对的文件中，可以使用#进行注释，被注释的键值对不会再被读取

3.存储键值对的文件中，键与值默认都是字符串，不用再加引号



六、缓冲流

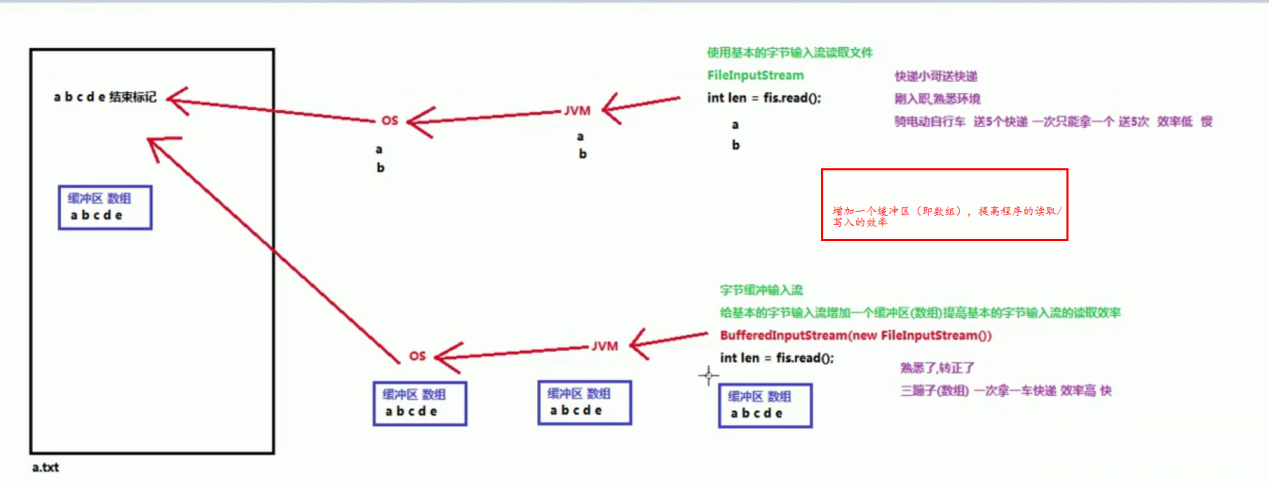
缓冲流都是在基本的流对象基础上创建而来的，相当于对基本流对象的一种增强。

缓冲流，也叫高效流，是对4个基本的**FileXxx流**的增强，所以也是4个流，分类：

**字节缓冲流**：BufferedInputStream, BufferedOutputStream

**字符缓冲流**：BufferedReader, BufferedWriter

缓冲流的基本原理，是在创建流对象时，会创建一个内置的默认大小的缓冲区数组，通过缓冲区读写，减少系统IO次数，从而提高读写的效率。



1.BufferedOutputStream字节缓冲输出流

-java.io.BufferedOutputStream extends OutputStream

继承自父类的共性成员方法：

-public void close():关闭输出流并释放与此流相关联的任何系统资源。

-public void flush():刷新此输出流并强制任何缓冲的输出字节被写出。

-public void write(byte[] b):将b.length字节从指定的字节数组写入此输出流。

-public void write(byte[] b, int off, int len):从指定的字节数组写入len字节，从偏移量off开始输出到此输出流。

-public abstract void write(int b):将指定的字节输出流。

构造方法：

BufferedOutputStream(OutputStream out):创建一个新的缓冲输出流，以将数据写入指定的底层输出流。

BufferedOutputStream（OutputStream out，int size）:创建一个新的缓冲区输出流，以将具有指定缓冲区大小的数据写入指定的底层输出流。

参数：

参数一：字节输出流。我们可以传递FileOutputStream,缓冲流会给FileOutputStream增加一个缓冲区，提高FileOutputStream的写入效率

参数二：指定缓冲流内部缓冲区的大小，不指定则默认

使用步骤：

1.创建FileOutputStream对象，构造方法中绑定要输出的目的地

2.创建BufferedOutputStream对象，构造方法中传递FileOutputStream对象对象，提高FileOutputStream对象效率。

3.使用BufferedOutputStream对象中的方法write，把数据写入到内部缓冲区。

4.使用BufferedOutputStream对象中的方法flush，把内部缓冲区中的数据，刷新到文件中。

5.释放资源（会先调用flush方法刷新数据，第四步可以省略）



2.BufferedInputStream字节缓冲输入流

继承自父类的成员方法

-int read()从输入流中读取数据的下一个字节。

-int read(byte[] b)从输入流中读取一定数量的字节，并将其存储在缓冲区数组b中。

-void close()关闭此输入流并释放与该流没关联的所有系统资源。

构造方法：

BufferedInputStream(InputStream in):创建一个BufferedInputStream，并保存其参数，即输入流in，以便将来使用。

BufferedInputStream（InputStream in，int size）:创建具有指定缓冲区大小的BufferedInputStream,并保存其参数in。

参数：

参数一：字节输入流。我们可以传递FileIutputStream,缓冲流会给FileIutputStream增加一个缓冲区，提高FileIutputStream的读取效率

参数二：指定缓冲流内部缓冲区的大小，不指定则默认

使用步骤：

1.创建FileIutputStream对象，构造方法中绑定要读取的数据源

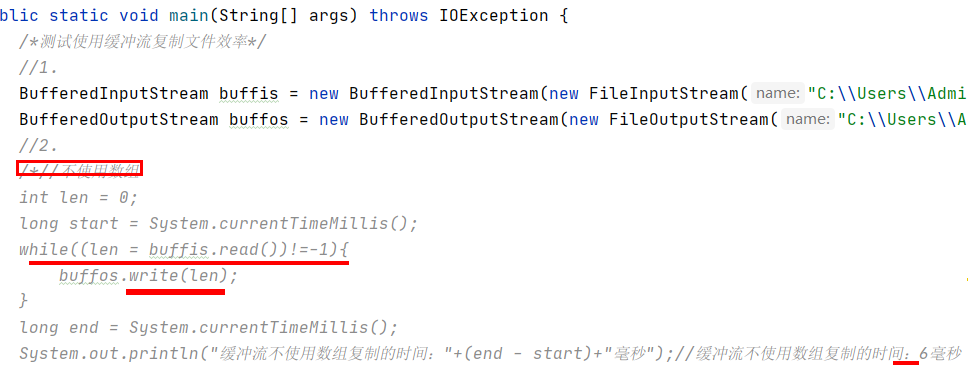
2.创建BufferedInputStream对象，构造方法中传递FileInputStream对象，提高FileInputStream对象效率。

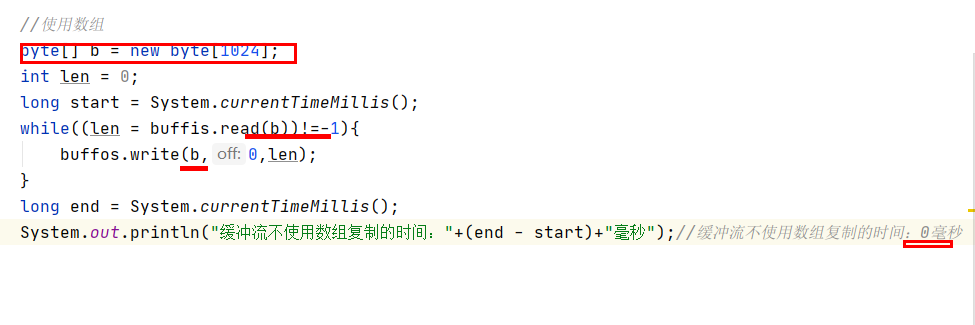
3.使用BufferedInputStream对象中的方法read，读取文件。

4.释放资源。



3.缓冲流的效率测试\_复制文件





4.BufferedWriter字符缓冲输出流

-java.io.BufferedWriter extends Writer

继承自父类的共性成员方法：

-void write(int c):写入单个字符。

-void write(char[] cbuf)写入字符数组。

-abstract void write(char[] cbuf,int off,int len)写入指定字符数组的部分。

-void write(String str):写入字符串。

-void write(String str,int off,int len)写入指定字符串的部分。

-void flush()刷新该流的缓冲。

-void close()关闭该流，但要先刷新它。

构造方法：

BufferedWriter（Writer out）创建一个使用默认大小输出缓冲区的缓冲字符输出流

BufferedWriter（Writer out,int sz）创建一个使用给定大小输出缓冲区的新缓冲区字符输出流。

参数：

Writer out:字符输出流。我们可以传递FilterWriter，缓冲流会给FileWriter增加一个缓冲区，提高FileWriter的写入效率。

int sz:指定缓冲区的大小，不写默认大小

**有一个特有的成员方法：**

-void **newLine**()**写一个行分隔符**。会根据不同的操作系统，获取不同的行分隔符。

换行：换行符号

Windows \r\n

Linux /n

Mac /r

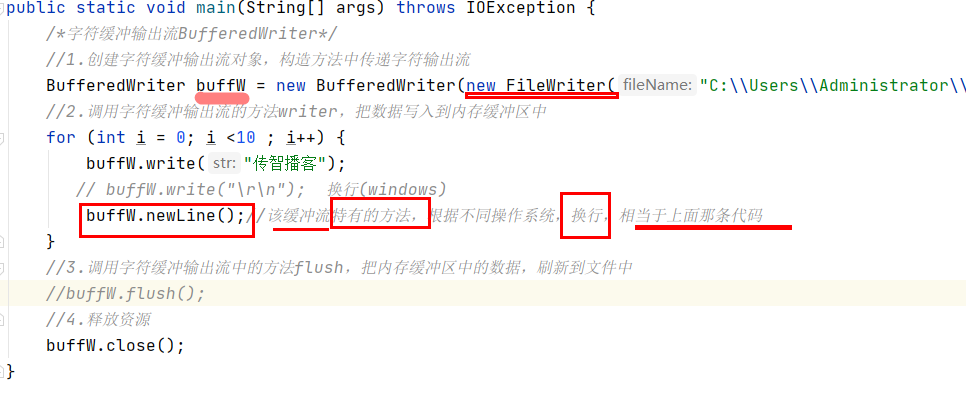
使用步骤：

1.创建字符缓冲输出流对象，构造方法中传递字符输出流

2.调用字符缓冲输出流的方法writer，把数据写入到内存缓冲区中

3.调用字符缓冲输出流中的方法flush，把内存缓冲区中的数据，刷新到文件中

4.释放资源



5.BufferedReader字符缓冲输入流

-java.io.BufferedReader extends Reader

继承自父类的共性成员方法：

-int read():读取当个字符并返回。

-int read(char[] cbuf):一次读取多个字符，将字符读入数组。

-void close()关闭该流并释放与之关联的所有资源。

构造方法：

BufferedReader（Reader in）创建一个使用默认大小输入缓冲区的缓冲字符输入流

BufferedReader（Reader in,int sz）创建一个使用给定大小输入缓冲区的新缓冲区字符输入流。

参数：

Reader in:字符输入流。我们可以传递FileReader，缓冲流会给FileReader增加一个缓冲区，提高FileReader的读取效率。

**特有方法：**

String **readLine**（）：**读取一个文本行**。即读取一行数据。行的终止符号：通过下列字符之一即可认为某行已终止。换行（“\n”），回车（“\r”）或者回车后直接跟着换行（“\r\n”）。

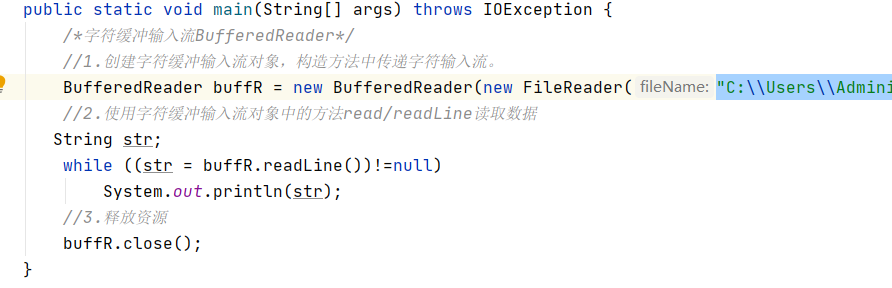
返回值：包含该行内容的字符串，不包含任何终止符，如果已到达流末尾，则返回null。

使用步骤：

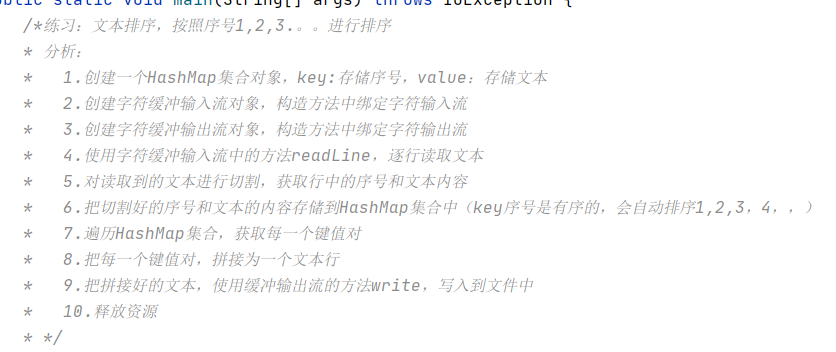
1.创建字符缓冲输入流对象，构造方法中传递字符输入流。

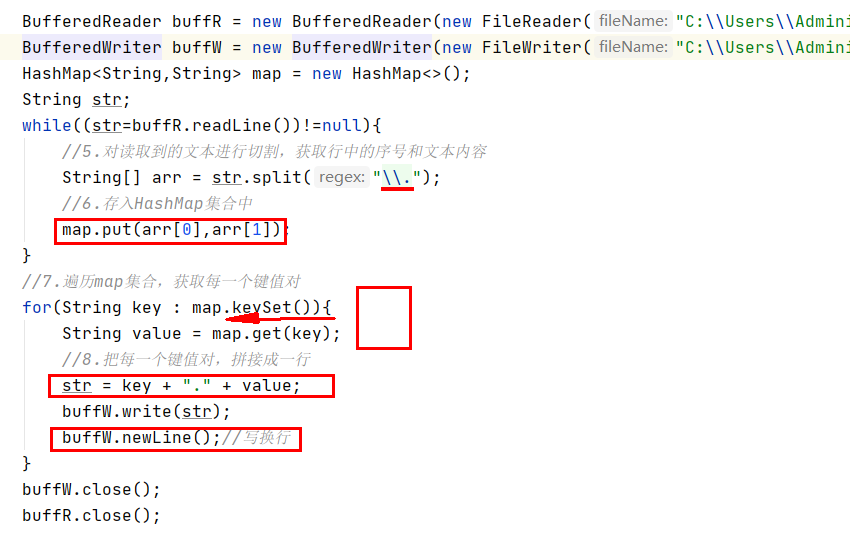
2.使用字符缓冲输入流对象中的方法read/readLine读取数据

3.释放资源



6.练习：文本排序





转换流

1.字符编码和字符集

编码：字符（能看懂的）--🡪字节（看不懂的）

解码：字节（看不懂的）--🡪字符（能看懂的）

字符编码（Character Encoding）:就是一套自然语言的字符与二进制数之间的对应规则。

编码表：生活中文字和计算机中二进制的对应规则。

字符集：（Charset）：也叫编码表，是一个系统支持的所有字符的集合，包括国家文字，标点符号，图形符号，数字等。

计算机要准确的存储和识别各种字符集符号，需要进行字符编码，一套字符集必然至少有一套字符编码。常见字符集有ASCII字符集，GBK字符集，Unicode字符集等。



可见，当指定了**编码**，它所对应的**字符集**自然就指定了，所以**编码**才是我们最终关心的。

2.转换流背景（产生原因）

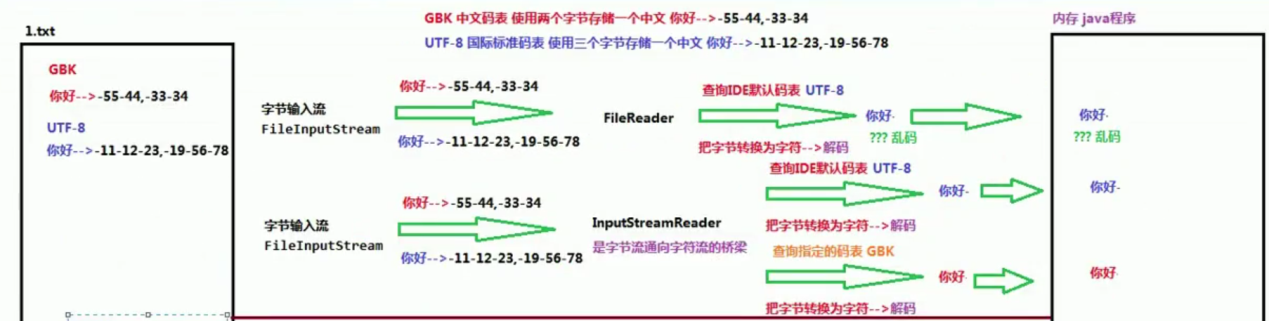
ANSI(系统默认编码，中文系统就是GBK)

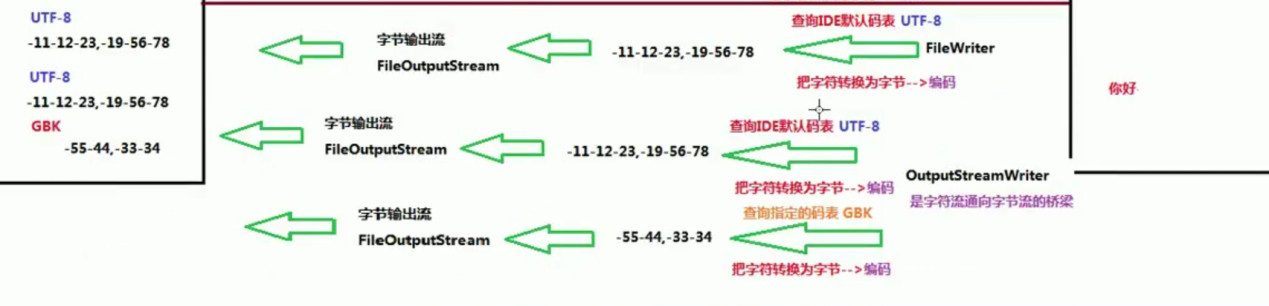
FileReader可以读取IDEA默认编码格式（UTF-8）的文件

FileReader读取系统默认编码（中文GBK）会产生乱码

这就需要用到转换流。

3.转换流的原理





4.OutputStreamWriter

-java.io. OutputStreamWriter extends Writer

OutputStreamWriter:是字符流通向字节流的桥梁，可以使用指定的charset，将要写入流中的字符编码成字节。

继承自父类的共性成员方法：

-void write(int c):写入单个字符。

-void write(char[] cbuf)写入字符数组。

-abstract void write(char[] cbuf,int off,int len)写入指定字符数组的部分。

-void write(String str):写入字符串。

-void write(String str,int off,int len)写入指定字符串的部分。

-void flush()刷新该流的缓冲。

-void close()关闭该流，但要先刷新它。

构造方法：

OutputStreamWriter（OutputStream out）：创建使用默认字符编码的OutputStreamWriter。

OutputStreamWriter（OutputStream out，String charsetName）：创建使用指定字符集的OutputStreamWriter。

参数：

OutputStream out：字节输出流，可以用来写转换之后的字节到文件中

String charsetName：编码表名称，**指定的编码表名称**，不区分大小写utf-8/UTF-8…不指定默认使用UTF-8

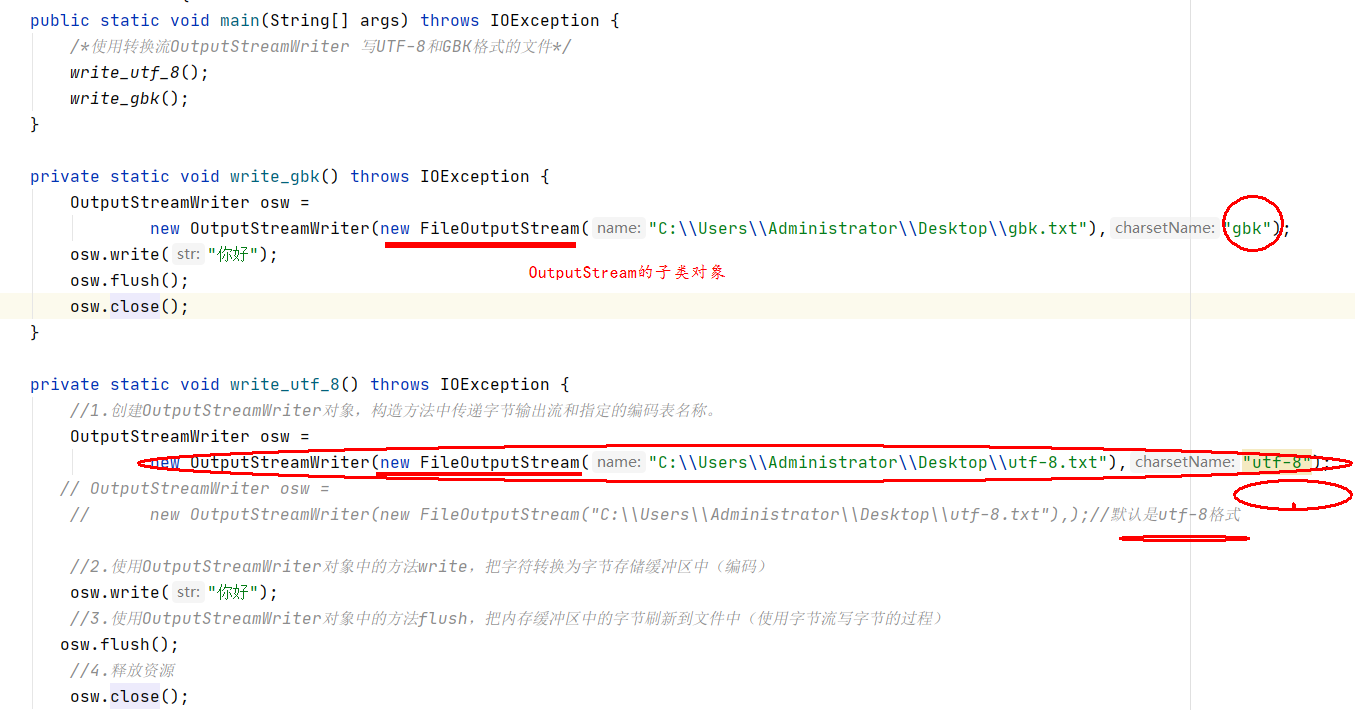
使用步骤：

1.创建OutputStreamWriter对象，构造方法中传递字节输出流和指定的编码表名称。

2.使用OutputStreamWriter对象中的方法write，把字符转换为字节存储缓冲区中（编码）

3.使用OutputStreamWriter对象中的方法flush，把内存缓冲区中的字节刷新到文件中（使用字节流写字节的过程）

4.释放资源



5.InputStreamReader

-java-io- InputStreamReader extends Reader

InputStreamReader:是字节流通向字符流的桥梁，可以使用指定的charset，将要读取流中的字节解码成字符。

继承自父类的共性成员方法：

-int read():读取当个字符并返回。

-int read(char[] cbuf):一次读取多个字符，将字符读入数组。

-void close()关闭该流并释放与之关联的所有资源。

构造方法：

InputStreamReader(InputStream in)创建一个使用默认字符集的InputStreamReader

InputStreamReader(InputStream in，String charsetName)创建一个使用指定字符集的InputStreamReader。

参数：

参数一：字节输入流，用来读取文件中保存的字节

参数二：编码表名称

使用步骤：

1.创建一个InputStreamReader对象，构造方法中传递字节输入流和指定的编码表名称

2.使用InputStreamReader对象中的方法read读取文件

3.释放资源

注意事项：

1构造方法中指定的编码表名称要和文件的编码相同，否则会发生乱码。



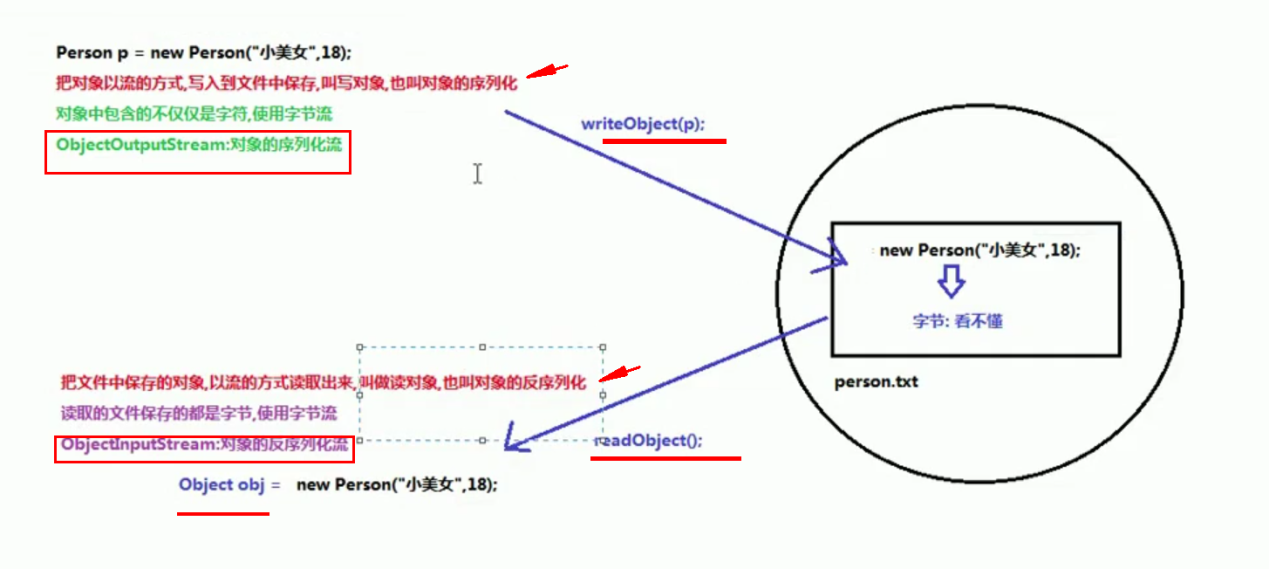
练习：转换文件编码

将GBK编码的文本文件，转换为UTF-8编码的文本文件。



（我如果用同一文件进行读取的写入的话，写入操作明明执行了，但是却没有内容，这是为什么？）

序列化流和反序列化流



1.java.io.ObjectOutputStream extends OutputStream

ObjectOutputStream:对象的序列化流。

作用：把对象以流的方式写入到文件中保存。

构造方法：

ObjectOutputStream（OutputStream out）创建写入指定OutputStream的ObjectOutputStream。

特有的成员方法：

void wirteObject(Object obj)将指定对象写入ObjectOutputStream

使用步骤：

1.创建ObjectOutputStream对象，构造方法中传递指定的字节输出流

2.使用ObjectOutputStream对象中的方法writeObject，把对象写入到文件中

3.释放资源。

NotSerializableException:没有序列化异常

当实例需要具有序列化接口时，抛出该异常，**序列化**运行时或实例的类会抛出此异常。参数应该为类的名称。

类通过实现java.io.Serializable 接口以启动其序列化功能。未实现此接口的类将无法使其任何状态序列化或反序列化。

反序列化流

-java.io.ObjectInputStream extends InputStream

作用：把文件中保存的对象，以流的方式读取出来使用。

构造方法：

ObjectInputStream（InputStream in）创建从指定InputStream读取的ObjectInputStream。

参数：字节输入流

特有的方法：

Object readObject()从ObjectInputStream读取对象。

使用步骤：

1.创建ObjectInputStream对象，构造方法中传递字节输入流

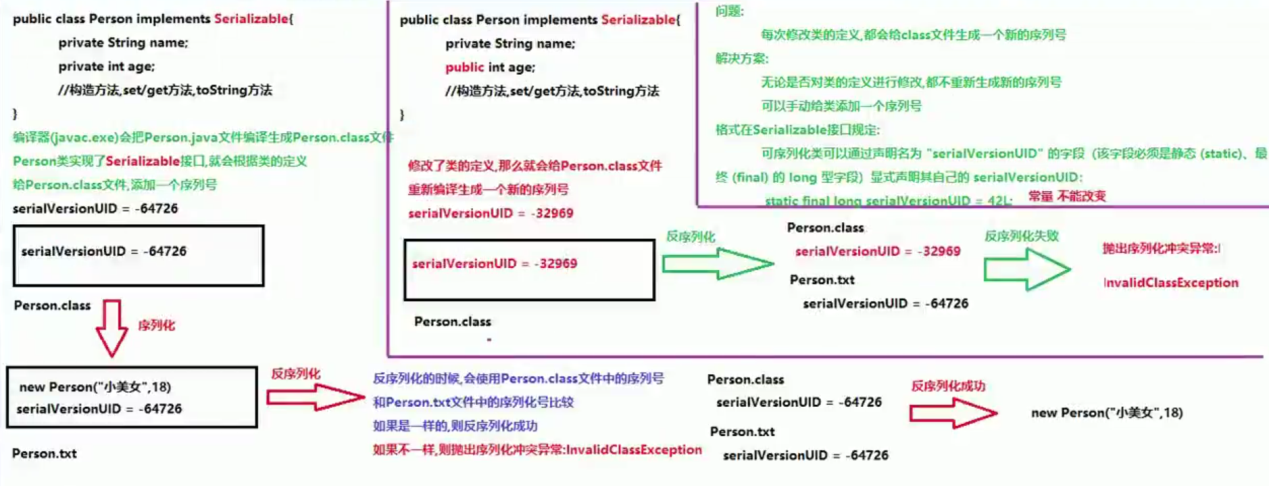
2.使用ObjectInputStream对象中的方法readObject读取保存对象的文件

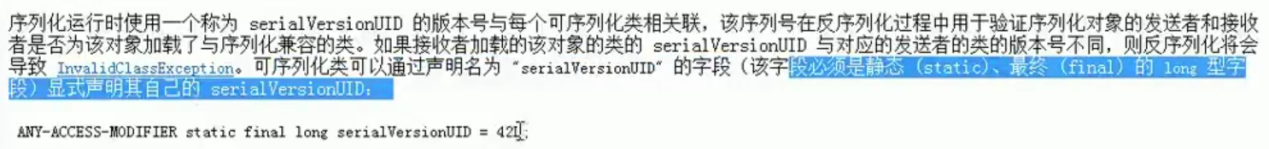
3.释放资源

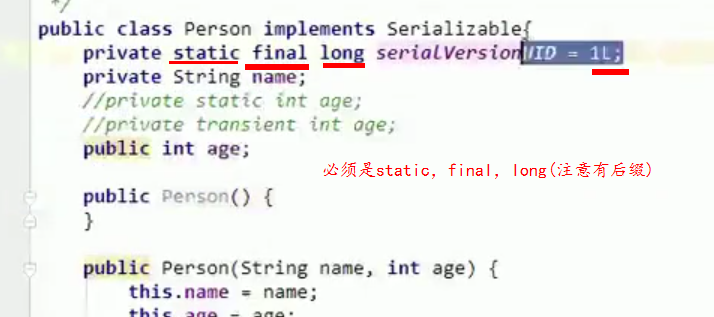
4.使用读取出来的对象（打印）



反序列化操作2（InvalidClassException异常\_原理）

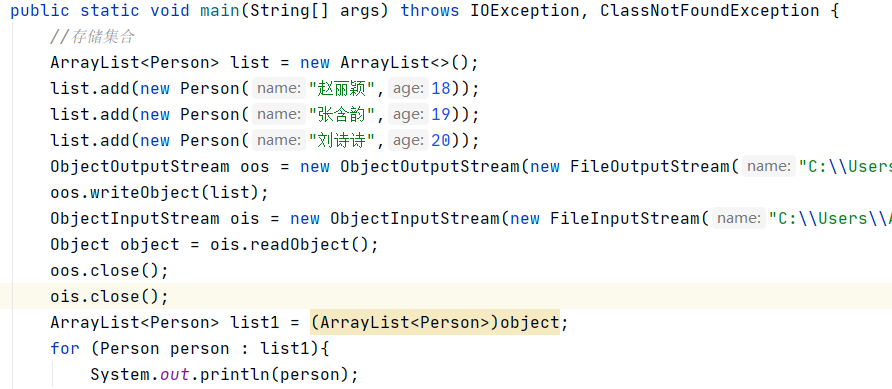


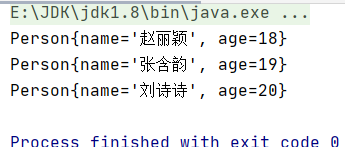




练习：序列化集合

当我们想在文件中保存多个对象的时候，可以把多个对象存储到一个集合中，对集合进行序列化和反序列化。



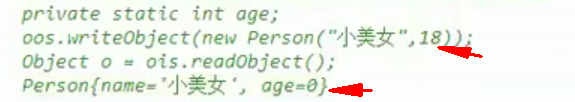


关键字transient关键字（又叫做瞬态关键字）

-static关键字：静态关键字

静态优先于非静态加载到内存中（静态优先于对象进入到内存中）

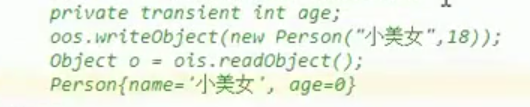
被**static修饰**的 ***成员变量* 不能被序列化**，因为**序列化的都是对象**。



-transient关键字：瞬态关键字

被*transient关键字修饰的成员变量，不能被序列化*。

（和被static修饰效果相同，但是含义不能，二者没有关系）



打印流（PrintStream）

-java.io.PrintStream:

PrintStream为其他输出流添加了功能，使他们能够方便地打印各种数据值表示形式。

PrintStream的三个特点：

1.只负责数据的输出，不负责数据的读取。

2.与其他输出流不同，PrintStream永远不会抛出IOException

3.有特有的方法print,println

-void print(任意类型的值)

-void println(任意类型的值并换行)

构造方法：

PrintStream（File file）：输出的的目的地是一个文件

PrintStream（OutputStream out）：输出的目的地是一个字节输出流

PrintStream（String fileName）：输出的目的地是一个文件路径

PrintStream extends OutputStream

继承自父类的成员方法：

-public void close():关闭输出流并释放与此流相关联的任何系统资源。

-public void flush():刷新此输出流并强制任何缓冲的输出字节被写出。

-public void write(byte[] b):将b.length字节从指定的字节数组写入此输出流。

-public void write(byte[] b, int off, int len):从指定的字节数组写入len字节，从偏移量off开始输出到此输出流。

-public abstract void write(int b):将指定的字节输出流。

注意：如果使用继承自父类的write方法写数据，那么查看数据的时候就会查询编码表，97-》a；如果要使用自己特有的方法print/println方法写数据，写的数据原样输出97-->97



