3 字符流

3.1 为什么会出现字符流

由于字节流操作中文不是特别的方便，所以Java就提供字符流

- 字符流 = 字节流 + 编码表

用字节流存储文本文件时，文本文件也会有中文，但是没有问题，原因是最终底层操作会自动进行字节拼接成中文，如何识别是中文呢?

- 汉字在存储时候，无论哪种编码存储，第一个字节都是负数

3.2 编码表

基础知识

- 计算机存储的信息都是二进制表示的，所有屏幕上的字符等都是二进制转换后得到的
- 按规则存入计算机， → 编码
- 解析出来 → 解码
- 字符编码 就是一套自然语言和二进制数之间的对应规则

字符集

- 是一个系统支持的所有字符的集合，包括各国家文字、标点符号、图形符号、数字等
- 计算机要准确存储和识别各种字符集符号，就要进行字符编码，一套字符必然至少有一套字符编码。常见ASCII字符集，GBXXX字符集，Unicode 字符集等

ASCii 字符集

- ASCII(American Standard Code for Information Interchange,美国信息交换标准代码):是基于拉打母的一套电脑编码系统,于显现代英语，主要包括控制字符(回车键、退格、换行键等)和可显字符(英文大小写字符、阿拉伯数字和西文符号
- 基本的ASCII字符集，使用7位表示一个字符，共128字符。 ASCII的扩 展字符集使用8位表示一个字符,共256字符，方便支持欧洲常用字符。是一个系统支持的所有字符的集合，包括各国家文字、标点符号、图形符号、数字等，

GBXXX字符集

- GB2312：简体中文码表。一个小于127的字符的意义与原来相同，但两个大于127的字符连在一起时，就表示一个汉字，这样大约可以组合了包含7000多个简体汉字，此外数学符号、罗马希腊的字母、日文的假名等都编进去了，连在ASCII里本来就有的数字、标点、字母都统统重新编了两个字节长的编码，这就是常说的"全角"字符，而原来在127号以下的那些就叫"半角”字符了

- GBK：最常用的中文码表。是在GB2312标准基础上的扩展规范，使用了双字节编码方案，共收录了21003个汉字，完全兼容GB2312标准，同时支持繁体汉字以及日韩汉字等
- GB18030：最新的中文码表。收录汉字70244个,采用多字节编码,每个字可以由1个、2个或4个字节组成。支持中国国内少数民族的文字,同时支持繁体汉字以及日韩汉字等

Unicode

- 为表达任意语言的任意字符而设计，是业界的一种标准,也称为统一码、 标准万国码。它最多使用4个字节的数字来表达每个字母、符号，或者文字。有三种编码方案, UTF-8、 UTF-16和UTF32。 最为常用的UTF- 8编码
- UTF- 8编码：可以用来表示Unicode标准中任意字符，它是电子邮件、网页及其他存储或传送文字的应用中，优先采用的编码。互联网工程工作小组(IETF) 要求所有互联网协议都必须支持UTF -8编码。它使用一至四个字节为每 个字符编码
- 编码规则:
  128个US-ASCII字符，只需一个字节编码
  拉丁文等字符，需要二个字节编码
  大部分常用字(含中文)，使用三个字节编码
  其他极少使用的Unicode辅助字符，使用四字节编码
- 小结：采用何种规则编码，就要采用对应规则解码，否则就会出现乱码

3.3 字符串中的编解码问题

编码

- byte[] getBytes0：使用平台的默认字符集将该String编码为一系列字节，将结果存储到新的字节数组中
- byte[] getBytes(String charsetName)：使用指定的字符集将该String编码为-系列字节,将结果存储到新的字节数组中

解码:

- String(byte[] bytes)：通过使用平台的默认字符集解码指定的字节数组来构造新的String
- String(byte[] bytes, String charsetName)：通过指定的字符集解码指定的字节数组来构造新的String

3.4 字符流中的编码解码问题

字符流抽象基类

- Reader 字符输入流的抽象类
- Writer 字符输出流的抽象类

字符流中和编码解码问题相关的两个类

- InputStreamReader
- OutputStreamWriter

3.5 字符流写数据的5种方式

方法名	说明
void write(int c)	写一个字符
void write(char[] cbuf)	写入一个字符数组
void write(char[] cbuf, int off, int len)	写入字符数组的一部分
void write(String str)	写一个字符串
void write(String str, int off, int len)	写一个字符串的一部分

方法名	说明
flush()	刷新流，还可以继续写数据
close()	关闭流，释放资源，但是在关闭之前会先刷新流。一旦关闭，就不能再写数据

3.6 字符流读数据的两种方式

方法名	说明
int read()	一次读一个字符数据
int read(char[] cbuf)	一次读一个字符数组数据

3.7字符缓冲流

> 字符缓冲流

- BufferedWriter：将文本写入字符输出流，缓冲字符，以提供单个字符,数组和字符串的高效写入，可以指定缓冲区大小，或者可以接受默认大小。默认值足够大,可用于大多数用途
- BufferedReader:从字符输入流读取文本，缓冲字符，以提供字符，数组和行的高效读取，可以指定缓冲区大小，或者可以使用默认大小。默认值足够大，可用于大多数用途

> 构造方法:

- BufferedWriter(Writer out)
- BufferedReader(Reader in)

3.8 字符缓冲流特有功能

> BufferedWriter

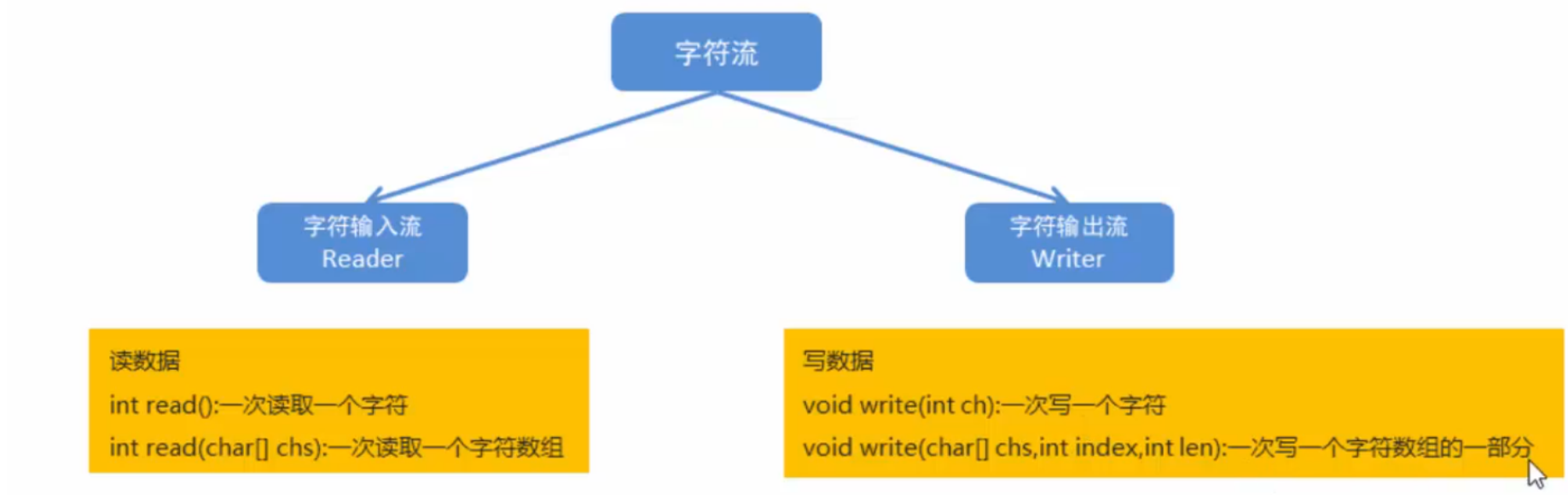
- `void String newLine()` 写一行行分隔符。具体由系统属性定义

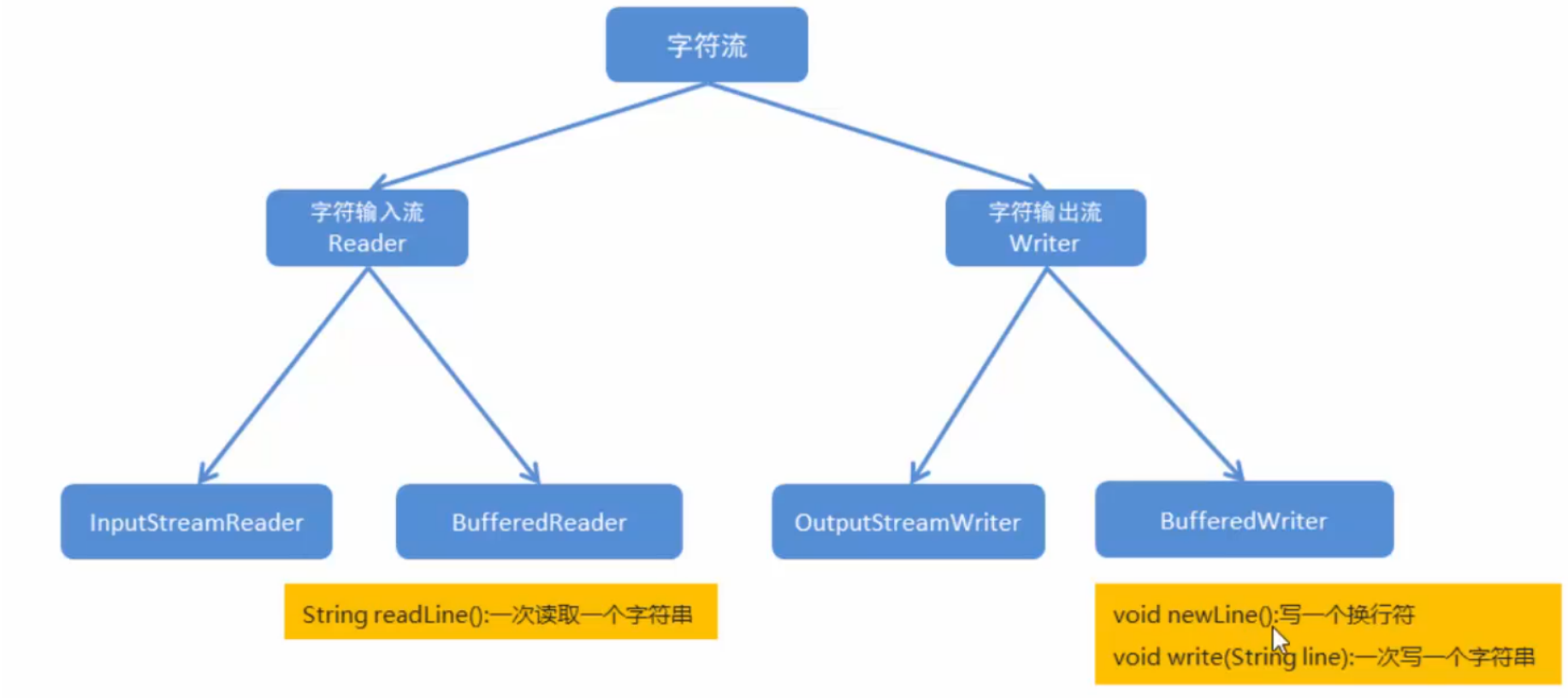
BufferedReader

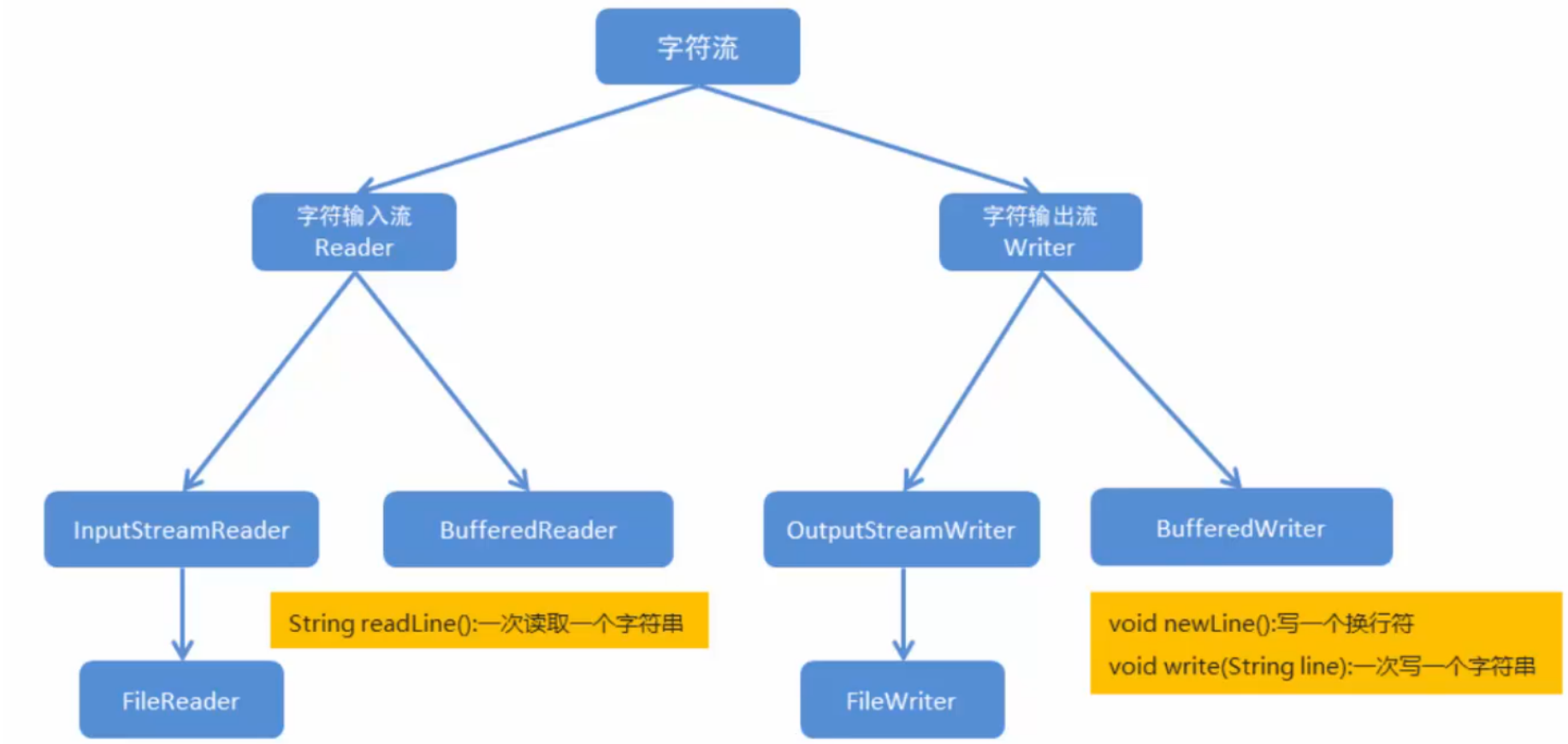
- `public String readLine()` 读一行文字，结果包含行的内容的字符串，不包括任何行终止字符，若达到结尾，则为null

3.9 IO流小结

小结：字节流可以复制任意文件数据，有4种方式一般采用字节缓冲流一次读写一个字节数组的方式







小结：字符流只能复制文本数据，有5种方式，一般采用字符缓冲流的特有功能

3.10 复制文件的异常处理

try...catch...finally的做法：

```
try{
        可能出现异常的代码;
}catch(异常类名 变量名){
        异常的处理代码;
}finally{
        执行所有清除操作;
}
```

JDK7改进方案：

```
try(定义流对象){
        可能出现异常的代码;
}catch(异常类名 变量名){
        异常的处理代码;
}
```

自动释放资源