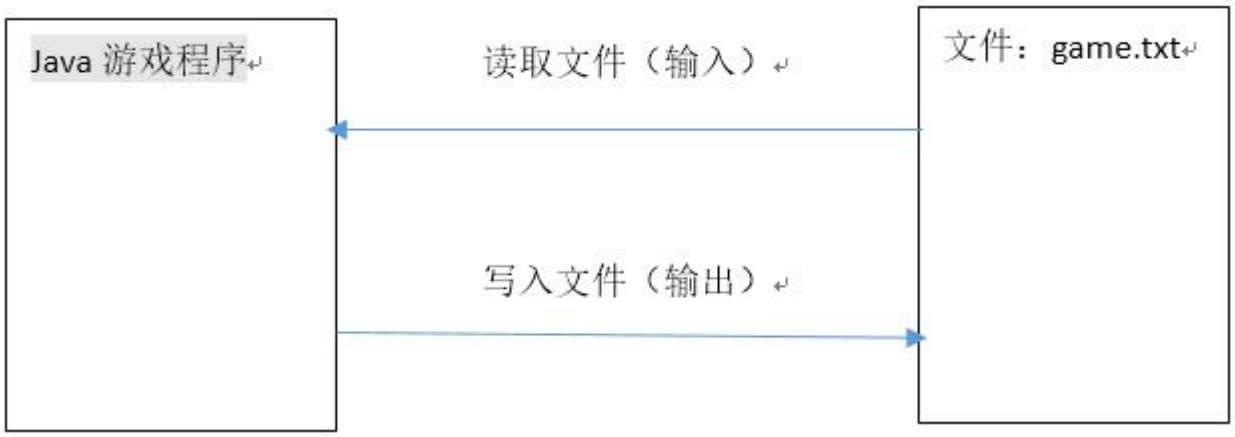
IO 概述

IO（Input/Output）:输入和输出，指的是某个设备或环境进行数据的输入或者输出。例如：键盘的输入，再比如显示器就是输出设备，输出图像。
对于 java 来说输入输出问题，java 将它抽象成流对象来解决。
以游戏程序为中心读取文件就是输入，写入文件是输出。



IO 流在 java 中从输入输出角度分类：

1.输入流

2.输出流

IO 流在 java 中从数据的角度来分类：

1.字符流

文本，我们能读的懂的都可以认为是字符流。比如：文章，java 文件等等

2.字节流

二进制的数据，这种数据一般用文本打开我们读不懂。比如，图片文件，mp3 文件，等等。

1.字符流

文本，我们能读的懂的都可以认为是字符流。比如：文章，java 文件等等

字符输入流的超类：

Reader： 子类 FileReader，BufferedReader

字符输出流的超类

Writer：子类 FileWriter，BufferedWriter

2.字节流

二进制的数据，这种数据一般用文本打开我们读不懂。比如，图片文件，mp3 文件。

字节输入流的超类：

InputStream： 子类 FileInputStream

字节输出流的超类

OutputStream：子类 FileOutputStream

字符输出流

使用字符流向一个文件输入 helloworld。

步骤：　　1.创建文件

2.创建输出流对象

3.把流指向指定的文件

4.释放资源

FileWriter 的普通构造器

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。

Writer 的 flush 方法，字符流需要清空缓冲区，当大量写入文件数据时要合理的使用 flush。

FileWriter 的追加构造器

FileWriter(File file, boolean append) 根据给定的 File 对象构造一个 FileWriter 对象。
FileWriter(String fileName, boolean append) 根据给定的文件名以及指示是否附加写入数据的 boolean 值来构造 FileWriter 对象。

输出换行

把文本写入文件中\n 代表换行

问题是不同的环境下换行的方式也不一样

Windows: \r\n

Linux:\n

Mac:\r

FileWriter 的写入功能

void	**write**(char[] cbuf) 写入字符数组。
abstract void	**write**(char[] cbuf, int off, int len) 写入字符数组的某一部分。
void	**write**(int c) 写入单个字符。
void	**write**(String str) 写入字符串。
void	**write**(String str, int off, int len) 写入字符串的某一部分。

字符输入流

FileReader 的构造器和读取

FileReader(File file) 在给定从中读取数据的 File 的情况下创建一个新 FileReader。
FileReader(String fileName) 在给定从中读取数据的文件名的情况下创建一个新 FileReader。

范例：读取文件 helloworld.txt 打印到控制台
分析：
1.创建出入流对象 FileReader
2.读取数据
3.关闭输入流
读取字符的方法

int	**read**() 读取单个字符。
int	**read**(char[] cbuf) 将字符读入数组。
abstract int	**read**(char[] cbuf, int off, int len) 将字符读入数组的某一部分。

使用字符流来做文件的拷贝
范例：把一个 java 文件拷贝到项目根目录。
分析：　1.创建字符输入流的对象
　　2.创建字符输出流对象
　　3.把输出流输入的数据写入输出流中
　　4.关闭资源

BufferedReader

从字符输入流中读取文本，缓冲各个字符，从而实现字符、数组和行的高效读取

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。

扩展方法 readLine 读取一行数据

BufferWriter

将文本写入字符输出流，缓冲各个字符，从而提供单个字符、数组和字符串的高效写入。

protected	**Writer**() ↓ 创建一个新的字符流 writer，其关键部分将同步 writer 自身。

高效缓冲区的输出流的扩展方法：
newline()可以换行

字节流概述

字节输入流：InputStream：常用子类 FileInputStream
字节输出流：OutputStream:常用子类 FileOutputStream

OutputStream

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。
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。
FileOutputStream(String name) ↓ 创建一个向具有指定名称的文件中写入数据的输出文件流。
FileOutputStream(String name, boolean append) ↓ 创建一个向具有指定 name 的文件中写入数据的输出文件流。

范例：使用字节流向文件中写入”helloworld”
分析：使用三种写入方法

void	**write**(byte[] b) ↓ 将 b.length 个字节从指定 byte 数组写入此文件输出流中。
void	**write**(byte[] b, int off, int len) ↓ 将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此文件输出流。
void	**write**(int b) ↓ 将指定字节写入此文件输出流。

InputStream

FileInputStream(File file) 通过打开一个到实际文件的连接来创建一个 FileInputStream，该文件通过文件系统中的 File 对象 file 指定。
FileInputStream(String name) 通过打开一个到实际文件的连接来创建一个 FileInputStream，该文件通过文件系统中的路径名 name 指定。

读取字节流方法

int	**read**() 从此输入流中读取一个数据字节。
int	**read**(byte[] b) 从此输入流中将最多 b.length 个字节的数据读入一个 byte 数组中。
int	**read**(byte[] b, int off, int len) 从此输入流中将最多 len 个字节的数据读入一个 byte 数组中。

字节流的高效缓冲区

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。

字符流和字节流的转换桥梁

字节流转换字符流

java.io
类 InputStreamReader

java.lang.Object
  └ java.io.Reader
    └ **java.io.InputStreamReader**

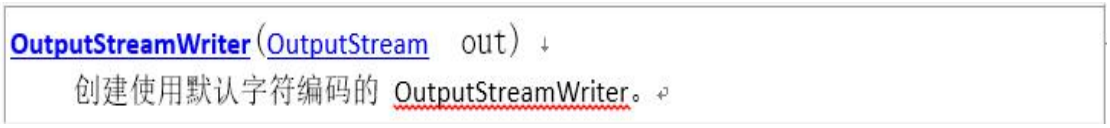
InputStreamReader(InputStream in) 创建一个使用默认字符集的 InputStreamReader。

字符流转换成字节流

OutputStreamWriter 是字符流通向字节流的桥梁

java.io
类 OutputStreamWriter

java.lang.Object
  └ java.io.Writer
    └ **java.io.OutputStreamWriter**



打印流

打印流：只做输出没有输入
打印流分为字节打印流和字符打印流
PrintWriter：字符打印流
特点

1.可以打印各种数据类型。

2.封装了字符输出流，还可以字符流和字节流的转换

3.可以使用自动刷新，则只有在调用 println、printf 或 format 的其中一个方法时才可能完成此操作

4.可以直接向文件中写数据

Properties 类

Properties 类表示了一个持久的属性集。Properties 可保存在流中或从流中加载。属性列表中每个键及其对应值都是一个字符串。
特点：
1.继承于 HashTable，是线程安全的键值对存储结构
2. Properties 可保存在流中或从流中加载
3. 只能保存字符串的键值对

序列化流（对象流）

如果想序列化某个对象，那么这个对象所对应的类必须实现
类通过实现 java.io.Serializable 接口以启用其序列化功能，在序列化的过程中要手动指定要序列化的类的的 serialVersionUID，这样可以在类改变后依然可以反序列化，否则会报错

字符编码

存储:
在计算机中存储字符都是存储的字符所对应的数值以二进制的形式表示。
展示：
去相关的编码表中去查找该值（存储的值）所对应的字符。

常见的：

ASCII 表：

用 7bit 来表示存储数据

ISO-8859-1:拉丁码表

用 8bit 来表示

GB2312:

简体中文编码（国标码）