# 简要

以下所有代码都来自于JDK1.8.

# IO

学习Java IO，主要的方式通过阅读java.io包下的源码，阅读源码之前最主要的工作是理清包结构，各个class之间的继承关系。方法是在IDEA ide中，在java.io包上选择diagram即可查看class继承关系。由于这个继承关系截图太大，此处不便贴图。

## 概述

通过class继承关系图，可以直观的各个class的层级关系，按照这种层级关系逐层阅读。

## 第一层---顶层接口

第一层级包括两个接口，属于最顶层接口，其他class会实现其中一个或者两个接口。

* Flushable：只用于输出流，包含一个flush方法，将缓存中的stream刷新到对应的destination。
* Closeable：输入输出流都可以实现，提供一个close方法，用于关闭对应资源。

## 第二层---抽象类

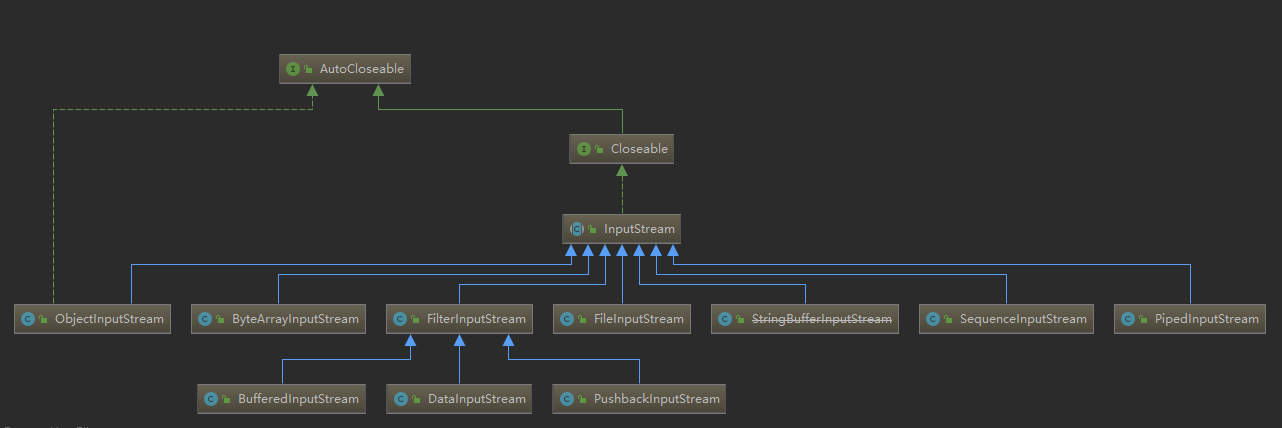
Java IO包含两种类型的流，一种是字节流，一种是字符流，每一种流有包含两种形式，分别是输入和输出，因此Java IO一共包含四种具体的流类型，分别是：

* 输入字节流：java.io包下\*InputStream格式的类
* 输出字节流：java.io包下\*OutputSteam格式的类
* 输入字符流：java.io包下\*Reader格式的类
* 输出字符流：java.io包下\*Writer格式的类

第二层主要包含四个顶层抽象class，分别是InputSteam，OutputSteam，Reader和Writer，四个类分别作为对应流类型的抽象类。

### 输入字节流

输入字节流的主要成员包括如下图，按照这种类继承关系这个阅读代码。



所有的字节流都是在InputStream的基础上增加额外功能

#### BufferedInputStream

通过Buffer修饰的的BufferedInputStream有两层含义，一是每次从指定的输入字节流中读取固定长度字节数存放到buffer中，二是这个类的大部分方法被synchronized关键字修饰。

主要操作是填充方法fill，如果buffer的数据已被读取，需要重新从inputstream中读取并存放在buffer。

#### ByteArrayInputStream

这个类就是将一个byte数组包装成一个InputStream，利用InputStream提供的接口从数组中读取字节数据。

该类提供的read操作被synchronized关键字修饰。

#### DataInputStream

这个类主要是将java内置类型数据读入内存。read方法是非线程安全的。

#### FileInputStream

读取文件的InputStream。非线程安全的。

#### FilterInputStream

该类是InputStream的包装类，包含一个InputStream实例对象的引用，所有接口都是通过这个引用对象调用对应接口来实现。

#### LineNumberInputStream

该类被标记为@Deprecated，是不鼓励使用的。附件功能就是能够统计当前InputStream已读取内容中的行数。

#### ObjectInputStream

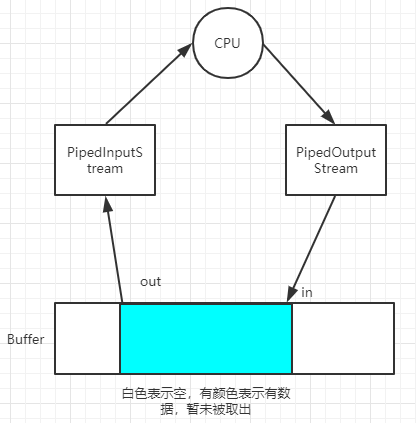
该类能够将已经序列后的内置类型和自定义类型对象读入内存。一般在将对象序列化写入文件使用。

#### PipedInputStream

这个类是通过管道读入流，必须和PipedOutputStream成对出现。通过Read操作读取流时，如果流是empty会被blocked。block的方式是自旋+wait（释放锁），while循环直到有结果，或者pipe被关闭；

PipedOutputStream会引用一个PipedInputStream对象，调用PipedOutputStream的write方法其实就是调用PipeInputStream对象的receive方法，将数据写到PipedInputStream的Buffer中。

Buffer是一个循环数组。有两个下标in和out（此处的in和out是站在buffer的角度判断，InputStream和OutputStream是站在cpu角度判断），in表示PipedOutStream下一次写入Buffer的位置，out表示PipedInputStream下次读取Buffer的位置。in等于out表示Buffer已满。



总结：

PipedInputStream和PipedOutputStream的核心是共享循环Buffer。

#### PushbackInputStream

顾名思义pushback，能将从InputStream中读取出来的数据“退回”。为了实现这个“退回”功能，该类提供一个Buffer缓存“退回”的数据，通过unread操作“退回”，下一次执行read操作时，首先在Buffer中判断是否有之前“退回”的数据，如果有就从buffer中读取，如果数据不够，接着从InputStream中read。以此来达到一种“退回”功能。

#### SequenceInputStream

该类是个集合，能同时注册多个InputStream，在读取数据时按照注册的InputStream顺序逐个读取，读取完一个InputStream接着读取下一个。

#### StringBufferInputStream

该类被标记为@Deprecated，是不鼓励使用的。同时该类是个伪InputStream，真正读取的是从一个String中读取，而不是从InputStream对象。

### 输出字节流

输出字节流和输入字节流具有相对关系，其实现逻辑和输入字节流是类似。

#### BufferedOutputStream

该类将需要输出的流缓存到Buffer中，然后调用flush批量写入OutputStream。write方法是线程安全的。

#### ByteArrayOutputStream

ByteArrayOutputStream和BufferedOutputStream的区别是，ByteArrayOutputStream每次调用write时，如果buf数组长度不够，需要按需扩展，而BufferedOutputStream在buf数组长度不够时会调用flush操作，将buf中的字节流写入OutputStream，然后从头缓存。

该类的write方法是线程安全的。

#### DataOutputStream

该类将Java 内置类型数据写入OutputStream流。所有write方法都是非线程安全的。

##### FileOutputStream

写入文件的OutputStream。非线程安全。

#### FilterOutputStream

OutputStream的包装类，可以包装OutputStream类的所有子类。非线程安全。

#### ObjectOutputStream

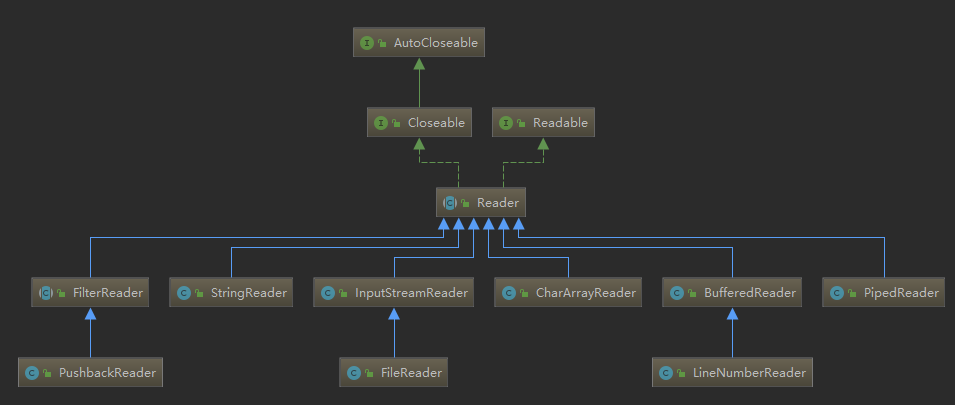
Java对象序列化输出流。非线程安全。

#### PipedOutputStream

参考PipedInputStream。线程安全的。会阻塞。

### 输入字符流

输入字符流成员类之间的关系如下。



#### BufferedReader

#### CharArrayReader

把一个char数组封装成一个Reader。

#### FileReader

#### FilterReader

#### InputStreamReader

#### LineNumberReader

#### PipedReader

#### PushbackReader

#### StringReader

### 输出字符流

# NIO

## 基础概念

* 阻塞、非阻塞：相对于客户端来说，如果调用io请求，客户端一直处于等待，则称为阻塞，相反，如果客户端调用完io请求后接着执行其他代码，然后周期性去轮训请求是否结束，则被称为非阻塞；
* 同步、异步：相对于服务端来说

## 概述

Java NIO包含三个模块，分别是：

* Channel
* Buffer
* Selector

Java NIO和Java IO存在的最大不同是NIO操作的是Buffer，而Java IO操作的是流。同时Java NIO是将java内置类型（除布尔类型外）写入对应类型的数组。

Java NIO包下的类具有很多共性，并且规律性也很强。和Java IO类似，NIO包中的类都是分层级的。

## Buffer

Buffer主要是将Java内置类型（bool类型除外）按照字节的形式存储，按照存储方式可以将Buffer分为三类，一类是大端存储，一类是小端存储，一类是read-only存储。

### 共性

* 所有的Buffer子类都会转换为ByteBuffer存放在Byte数组中；
* 按照大端（Big-endian）、小端（Little-endian），以及Read-only的不同形式，每个Buffer会有4种不同的实现方式。以Int为例，分别是ByteBufferAsIntBufferB，ByteBufferAsIntBufferL，ByteBufferAsIntBufferRB，ByteBufferAsIntBufferRL

注意：大端通俗的理解就是把数值看做字符串从内存低位开始存放。

### 第一层

第一层包含两个，一个是Buffer类，一个是Comparable接口。Buffer是个抽象类，NIO包下的所有类都继承自该类，同时每个类都实现了Comparable接口，因此其子类都是可以比较的，具有偏序关系。

### 第二层

第二层是内置类型对应的Buffer，所有类都是抽象类。分别是：

* LongBuffer
* ShortBuffer
* IntBuffer
* DoubleBuffer
* FloatBuffer
* ByteBuffer
* CharBuffer

# 参考

英文文档（包含各种demo）：<http://tutorials.jenkov.com/java-io/index.html>

对应中文网：<http://ifeve.com/overview/>

nio博客<https://www.ibm.com/developerworks/cn/education/java/j-nio/j-nio.html>