

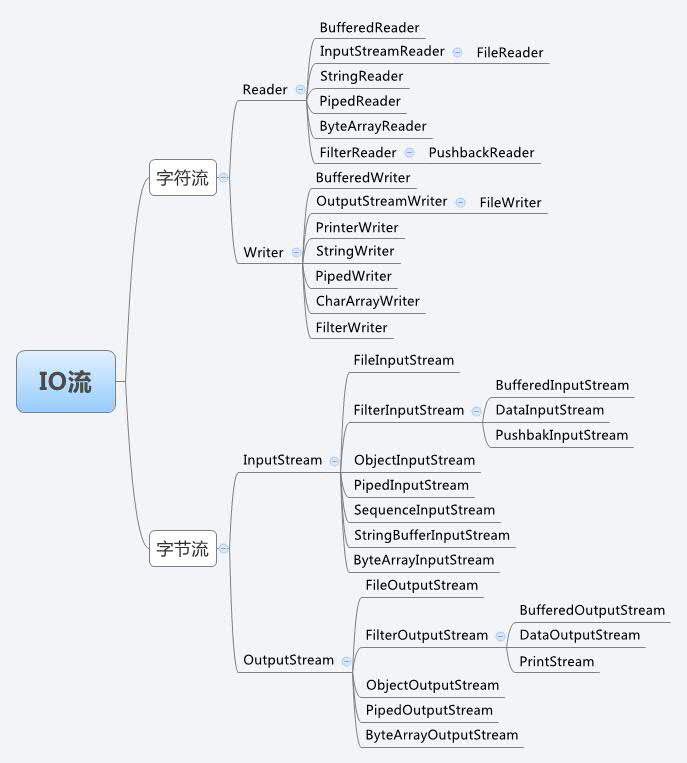
- 继承inputStream/outputStream的，为真正意义上的流

- 继承reader/writer的，为流的处理方式，内部创建了流，解决编码问题

- 继承buffer的，为流的处理方式，使用外部传入的流，解决IO频繁问题

把一个对象序列化有一个前提是：这个对象的类，必须实现了Serializable接口

[write](mk:@MSITStore:D:\Java%20Work\jdk%20api%201.8_google.CHM::/java/io/FileOutputStream.html#write-byte:A-int-int-)(byte[] b, int off, int len) len+off-1<b.length



流的分类：

1. 数据单位不同：字节流（8bit），字符流（16bit）
2. 流向不同：输入流，输出流
3. 角色不同：节点流，处理流
4. 抽象基类：
   1. 字节流：InputStream，OutputStream
   2. 字符流：Reader，Writer



InputStream & Reader：

1. InputStream：
   1. Int read（）从输入流中读取数据的下一个字节。返回 0 到 255 范围内的 int 字节值。如果因为已经到达流末尾而没有可用的字节，则返回值 -1。
   2. 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。如果因为已经到达流末尾而没有可用的字节，则返回值 -1。否则以整数形式返回实际读取的字节数。
   3. Int read（byte[] b，int off，int len）将输入流中最多 len 个数据字节读入 byte 数组。尝试读取 len 个字节，但读取的字节也可能小于该值。以整数形式返回实际读取的字节数。如果因为流位于文件末尾而没有可用的字节，则返回值 -1。
2. Reader：
   1. Int reader（）读取单个字符。作为整数读取的字符，范围在 0 到 65535 之间 (0x00-0xffff)（2个字节的Unicode码），如果已到达流的末尾，则返回 -1
   2. Int reader（char[] c）将字符读入数组。如果已到达流的末尾，则返回 -1。否则返回本次读取的字符数。
   3. Int reader（char[] c，int off，int len）将字符读入数组的某一部分。存到数组cbuf中，从off处开始存储，最多读len个字符。如果已到达流的末尾，则返回 -1。否则返回本次读取的字符数。
3. IO流应该显示关闭
4. Int off为偏移量，指的是，存放在缓冲数组中的偏移，例如偏移量为1，每次读取2，则第一个存放的位置在该数组下表为1的地方然后依次存放，len<缓冲数组的长度-偏移量

OutputStream&Writer

1. void write(int b/int c);
2. void write(byte[] b/char[] cbuf);
3. void write(byte[] b/char[] buff, int off, int len);
4. void flush();
5. 因为字符流直接以字符作为操作单位，所以 Writer 可以用字符串来替换字符数组，即以 String 对象作为参数
   1. void write(String str);
   2. void write(String str, int off, int len);
6. OutputStream：
   1. void write(int b)将指定的字节写入此输出流。write 的常规协定是：向输出流写入一个字节。要写入的字节是参数 b 的八个低位。b 的 24 个高位将被忽略。 即写入0~255范围的。
   2. void write(byte[] b)将 b.length 个字节从指定的 byte 数组写入此输出流。write(b) 的常规协定是：应该与调用 write(b, 0, b.length) 的效果完全相同。
   3. void write(byte[] b,int off,int len)将指定 byte 数组中从偏移量 off 开始的 len 个字节写入此输出流
7. Writer：
   1. void write(int c)写入单个字符。要写入的字符包含在给定整数值的 16 个低位中，16 高位被忽略。 即写入0 到 65535 之间的Unicode码。
   2. void write(char[] cbuf)写入字符数组。
   3. void write(char[] cbuf,int off,int len)写入字符数组的某一部分。从off开始，写入len个字符
   4. void write(String str)写入字符串。
   5. void write(String str,int off,int len)写入字符串的某一部分。
8. 在写入一个文件时，如果使用构造器FileOutputStream(file)，则目录下有同名文件将被覆盖。但是在构造器后加入参数true，则不会被覆盖，会在末尾追加内容
9. 在读取文件时，必须保证该文件已存在，否则报异常。

缓冲流：会在内部默认使用一个8192字节（8kb）的缓冲区

1. BufferedInputStream 和 BufferedOutputStream
2. BufferedReader 和 BufferedWriter
3. 当读取数据时，数据按块读入缓冲区，其后的读操作read方法则直接访问缓冲区，若缓冲区无数据则向物理文件读取进入缓冲区

转换流：转换流提供了在字节流和字符流之间的转换

1. InputStreamReader：将InputStream转换为Reader
2. OutputStreamWriter：将Writer转换为OutputStream

InputStreamReader：实现将字节的输入流按指定 字符集 转换为字符的输入流。

1. 需要和InputStream“套接”
   1. 构造器
      1. public InputStreamReader(InputStream in)
      2. public InputSreamReader(InputStream in,String charsetName)
      3. 例如：Reader isr = new InputStreamReader(System.in,”gbk”);

OutputStreamWriter：实现将字符的输出流按指定字符集转换为字节的输出流。

1. 需要和OutputStream“套接”。
   1. 构造器
      1. public OutputStreamWriter(OutputStream out)
      2. public OutputSreamWriter(OutputStream out,String charsetName)

标准输入输出流：

1. System.in和System.out分别代表了系统标准的输入和输出设备
2. System.in的类型是InputStream
3. System.out的类型是PrintStream，其是OutputStream的子类FilterOutputStream 的子类
4. 重定向：通过System类的setIn，setOut方法对默认设备进行改变。
   1. public static void setIn(InputStream in)
   2. public static void setOut(PrintStream out)

打印流：

1. 实现将基本数据类型的数据格式转化为字符串输出
2. 打印流：PrintStream和PrintWriter
   1. 提供了一系列重载的print()和println()方法，用于多种数据类型的输出
   2. PrintStream和PrintWriter的输出不会抛出IOException异常
   3. PrintStream和PrintWriter有自动flush功能
   4. PrintStream 打印的所有字符都使用平台的默认字符编码转换为字节。
   5. 在需要写入字符而不是写入字节的情况下，应该使用 PrintWriter 类。
   6. System.out返回的是PrintStream的实例

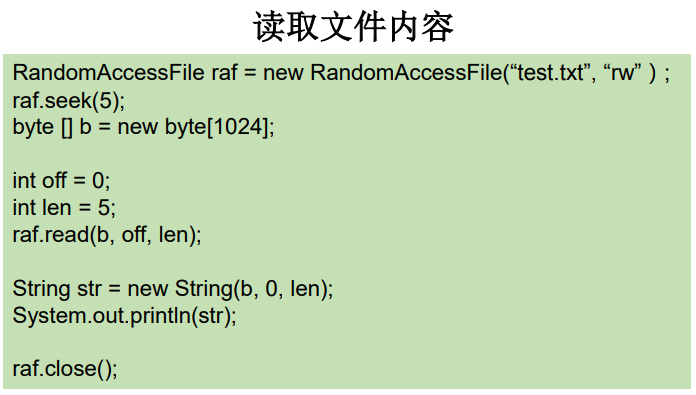
数据流：

1. 为了方便地操作Java语言的基本数据类型和String的数据，可以使用数据流。
2. DataInputStream 和 DataOutputStream，分别“套接”在 InputStream 和 OutputStream 子类的流上
3. DataInputStream中的方法：
   1. boolean readBoolean() byte readByte()
   2. char readChar() float readFloat()
   3. double readDouble() short readShort()
   4. long readLong() int readInt()
   5. String readUTF() void readFully(byte[] b)
4. DataOutputStream中的方法
   1. 将上述的方法的read改为相应的write即可。

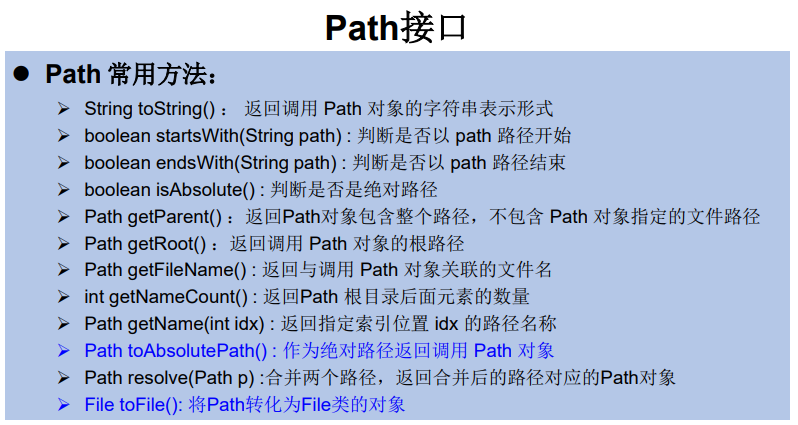
对象流：

1. 用于存储和读取基本数据类型数据或对象的处理流。它的强大之处就是可以把Java中的对象写入到数据源中，也能把对象从数据源中还原回来。
2. ObjectInputStream和OjbectOutputSteam
3. 不能序列化static和transient修饰的成员变量

随机存取文件流：

1. RandomAccessFile 类支持 “随机访问” 的方式，程序可以直接跳到文件的任意地方来读、写文件
   1. 支持只访问文件的部分内容
   2. 可以向已存在的文件后追加内容
2. RandomAccessFile 对象包含一个记录指针，用以标示当前读写处的位置。
3. RandomAccessFile 类对象可以自由移动记录指针：
   1. long getFilePointer()：获取文件记录指针的当前位置
   2. void seek(long pos)：将文件记录指针定位到 pos 位置
4. 构造器
   1. public RandomAccessFile(File file, String mode)
   2. public RandomAccessFile(String name, String mode)
5. 创建 RandomAccessFile 类实例需要指定一个 mode 参数，该参数指定 RandomAccessFile 的访问模式：
   1. r: 以只读方式打开，文件需要已存在
   2. rw：打开以便读取和写入，若文件不存在则创建
   3. rwd:打开以便读取和写入；同步文件内容的更新
   4. rws:打开以便读取和写入；同步文件内容和元数据的更新
6. 
7. 

NIO.2：

1. Path、Paths和Files
2. Paths 类提供的静态 get() 方法用来获取 Path 对象：
   1. static Path get(String first, String … more) : 用于将多个字符串串连成路径
   2. static Path get(URI uri): 返回指定uri对应的Path路径
3. 
4. 
5. 