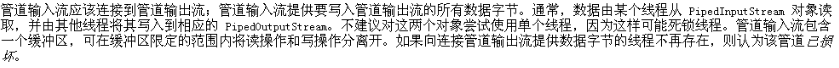
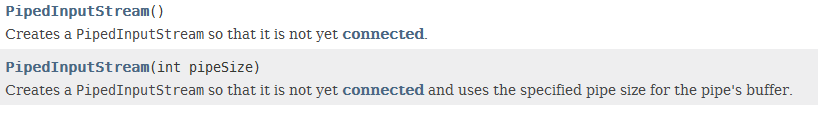
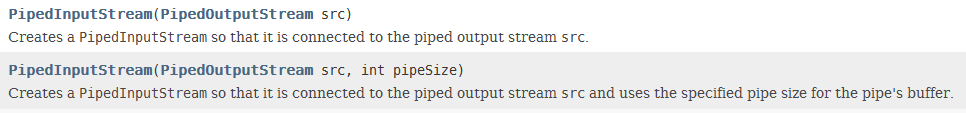
管道流PipedInputStream和PipedOutputStream

1. **PipedInputStream和PipedOutputStream**适用于多线程，单线程时容易发生死锁。read方法是一个阻塞式方法。

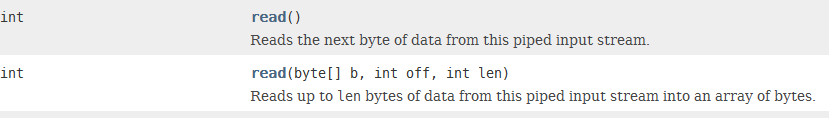


1. **PipedInputStream管道输入流：**
2. **构造方法：**





1. **成员方法：**
2. read方法：



1. connect方法：



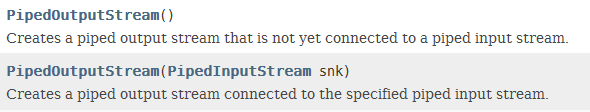
1. close方法：receive方法



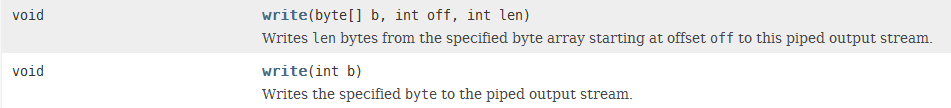
1. available方法：



1. **PipedOutputStream管道输出流：**
2. **构造方法：2个**



1. **成员方法:**
2. **write方法：**



1. flush、close方法；
2. **connect方法**：



1. **既然是管道流，那么输入流和输出流就可以连接在一起，PipedInputStream和PipedOutputStream连接的方法：**
2. **在构造方法中连接：**

**PipedInputStream的构造方法：**





**PipedOutputStream的构造方法：**



1. **利用PipedInputStream或者PipedOutputStream的connect方法：**





1. 练习：

主类：

PipedInputStream pin = **new** PipedInputStream();

PipedOutputStream pout = **new** PipedOutputStream();

**pin.connect(pout);//把管道流连接起来**

**new** Thread((**new** Input(pin))).start();

**new** Thread((**new** Output(pout))).start();

线程1：

public class Input implements Runnable{

private PipedInputStream pin;

public Input(PipedInputStream pin) {

this.pin = pin;

}

public void run() {

try {

byte[] buf = new byte[1024];

int len = pin.read(buf);

String str = new String(buf,0,len);

System.out.println(str);

} catch (IOException e) {

e.printStackTrace();

}finally {

try {

pin.close();

} catch (IOException e) {

e.printStackTrace();

}}}}

线程2：

**public class Output implements Runnable{**

**private PipedOutputStream pout;**

**public Output(PipedOutputStream pout) {**

**this.pout = pout;**

**}**

**public void run() {**

**try {**

**pout.write("我是管道流".getBytes());**

**} catch (IOException e) {**

**}finally {**

**try {**

**pout.close();**

**} catch (IOException e) {**

**}}}}**