**Java IO流的基本知识**

* 对一下几个类的理解：
  + 类File

|  |  |
| --- | --- |
|  | |
| [**File**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#File(java.io.File, java.lang.String))([File](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html) parent, [String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) child)  根据 parent 抽象路径名和 child 路径名字符串创建一个新 File 实例。 |  |
| [**File**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#File(java.lang.String))([String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) pathname)  通过将给定路径名字符串转换为抽象路径名来创建一个新 File 实例。 |  |
| [**File**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#File(java.lang.String, java.lang.String))([String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) parent, [String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) child)  根据 parent 路径名字符串和 child 路径名字符串创建一个新 File 实例。 |  |
| [**File**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html#File(java.net.URI))([URI](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/net/URI.html) uri)  通过将给定的 file: URI 转换为一个抽象路径名来创建一个新的 File 实例。 |  |

* 类FileInputStream:

|  |  |
| --- | --- |
| **构造方法摘要** | |
| [**FileInputStream**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#FileInputStream(java.io.File))([File](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html) 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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#FileInputStream(java.io.FileDescriptor))([FileDescriptor](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileDescriptor.html) fdObj)  通过使用文件描述符 fdObj 创建一个 FileInputStream，该文件描述符表示到文件系统中某个实际文件的现有连接。 |  |
| [**FileInputStream**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#FileInputStream(java.lang.String))([String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) 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。 |  |

|  |  |
| --- | --- |
| **方法摘要** | |
| int | [**available**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#available())()  返回下一次对此输入流调用的方法可以不受阻塞地从此输入流读取（或跳过）的估计剩余字节数。 |
| void | [**close**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#close())()  关闭此文件输入流并释放与此流有关的所有系统资源。 |
| protected void | [**finalize**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#finalize())()  确保在不再引用文件输入流时调用其 close 方法。 |
| [FileChannel](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/nio/channels/FileChannel.html) | [**getChannel**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#getChannel())()  返回与此文件输入流有关的唯一 [FileChannel](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/nio/channels/FileChannel.html) 对象。 |
| [FileDescriptor](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileDescriptor.html) | [**getFD**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#getFD())()  返回表示到文件系统中实际文件的连接的 FileDescriptor 对象，该文件系统正被此 FileInputStream 使用。 |
| int | [**read**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#read())()  从此输入流中读取一个数据字节。 |
| int | [**read**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#read(byte[]))(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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#read(byte[], int, int))(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。 |
| long | [**skip**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileInputStream.html#skip(long))(long n)  从输入流中跳过并丢弃 n 个字节的数据。 |

* 类 FireOutputStream :

|  |  |
| --- | --- |
| **构造方法摘要** | |
| [**FileOutputStream**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#FileOutputStream(java.io.File))([File](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html) 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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#FileOutputStream(java.io.File, boolean))([File](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/File.html) 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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#FileOutputStream(java.io.FileDescriptor))([FileDescriptor](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileDescriptor.html) fdObj)  创建一个向指定文件描述符处写入数据的输出文件流，该文件描述符表示一个到文件系统中的某个实际文件的现有连接。 |  |
| [**FileOutputStream**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#FileOutputStream(java.lang.String))([String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) 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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#FileOutputStream(java.lang.String, boolean))([String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) 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。 |  |

|  |  |
| --- | --- |
| **方法摘要** | |
| void | [**close**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#close())()  关闭此文件输出流并释放与此流有关的所有系统资源。 |
| protected void | [**finalize**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#finalize())()  清理到文件的连接，并确保在不再引用此文件输出流时调用此流的 close 方法。 |
| [FileChannel](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/nio/channels/FileChannel.html) | [**getChannel**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#getChannel())()  返回与此文件输出流有关的唯一 [FileChannel](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/nio/channels/FileChannel.html) 对象。 |
| [FileDescriptor](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileDescriptor.html) | [**getFD**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#getFD())()  返回与此流有关的文件描述符。 |
| void | [**write**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#write(byte[]))(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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#write(byte[], int, int))(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**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/FileOutputStream.html#write(int))(int b)  将指定字节写入此文件输出流。 |

* 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。通常采用此操作时常常将BufferedReader包装其所有read()操作以提高开销很高的Reader(如FileReader ,Inputstream Reader)。

具体实现：

BufferdReader br = new BufferdReader (new FileReader(“Sharon.text”)); “Sharon.text”为文件名

|  |  |
| --- | --- |
| **构造方法摘要** | |
| [**BufferedReader**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#BufferedReader(java.io.Reader))([Reader](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/Reader.html) in)  创建一个使用默认大小输入缓冲区的缓冲字符输入流。 |  |
| [**BufferedReader**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#BufferedReader(java.io.Reader, int))([Reader](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/Reader.html) in, int sz)  创建一个使用指定大小输入缓冲区的缓冲字符输入流。 |  |

|  |  |
| --- | --- |
| **方法摘要** | |
| void | [**close**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#close())()  关闭该流并释放与之关联的所有资源。 |
| void | [**mark**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#mark(int))(int readAheadLimit)  标记流中的当前位置。 |
| boolean | [**markSupported**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#markSupported())()  判断此流是否支持 mark() 操作（它一定支持）。 |
| int | [**read**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#read())()  读取单个字符。 |
| int | [**read**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#read(char[], int, int))(char[] cbuf, int off, int len)  将字符读入数组的某一部分。 |
| [String](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/lang/String.html) | [**readLine**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#readLine())()  读取一个文本行。 |
| boolean | [**ready**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#ready())()  判断此流是否已准备好被读取。 |
| void | [**reset**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#reset())()  将流重置到最新的标记。 |
| long | [**skip**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/BufferedReader.html#skip(long))(long n)  跳过字符。 |

* BufferedWriter:

将文本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。

可以指定缓冲区的大小，或者接受默认的大小。在大多数情况下，默认值就足够大了。该类提供了 newLine() 方法，它使用平台自己的行分隔符概念，此概念由系统属性 line.separator 定义。并非所有平台都使用新行符 ('\n') 来终止各行。因此调用此方法来终止每个输出行要优于直接写入新行符。

通常 Writer 将其输出立即发送到底层字符或字节流。除非要求提示输出，否则建议用 BufferedWriter 包装所有其 write() 操作可能开销很高的 Writer（如 FileWriters 和 OutputStreamWriters）。例如，

PrintWriter out

= new PrintWriter(new BufferedWriter(new FileWriter("foo.out")));

* DataInputStream:

数据输入流允许应用程序对于底层输入流中基本数据的读取。

* DataInputStream:

数据输出流允许应用程序以适当方式将基本 Java 数据类型写入输出流中。然后，应用程序可以使用数据输入流将数据读入。

|  |  |
| --- | --- |
| **构造方法摘要** | |
| [**DataOutputStream**](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/DataOutputStream.html#DataOutputStream(java.io.OutputStream))([OutputStream](mk:@MSITStore:E:\java之路\各类查询文档\java查询文档\JDK_API_1_6_zh_CN.CHM::/java/io/OutputStream.html) out)  创建一个新的数据输出流，将数据写入指定基础输出流。 |  |

* 关于流：
  + 理解文件的存储文件的读取都是以流的形式实现的.
  + 流是一连串的字符，是以先进先出发送信息的通道。操作的是字节数据，写进流的时候定义的是字节的类型。
  + 流的分类
    - 字节流 （8位） inputStream OutputStream
    - 字符流 （16位）Reader Writer
* 用FileInputStream读取硬盘上文件的操作
  + 首先引入相关的类
  + 构建一个输入流的对象

InstreamStream is = new FileInputStream (“text.txt”);

* + 应用文件输入输出的方法读取文本文件

Is.available();//可读取的字节数

可以用循环来读取

Is.read(); //读取文件的数据

Is.close();//关闭文件输出流对象

* + 案例应用

|  |
| --- |
| **package** com\_workit\_io;  **import** java.io.FileInputStream;  **import** java.io.FileNotFoundException;  **import** java.io.IOException;  **import** java.io.InputStream;  **public** **class** demo01 {  /\*\*  \* **@param** args  \*/  **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **int** size ;  **try** {  InputStream is = **new** FileInputStream("Sharon.txt"); 创建一个FileInputStream 流对象  System.*out*.println("文本可读取的字节数为："+ (size = is.available()));求的文件大小的语句  **char**[] temp = **new** **char**[1024]; //一个一个字节的读取  **for** (**int** count = 1;count <= size;count++)  {  temp[count] = (**char**)(is.read());  System.*out*.print(temp[count]);  }  System.*out*.println("");  Is.close();  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  }  } |

* 用FileOutStream写文本文件的操作
  + 首先构建输出流对象

outputStream os = new FileoutputStream(“text.txt”);

* + 用文件输出流的方法写文本文件

String str = “欢迎您来到文理学院！！！” //定义您要写入的文件字符串

Byte[] buff = str.getBytes();//流操作的是字符串的。将定义的空间为字符的数组

Os.write(buff,0,buff.length);//用写的方法将文件从第一个字节写到最后有一个写进文件中

* + 最后关闭输出流

Os.close();

* + 案例应用

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **try** {  OutputStream os = **new** FileOutputStream("huahua.txt"); 创建一个输入流的对象将文件保存在huahua.txt文件中  String str = "欢迎您来到湖南文理学院计算机学院！";  **byte**[] buff = str.getBytes();  **try** {  os.write(buff, 0, buff.length);  System.*out*.println("文件保存成功！！！");  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |

* + 了解了文件读和写的操作，将读取出的文件在写进另外的文件中即实现的文件的备份

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **try** {  InputStream is = **new** FileInputStream("Sharon.txt");  OutputStream os = **new** FileOutputStream("Sharonbackup.txt");  //同时引进出如何输出流的对象  **byte**[] buff\_source = **new** **byte**[1024];  **try** {  is.read(buff\_source, 0, buff\_source.length);  //读取文件从第一个直到最后一个字节流  os.write(buff\_source, 0, buff\_source.length);  //写进备份的文件中从第一个到最后一个字节  Is.close(); //勿忘流的关闭  os.close();  System.*out*.println("文件备份成功！");  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |

* 用BufferedReader 读文本文件(缓冲一次读取一行的)
  + 构造相关的对象

FileReader fr = new FileReader(“Sharon.txt”);

BufferedReader br = new BufferedReader(fr);//读取文件一行一行的读取，相对于一个一个字节的读取提高的效率

* + 用BufferedReader的方法一行一行的读取文件的数据

Br.readLine();//读取一行数据返回一个字符串

* + 最后关闭相关的流对象

Br.close();

Fr.close();

* + 案例应用

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **try** {  FileReader fr = **new** FileReader("Sharon.txt");//创建fileReader对象  BufferedReader br = **new** BufferedReader(fr);//将文本中的流关联起来  **try** {  String linestr = br.readLine(); //每读取到一行就用lineStr表述  **while**(linestr != **null**) //只要读取到下一行还有数据就继续读下去  {  System.*out*.println(linestr);  linestr = br.readLine();  }  br.close();  fr.close(); //关闭相关的流对象时最后打开的要先关  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }    } |

* + 类似于BufferedReader用BufferedWriter写文本文件
  + 构造相关的对象

FileWriter fw = new FileWriter(“Sharon.txt”);

BufferedWriter bw = new BufferedWriter (fw);//读取文件一行一行的读取，相对于一个一个字节的读取提高的效率

* + 用BufferedReader的方法写文件

Bw.writer();

* + 最后相关的流对象的清空和关闭

Bw.flush();

Fw.close();

* + 案例应用（结合bufferedReaderh,bufferedWriter实现文件的备份）

|  |
| --- |
| **public** **static** **void** main(String[] args) {  // **TODO** Auto-generated method stub  **try** {  FileReader fr = **new** FileReader("Sharon.txt");  BufferedReader br = **new** BufferedReader(fr);    **try** {  FileWriter fw = **new** FileWriter("xiaolianzai.txt");  BufferedWriter bw = **new** BufferedWriter(fw);  String linestr = br.readLine();//一行一行的读取  **while**( linestr != **null**)  {  bw.write(linestr);  bw.newLine(); //另起一行否则原来文件中的文件将以一行的形式输出  linestr = br.readLine();  }  bw.close();  fw.close();  br.close();  fr.close(); //最后打开先关闭  System.*out*.println("文件备份成功！！！");  } **catch** (IOException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } **catch** (FileNotFoundException e) {  // **TODO** Auto-generated catch block  e.printStackTrace();  }  } |

* 总结
  + 熟练了解File的操作，流的操作总是成对出现的
  + FileinputStream 和 FileoutPutStream 一个一个字节的读取效率较低
  + Bufferedreader 和 BufferedWriter 一行一行的读取

相对于InputStream而言效率较高，同时这两者都可以实现文件的读取操作，效率的高低是两者的主要区别

* + DataInputStream 和DataoutStream 实现二进制文件的读写操作，代码雷同于上面两种对流的操作