**Class ByteArrayInputStream**

* [java.lang.Object](https://docs.oracle.com/javase/8/docs/api/java/lang/Object.html)
  + [java.io.InputStream](https://docs.oracle.com/javase/8/docs/api/java/io/InputStream.html)
    - java.io.ByteArrayInputStream
* **All Implemented Interfaces:**

[Closeable](https://docs.oracle.com/javase/8/docs/api/java/io/Closeable.html), [AutoCloseable](https://docs.oracle.com/javase/8/docs/api/java/lang/AutoCloseable.html)

|  |  |
| --- | --- |
| void | [**mark**](https://docs.oracle.com/javase/8/docs/api/java/io/ByteArrayInputStream.html#mark-int-)(int readAheadLimit) |
| void | **reset()** |

两个方法一般配合使用.mark方法的作用是在当前文职做一个标记,当使用reset方法时,会从mark方法标记的地方开始读取读取流中的内容.

关于mark方法中的参数readAheadLimit,跟mark方法本身没关系,参数值的设置是用来在其他方法中使用的.具体参考最后部分的解释和实例

|  |
| --- |
| 实例 |
| **public** **class** *Test* {  **public** **static** **void** **main**(*String*[] args) **throws** *IOException* {  **byte**[] b = **new** **byte**[] { 1, 2, 3 ,4 };  *ByteArrayInputStream* bis = **new** **ByteArrayInputStream**(b);  *System*.***out***.**println**(bis.**read**()); //1  *System*.***out***.**println**(bis.**read**()); //2    bis.**mark**(90000);//reset方法后，定位在3上，以后对流中数据的操作会从3开始  *System*.***out***.**println**(bis.**read**()); //3  *System*.***out***.**println**(bis.**read**()); //4    bis.**reset**(); //从mark定位的2开始输出  *System*.***out***.**println**(bis.**read**()); //3  *System*.***out***.**println**(bis.**read**()); //4  }  } |

注意:mark(9000)并不是在流中index为9000的地方做标记。这个参数跟mark方法没关系。

|  |  |
| --- | --- |
| int | [**available**](https://docs.oracle.com/javase/8/docs/api/java/io/ByteArrayInputStream.html#available--)()  返回字节流中还可以被读取的字节的个数。 |

|  |
| --- |
| 实例 |
| **public** **class** *Test* {  **public** **static** **void** **main**(*String*[] args) **throws** *IOException* {  **byte**[] b = **new** **byte**[] { 1, 2,3 };  *ByteArrayInputStream* bis = **new** **ByteArrayInputStream**(b);  *System*.***out***.**println**(bis.**read**()); //1  *System*.***out***.**println**(bis.**available**());//返回1,字节流中还剩2个字节可以被读取  *System*.***out***.**println**(bis.**read**()); //2  *System*.***out***.**println**(bis.**read**()); //3  *System*.***out***.**println**(bis.**available**());//返回0,字节流中已经么有可以被读取的字节了  }  } |

|  |  |
| --- | --- |
| int | [**close**](https://docs.oracle.com/javase/8/docs/api/java/io/ByteArrayInputStream.html#available--)()  ByteArrayInputStream这个方法没有用，因为没有任何实现 |

|  |  |
| --- | --- |
| int | read ()  从流中读取字节。 |

|  |  |
| --- | --- |
| int | read(byte[] b, int off, int len)  从流的当前位置，读取出len个字节，放到新数组b中，  从b中哪个位置（索引值）开始放由参数off指定，  首先Off+len不能超过b的容量，否则回报异常。  然后，如果b的pos（当前位置）等于b的count，也就是说已经到了b的end file位置，返回-1  其他约束看api |

|  |
| --- |
| 实例 |
| **public** **class** *Test* {  **public** **static** **void** **main**(*String*[] args) **throws** *IOException* {  **byte**[] b = **new** **byte**[] { 1, 2,3 };  *ByteArrayInputStream* bis = **new** **ByteArrayInputStream**(b);  bis.**mark**(1);//reset后，可以继续从流的开头开始操作流  bis.**skip**(3);  **byte**[] c = **new** **byte**[5];//初始化后c为[0,0,0,0,0]    //返回值为-1，因为前面skip（3）已经到了流的结尾了。数组c中没有被添加任何东西，因为没有从b中读取到东西。  *System*.***out***.**println**(bis.**read**(c,2, 3)); //-1    bis.**reset**();  *System*.***out***.**println**(bis.**read**(c,2, 3)); //[0,0,1,2,3]    bis.**reset**();  *System*.***out***.**println**(bis.**read**(c,2, 4));//异常。off+len(0+4)大于b的长度了  }  } |

|  |  |
| --- | --- |
| Long | skip (long n)  从流中当前位置跳过n个byte。 |

|  |
| --- |
| 实例 |
| **public** **class** *Test* {  **public** **static** **void** **main**(*String*[] args) **throws** *IOException* {  **byte**[] b = **new** **byte**[] { 1, 2,3 };  *ByteArrayInputStream* bis = **new** **ByteArrayInputStream**(b);  *System*.***out***.**println**(bis.**skip**(2));  *System*.***out***.**println**(bis.**read**()); //3  }  } |

Tip:关于mark(int readlimit)的作用

 根据JAVA官方文档的描述，mark(int readlimit)方法表示，标记当前位置，并保证在mark以后最多可以读取readlimit字节数据，mark标记仍有效。如果在mark后读取 超过readlimit字节数据，mark标记就会失效，调用reset()方法会有异常。

**但实际的运行情况却和JAVA文档中的描述并不完全相符。** 有时候在BufferedInputStream类中调用mark(int readlimit)方法后，即使读取超过readlimit字节的数据，mark标记仍有效，仍然能正确调用reset方法重置。

 事实上，mark在JAVA中的实现是和缓冲区相关的。只要缓冲区够大，mark后读取的数据没有超出缓冲区的大小，mark标记就不会失效。如果不够大，mark后又读取了大量的数据，导致缓冲区更新，原来标记的位置自然找不到了。

**因此，mark后读取多少字节才失效，并不完全由readlimit参数确定，也和BufferedInputStream类的缓冲区大小有关。** 如果BufferedInputStream类的缓冲区大小大于readlimit，在mark以后只有读取超过缓冲区大小的数据，mark标记才会失效。看下面的例子。

|  |
| --- |
| 实例 |
| 1. **package** packet1; 3. **import** java.io.BufferedInputStream; 4. **import** java.io.ByteArrayInputStream; 5. **import** java.io.IOException; 7. /\*\* 8. \* @author WuDian 9. \* 10. \*/ 11. **public** **class** MarkExample { 12. **public** **static** **void** main(String[] args) { 14. **try** { 15. // 初始化一个字节数组，内有5个字节的数据 16. **byte**[] bytes={1,2,3,4,5}; 17. // 用一个ByteArrayInputStream来读取这个字节数组 18. ByteArrayInputStream in=**new** ByteArrayInputStream(bytes); 19. // 将ByteArrayInputStream包含在一个BufferedInputStream，并初始化缓冲区大小为2。 20. BufferedInputStream bis=**new** BufferedInputStream(in,2); 21. // 读取字节1 22. System.out.print(bis.read()+","); 23. // 在字节2处做标记，同时设置readlimit参数为1 24. // 根据JAVA文档mark以后最多只能读取1个字节，否则mark标记失效，但实际运行结果不是这样 25. System.out.println("mark"); 26. bis.mark(1); 28. /\* 29. \* 连续读取两个字节，超过了readlimit的大小，mark标记仍有效 30. \*/ 31. // 连续读取两个字节 32. System.out.print(bis.read()+","); 33. System.out.print(bis.read()+","); 34. // 调用reset方法，未发生异常，说明mark标记仍有效。 35. // 因为，虽然readlimit参数为1，但是这个BufferedInputStream类的缓冲区大小为2， 36. // 所以允许读取2字节 37. System.out.println("reset"); 38. bis.reset(); 40. /\* 41. \* 连续读取3个字节，超过了缓冲区大小，mark标记失效。 42. \* 在这个例子中BufferedInputStream类的缓冲区大小大于readlimit, 43. \* mark标记由缓冲区大小决定 44. \*/ 45. // reset重置后连续读取3个字节，超过了BufferedInputStream类的缓冲区大小 46. System.out.print(bis.read()+","); 47. System.out.print(bis.read()+","); 48. System.out.print(bis.read()+","); 49. // 再次调用reset重置，抛出异常，说明mark后读取3个字节，mark标记失效 50. System.out.println("reset again"); 51. bis.reset(); 52. } **catch** (IOException e) { 53. // TODO Auto-generated catch block 54. e.printStackTrace(); 55. } 56. } 57. } |

 运行结果如下：

1. 1,mark
2. 2,3,reset
3. 2,3,4,reset again
4. java.io.IOException: Resetting to invalid mark
5. at java.io.BufferedInputStream.reset(BufferedInputStream.java:416)
6. at packet1.MarkExample.main(MarkExample.java:51)

 同样的，在调用mark(int readlimit)方法时，如果readlimit大于BufferedInputStream类缓冲区的大小，缓冲区会被扩大，那mark后最多就可以读readlimit字节。

**简言之，BufferedInputStream类调用mark(int readlimit)方法后读取多少字节标记才失效，是取readlimit和BufferedInputStream类的缓冲区大小两者中的最大值，而 并非完全由readlimit确定。这个在JAVA文档中是没有提到的。**