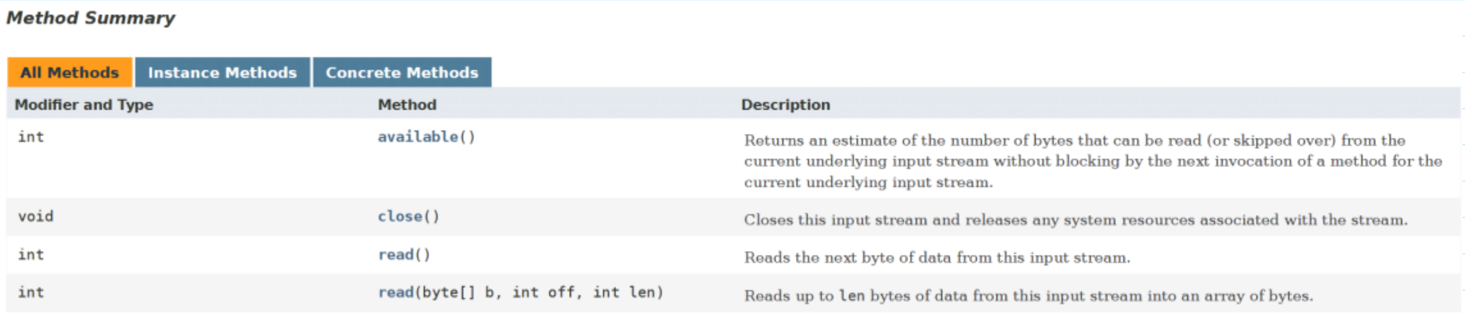
Java SequenceInputStream Class

01 September 2025 09:02

Java SequenceInputStreamClass is used to read data from multiple streams. It reads data sequentially(one by one).

```
public class SequenceInputStream extends InputStream
```

Constructors	Description
SequenceInputStream(InputStream s1, InputStream s2)	Creates a new input stream by reading the data of two input stream in order, first S1 and then S2.
SequenceInputStream(Enumeration e)	Creates a new input stream by reading the data of an enumeration whose type is InputStream.



Method Summary

Modifier and Type	Method	Description
int	available()	Returns an estimate of the number of bytes that can be read (or skipped over) from the current underlying input stream without blocking by the next invocation of a method for the current underlying input stream.
void	close()	Closes this input stream and releases any system resources associated with the stream.
int	read()	Reads the next byte of data from this input stream.
int	read(byte[] b, int off, int len)	Reads up to len bytes of data from this input stream into an array of bytes.

```
public int read(byte[] b,
            int off,
            int len)
     throws IOException
```

Reads up to len bytes of data from this input stream into an array of bytes. If len is not zero, the method blocks until at least 1 byte of input is available; otherwise, no bytes are read and 0 is returned.
The read method of SequenceInputStream tries to read the data from the current substream. If it fails to read any characters because the substream has reached the end of the stream, it calls the close method of the current substream and begins reading from the next substream.

Overrides:
 read in class InputStream
Parameters:
 b - the buffer into which the data is read.
 off - the start offset in array b at which the data is written.
 len - the maximum number of bytes read.
Returns:
 int the number of bytes read.
Throws:
 NullPointerException - If b is null.
 IndexOutOfBoundsException - If off is negative, len is negative, or len is greater than b.length - off
 IOException - if an I/O error occurs.

From <https://docs.oracle.com/javase/8/docs/api/java/io/SequenceInputStream.html#read-byte:A-int-int->

For example:

```
import java.util.*;
import java.io.FileInputStream;
import java.io.SequenceInputStream;
import java.io.IOException;

/**
* Write a description of class InputStreamExample here.
*
```

```
* @author (your name)
* @version (a version number or a date)
*/
public class InputStreamExample
{
    public static void main(String[] args)
    {
        String path= "./";
        String filename1;
        String filename2;
        Scanner sc=new Scanner(System.in);
        System.out.print("\f Enter first file name: ");
        filename1 = sc.nextLine();
        System.out.print("Enter second file name: ");
        filename2 = sc.nextLine();
        try(FileInputStream fis1 = new FileInputStream(path+filename1);
        FileInputStream fis2 = new FileInputStream(path+filename2);
        SequenceInputStream sis = new SequenceInputStream(fis1,fis2))
        {
            int x;
            while((x=sis.read())!=-1)
            {
                System.out.print((char)x);

            }
        }
        catch(IOException ioe)
        {
            System.err.print("\n I/O problem.");
        }


    }
}
```

Enumeration Example:

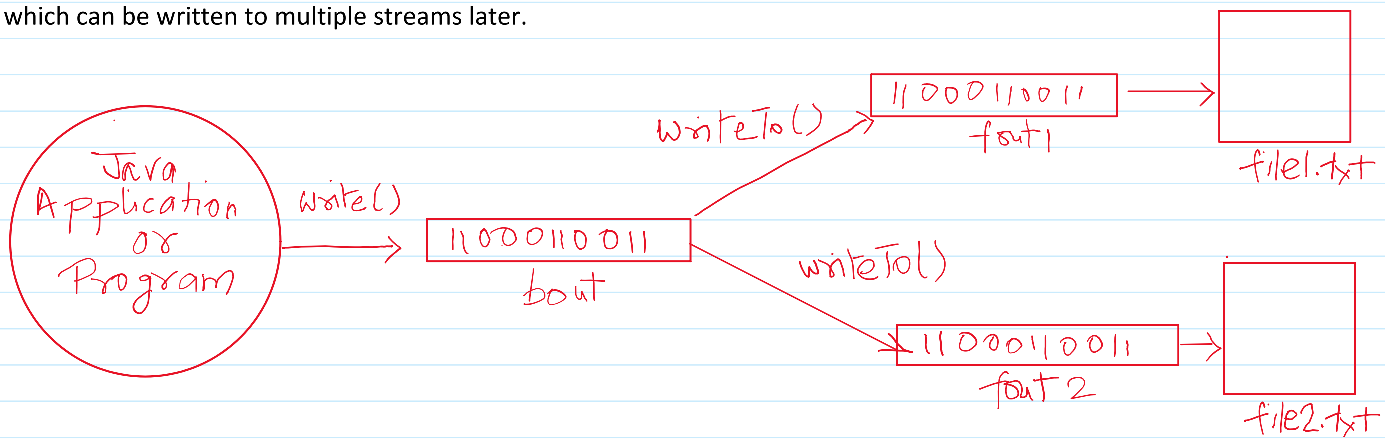
```
import java.util.*;
import java.io.*;

//hw wap to count the num of files present in the current path
/**
* Write a description of class InputStreamExample2 here.
*
* @author (your name)
* @version (a version number or a date)
*/
public class InputStreamExample2
{
    public static void main(String[] args)
    {
        Scanner sc=new Scanner(System.in);
        System.out.print("\f Enter number of files you want to read: ");
        int n=sc.nextInt();
        sc.nextLine();
        int i;
        String path="./"; //current project folder
    String[] filenames = new String[n];
    FileInputStream[] fis=new FileInputStream[n];
    try
    {
      Vector v = new Vector();

      for(i=0;i<n;i++)
      {
        System.out.print("Enter file name: ");
        filenames[i]=sc.nextLine();
        fis[i]=new FileInputStream(path+filenames[i]);
        v.add(fis[i]);
      }
      //creating enumeration object by calling the elements method.
      Enumeration e = v.elements();
      //passing the enumeration object in the constructor.
      SequenceInputStream sip = new SequenceInputStream(e);
      for(;(i=sip.read())!=-1;System.out.print((char)i));
      sip.close();
      for(i=0;i<n;fis[i++].close());
    }
    catch(IOException ieo)
    {
      System.err.print("\n IO problem.");
    }
  }
}
```

https://docs.oracle.com/en/java/javase/24/docs/api/java.base/java/util/Enumeration.html

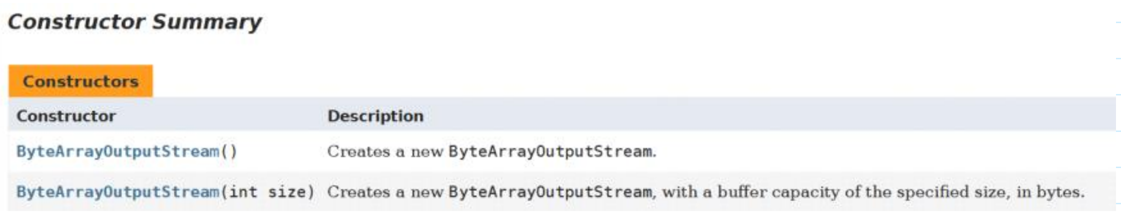
https://docs.oracle.com/en/java/javase/24/docs/api/java.base/java/util/Vector.html

ByteArrayOutputStream Class

Java ByteArrayOutputStream class is used to write common data into multiple files. In this stream, the data is written into a byte array which can be written to multiple streams later.



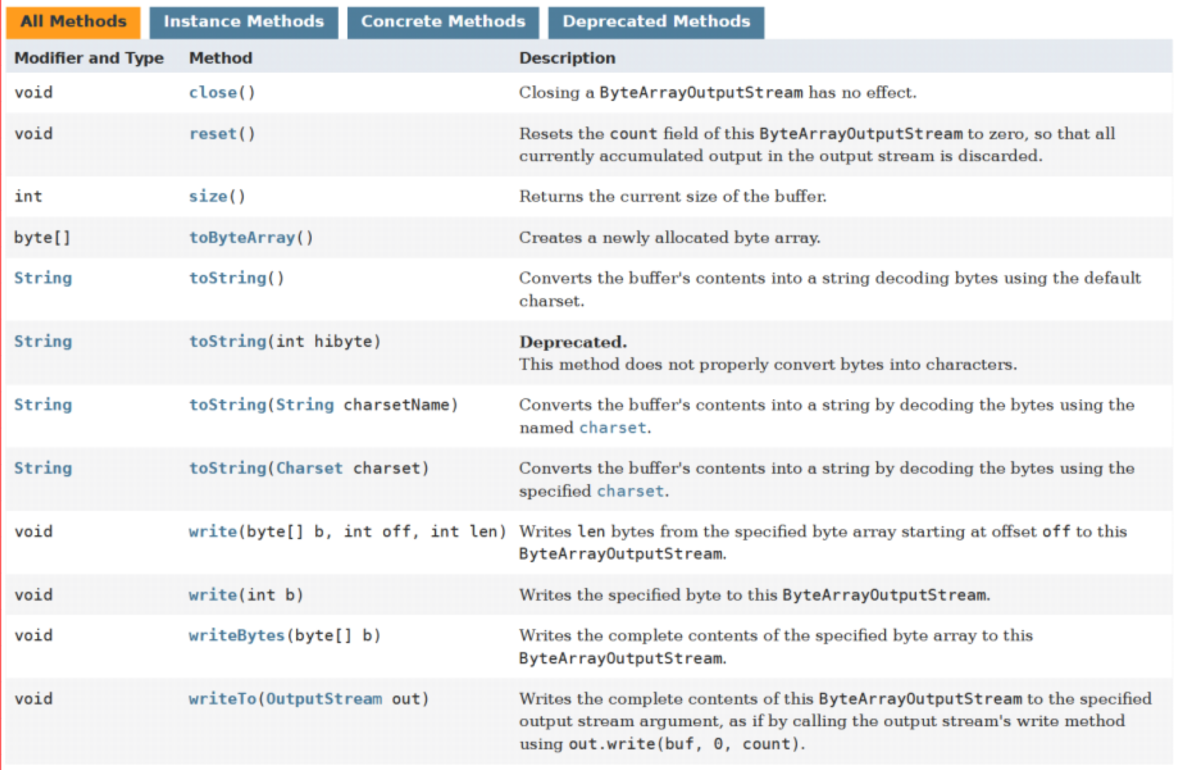
The ByteArrayOutputStream holds a copy of data and forwards it to multiple streams.
The buffer of ByteArrayOutputStream automatically grows according to data.



Constructor Summary

Constructors

Constructor	Description
`ByteArrayOutputStream()`	Creates a new `ByteArrayOutputStream`.
`ByteArrayOutputStream(int size)`	Creates a new `ByteArrayOutputStream`, with a buffer capacity of the specified size, in bytes.



All Methods | Instance Methods | Concrete Methods | Deprecated Methods

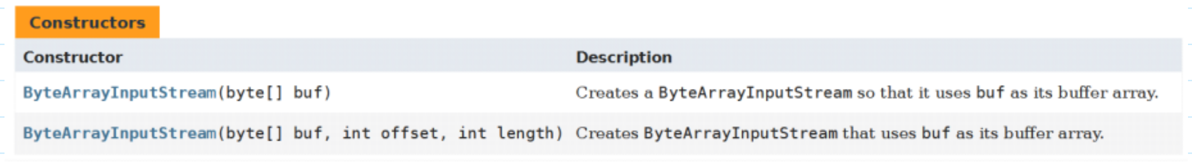
Modifier and Type	Method	Description
`void`	`close()`	Closing a `ByteArrayOutputStream` has no effect.
`void`	`reset()`	Resets the `count` field of this `ByteArrayOutputStream` to zero, so that all currently accumulated output in the output stream is discarded.
`int`	`size()`	Returns the current size of the buffer.
`byte[]`	`toByteArray()`	Creates a newly allocated byte array.
`String`	`toString()`	Converts the buffer's contents into a string decoding bytes using the default charset.
`String`	`toString(int hibyte)`	**Deprecated.** This method does not properly convert bytes into characters.
`String`	`toString(String charsetName)`	Converts the buffer's contents into a string by decoding the bytes using the named `charset`.
`String`	`toString(Charset charset)`	Converts the buffer's contents into a string by decoding the bytes using the specified `charset`.
`void`	`write(byte[] b, int off, int len)`	Writes `len` bytes from the specified byte array starting at offset `off` to this `ByteArrayOutputStream`.
`void`	`write(int b)`	Writes the specified byte to this `ByteArrayOutputStream`.
`void`	`writeBytes(byte[] b)`	Writes the complete contents of the specified byte array to this `ByteArrayOutputStream`.
`void`	`writeTo(OutputStream out)`	Writes the complete contents of this `ByteArrayOutputStream` to the specified output stream argument, as if by calling the output stream's write method using `out.write(buf, 0, count)`.

```
import java.io.*;
import java.util.*;

/**
* Write a description of class ByteArrayOutputStreamEx here.
*
* @author (your name)
* @version (a version number or a date)
*/
public class ByteArrayOutputStreamEx
{
    public static void main(String[] args)
    {
        String path ="./Datafiles/";
        String filename1, filename2;
        Scanner sc= new Scanner(System.in);
        System.out.println("\f Enter first filename:");
        filename1 = sc.nextLine();
        System.out.println("Enter the second filename:");
        filename2= sc.nextLine();
        try(FileOutputStream fos1 = new FileOutputStream(path+filename1);
        FileOutputStream fos2= new FileOutputStream(path+filename2))
        {
            ByteArrayOutputStream baos=new ByteArrayOutputStream();
            System.out.print("\n Enter line:");
            String sentence=sc.nextLine();
            byte arr[]= sentence.getBytes();

            baos.write(arr,0,arr.length);
            baos.writeTo(fos1);
            baos.writeTo(fos2);
            baos.flush();
            System.out.println("\n Success...");
        }
        catch(IOException ioe)
        {
          System.err.println("\n Invalid path.");
        }
    }
}
```

ByteArrayInputStream

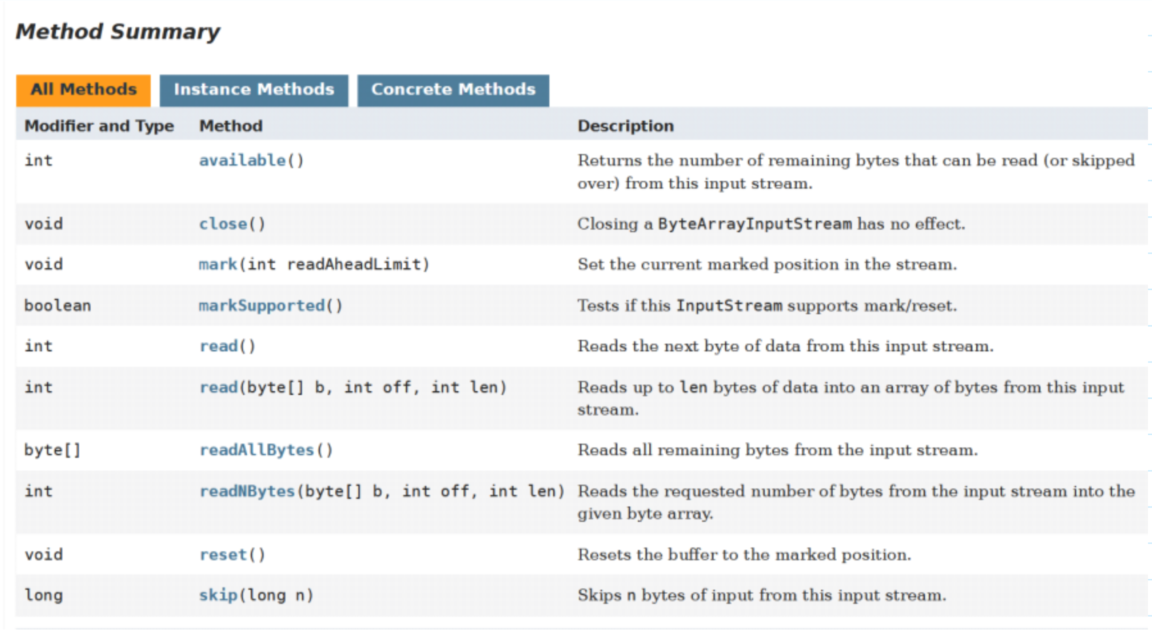
Java ByteArrayInputStream class contains an internal buffer which is used to read byte array as stream. In this stream, the data is read from byte array.

The buffer of ByteArrayInputStream automatically grows according to data.



Constructors

Constructor	Description
`ByteArrayInputStream(byte[] buf)`	Creates a `ByteArrayInputStream` so that it uses `buf` as its buffer array.
`ByteArrayInputStream(byte[] buf, int offset, int length)`	Creates `ByteArrayInputStream` that uses `buf` as its buffer array.



Method Summary

All Methods
Instance Methods
Concrete Methods

Modifier and Type	Method	Description
`int`	`available()`	Returns the number of remaining bytes that can be read (or skipped over) from this input stream.
`void`	`close()`	Closing a `ByteArrayInputStream` has no effect.
`void`	`mark(int readAheadLimit)`	Set the current marked position in the stream.
`boolean`	`markSupported()`	Tests if this `InputStream` supports mark/reset.
`int`	`read()`	Reads the next byte of data from this input stream.
`int`	`read(byte[] b, int off, int len)`	Reads up to `len` bytes of data into an array of bytes from this input stream.
`byte[]`	`readAllBytes()`	Reads all remaining bytes from the input stream.
`int`	`readNBytes(byte[] b, int off, int len)`	Reads the requested number of bytes from the input stream into the given byte array.
`void`	`reset()`	Resets the buffer to the marked position.
`long`	`skip(long n)`	Skips n bytes of input from this input stream.

```
package FileHandling;
import java.io.*;

/**
 * Write a description of class ByteArrayInputStreamExample here.
 *
 * @author (your name)
 * @version (a version number or a date)
 */
public class ByteArrayInputStreamExample
{
    public static void main(String[] args){
        String str = "";
        char x;
        try{
            while((x = (char)System.in.read())!= '0'){
                str += x;
            }

            ByteArrayInputStream byt = new ByteArrayInputStream(str.getBytes());
            int k;
            while((k = byt.read()) != -1){
                System.out.print("\n"+(char)k);
                if(Character.isLetter(k)){
                    System.out.print(" character is an alphabet");
                }
            }
        }
        catch(IOException ioe){
            System.err.print("\nStream error!");
        }
    }
}
```

Output:

```
Now the output is not confusing.0

N character is an alphabet
o character is an alphabet
w character is an alphabet

t character is an alphabet
h character is an alphabet
e character is an alphabet

o character is an alphabet
u character is an alphabet
t character is an alphabet
p character is an alphabet
u character is an alphabet
t character is an alphabet

i character is an alphabet
s character is an alphabet

n character is an alphabet
o character is an alphabet
t character is an alphabet

c character is an alphabet
o character is an alphabet
n character is an alphabet
f character is an alphabet
u character is an alphabet
s character is an alphabet
i character is an alphabet
n character is an alphabet
g character is an alphabet
.
```