Java.io.SequenceInputStream in Java

The SequenceInputStream class allows you to concatenate multiple InputStreams. It reads data of streams one by one. It starts out with an ordered collection of input streams and reads from the first one until end of file is reached, whereupon it reads from the second one, and so on, until end of file is reached on the last of the contained input streams.

Constructor and Description

- **SequenceInputStream(Enumeration e) :** Initializes a newly created SequenceInputStream , which must be an Enumeration that produces objects whose type is InputStream.
- **SequenceInputStream(InputStream s1, InputStream s2) :** Initializes a newly created SequenceInputStream by remembering the two arguments, which will be read in order, first s1 and then s2.

Important Methods:

read : Reads the next byte of data from this input stream.

Syntax:

public int read() throws IOException

Returns: the next byte of data, or -1 if the end of the stream is reached.

Throws: IOException - if an I/O error occurs.

read(byte[] b, int off, int len) : Reads up to len bytes of data from this input stream into an array of

Syntax:

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

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.

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.

Syntax :

public int available() throws IOException

Overrides: available in class InputStream
Returns: an estimate of the number of bytes that can be read (or skipped over) from the current underlying input stream without blocking or 0 if this input stream has been closed by invoking its close() method
Throws:
IOException - if an I/O error occurs.
close: Closes this input stream and releases any system resources associated with the stream.

Syntax :
public void close() throws IOException
Overrides: close in class InputStream
Throws:

IOException - if an I/O error occurs.

The following is an example of SequenceInputStream class that implements some of the important methods.

Program:

```
//Java program to demonstrate SequenceInputStream

import java.io.*;
import java.util.*;
class SequenceISDemp
{
   public static void main(String args[])throws IOException
   {

      //creating the FileInputStream objects for all the following files
      FileInputStream fin=new FileInputStream("file1.txt");
      FileInputStream fin2=new FileInputStream("file2.txt");
      FileInputStream fin3=new FileInputStream("file3.txt");
      //adding fileinputstream obj to a vector object
      Vector v = new Vector();
      v.add(fin);
      v.add(fin2);
      v.add(fin3);
      //creating enumeration object by calling the elements method
     Enumeration enumeration = v.elements();
     //passing the enumeration object in the constructor
     SequenceInputStream sin = new SequenceInputStream(enumeration);
    // determine how many bytes are available in the first stream
      System.out.println("" + sin.available());
      // Estimating the number of bytes that can be read
      // from the current underlying input stream
      System.out.println( sin.available());
      int i = 0;
```

```
        while((i = sin.read())! = -1)
        {
           System.out.print((char)i);
        }
        sin.close();
        fin.close();
        fin2.close();
        fin3.close();
    }
}
```

Output:

19

This is first file This is second file This is third file

Note: This program will not run on online IDE as there are no files associated with it.