| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MultiDoc.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/print/FlavorException.html)   [**NEXT CLASS**](http://docs.google.com/javax/print/MultiDocPrintJob.html) | [**FRAMES**](http://docs.google.com/index.html?javax/print/MultiDoc.html)    [**NO FRAMES**](http://docs.google.com/MultiDoc.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

## **javax.print**

Interface MultiDoc

public interface **MultiDoc**

Interface MultiDoc specifies the interface for an object that supplies more than one piece of print data for a Print Job. "Doc" is a short, easy-to-pronounce term that means "a piece of print data," and a "multidoc" is a group of several docs. The client passes to the Print Job an object that implements interface MultiDoc, and the Print Job calls methods on that object to obtain the print data.

Interface MultiDoc provides an abstraction similar to a "linked list" of docs. A multidoc object is like a node in the linked list, containing the current doc in the list and a pointer to the next node (multidoc) in the list. The Print Job can call the multidoc's [getDoc()](http://docs.google.com/javax/print/MultiDoc.html#getDoc()) method to get the current doc. When it's ready to go on to the next doc, the Print Job can call the multidoc's [next()](http://docs.google.com/javax/print/MultiDoc.html#next()) method to get the next multidoc, which contains the next doc. So Print Job code for accessing a multidoc might look like this:

void processMultiDoc(MultiDoc theMultiDoc) {  
   
 MultiDoc current = theMultiDoc;  
  
 while (current != null) {  
 processDoc (current.getDoc());  
 current = current.next();  
 }  
 }

Of course, interface MultiDoc can be implemented in any way that fulfills the contract; it doesn't have to use a linked list in the implementation.

To get all the print data for a multidoc print job, a Print Service proxy could use either of two patterns:

1. The **interleaved** pattern: Get the doc from the current multidoc. Get the print data representation object from the current doc. Get all the print data from the print data representation object. Get the next multidoc from the current multidoc, and repeat until there are no more. (The code example above uses the interleaved pattern.)
2. The **all-at-once** pattern: Get the doc from the current multidoc, and save the doc in a list. Get the next multidoc from the current multidoc, and repeat until there are no more. Then iterate over the list of saved docs. Get the print data representation object from the current doc. Get all the print data from the print data representation object. Go to the next doc in the list, and repeat until there are no more.

Now, consider a printing client that is generating print data on the fly and does not have the resources to store more than one piece of print data at a time. If the print service proxy used the all-at-once pattern to get the print data, it would pose a problem for such a client; the client would have to keep all the docs' print data around until the print service proxy comes back and asks for them, which the client is not able to do. To work with such a client, the print service proxy must use the interleaved pattern.

To address this problem, and to simplify the design of clients providing multiple docs to a Print Job, every Print Service proxy that supports multidoc print jobs is required to access a MultiDoc object using the interleaved pattern. That is, given a MultiDoc object, the print service proxy will call [getDoc()](http://docs.google.com/javax/print/MultiDoc.html#getDoc()) one or more times until it successfully obtains the current Doc object. The print service proxy will then obtain the current doc's print data, not proceeding until all the print data is obtained or an unrecoverable error occurs. If it is able to continue, the print service proxy will then call [next()](http://docs.google.com/javax/print/MultiDoc.html#next()) one or more times until it successfully obtains either the next MultiDoc object or an indication that there are no more. An implementation of interface MultiDoc can assume the print service proxy will follow this interleaved pattern; for any other pattern of usage, the MultiDoc implementation's behavior is unspecified.

There is no restriction on the number of client threads that may be simultaneously accessing the same multidoc. Therefore, all implementations of interface MultiDoc must be designed to be multiple thread safe. In fact, a client thread could be adding docs to the end of the (conceptual) list while a Print Job thread is simultaneously obtaining docs from the beginning of the list; provided the multidoc object synchronizes the threads properly, the two threads will not interfere with each other

| **Method Summary** | |
| --- | --- |
| [Doc](http://docs.google.com/javax/print/Doc.html) | [**getDoc**](http://docs.google.com/javax/print/MultiDoc.html#getDoc())()            Obtain the current doc object. |
| [MultiDoc](http://docs.google.com/javax/print/MultiDoc.html) | [**next**](http://docs.google.com/javax/print/MultiDoc.html#next())()            Go to the multidoc object that contains the next doc object in the sequence of doc objects. |

| **Method Detail** |
| --- |

### getDoc

[Doc](http://docs.google.com/javax/print/Doc.html) **getDoc**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Obtain the current doc object.

**Returns:**Current doc object. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Thrown if a error ocurred reading the document.

### next

[MultiDoc](http://docs.google.com/javax/print/MultiDoc.html) **next**()  
 throws [IOException](http://docs.google.com/java/io/IOException.html)

Go to the multidoc object that contains the next doc object in the sequence of doc objects.

**Returns:**Multidoc object containing the next doc object, or null if there are no further doc objects. **Throws:** [IOException](http://docs.google.com/java/io/IOException.html) - Thrown if an error occurred locating the next document

| | [**Overview**](http://docs.google.com/overview-summary.html) | [**Package**](http://docs.google.com/package-summary.html) | **Class** | [**Use**](http://docs.google.com/class-use/MultiDoc.html) | [**Tree**](http://docs.google.com/package-tree.html) | [**Deprecated**](http://docs.google.com/deprecated-list.html) | [**Index**](http://docs.google.com/index-files/index-1.html) | [**Help**](http://docs.google.com/help-doc.html) | | --- | --- | --- | --- | --- | --- | --- | --- | | | ***Java™ Platform***  ***Standard Ed. 6*** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| [**PREV CLASS**](http://docs.google.com/javax/print/FlavorException.html)   [**NEXT CLASS**](http://docs.google.com/javax/print/MultiDocPrintJob.html) | [**FRAMES**](http://docs.google.com/index.html?javax/print/MultiDoc.html)    [**NO FRAMES**](http://docs.google.com/MultiDoc.html)     [**All Classes**](http://docs.google.com/allclasses-noframe.html) |
| SUMMARY: NESTED | FIELD | CONSTR | [METHOD](#3znysh7) | DETAIL: FIELD | CONSTR | [METHOD](#2et92p0) |

[Submit a bug or feature](http://bugs.sun.com/services/bugreport/index.jsp)

For further API reference and developer documentation, see [Java SE Developer Documentation](http://docs.google.com/webnotes/devdocs-vs-specs.html). That documentation contains more detailed, developer-targeted descriptions, with conceptual overviews, definitions of terms, workarounds, and working code examples.

Copyright 2006 Sun Microsystems, Inc. All rights reserved. Use is subject to [license terms](http://docs.google.com/legal/license.html). Also see the [documentation redistribution policy](http://java.sun.com/docs/redist.html).