CS 535 Object-Oriented Programming & Design
Fall Semester, 2003
Doc 8 Streams & Files

Streams 2
Examples 4
WriteStream 4
ReadStream Examples 10
Files 13

References

VisualWorks Application Developer Guide, doc/vwadg.pdf in the VisualWorks installation. Chapter 20 Files.

http://www.iam.unibe.ch/~ducasse/WebPages/Smalltalk/ST00_01.pdf

Reading

VisualWorks Application Developer Guide, doc/vwadg.pdf in the VisualWorks installation. Chapter 20 Files.

Streams

Iterate or traverse over
- Sequenceable Collections
- File contents

Maintains pointer to current position in collection

Basic Streams

Stream
  PositionableStream
    ReadStream
    WriteStream
      ReadWriteStream

Stream Methods

next	Returns the next element
next: n	Returns the n next elements
nextPut: anElement	Inserts anElement at next position
nextPutAll: aCollection	Inserts collection elements starting at the next position
contents	Returns all the elements
flush	Write any unwritten information
atEnd	true if at the end of the collection
cr space tab crtab	Write the specified white space
print: anObject	Print anObject on the stream

PeekableStream Methods

skip: n	Increases the position by n
skipTo: anElement	Increases the position to after anElement
upToSeparator	Return contents up to a separator, skip over separtor
reset	Set position to 0
peek	Return next element, position not changed
peekFor: anObject	Return true if next element = anObject

Examples
WriteStream

Write streams must be created on a collection
The stream grows the underlying collection if needed

String Example

```
| x |
x := WriteStream on: String new.
x
   nextPut: $A;
   nextPutAll: ' Cat in the Hat';
   nextPutAll: ' Comes Back';
   contents
```

Result

'A Cat in the Hat Comes Back'

Array Example

```
x := WriteStream on: Array new.
x
   nextPut: 5;
   nextPut: 'cat';
   nextPut: $a.
x contents
```

Result

#(5 'cat' $a)

Collection Backing the Stream

A stream backed by a String

WriteStream on: String new.

A stream backed by an Array

An Array backs the write stream

Backing collection may limit the type of elements added

x := WriteStream on: String new.

x nextPut: 56. "Runtime error, must be character"

x nextPut: 56 printString. "Error, string is not a character"

x print: 56 "OK"

x nextPutAll: 56 printString "OK"

nextPut: & nextPutAll:

nextPut: adds one element to the stream

nextPutAll:

Argument must be a collection

Elements of the argument are added one at a time to the collection

Repositioning of the stream

```
(WriteStream on: String new)
   nextPutAll: 'Cat in the Hat';
   position: 4;
   nextPutAll: 'Comes Back';
   contents
```

Result

'Cat Comes Back'

WriteStream Examples Continued

You can start with a large collection to avoid the need to grow

But don't worry about it

```
(WriteStream on: (String new:40))
   nextPutAll: 'Cat in the Hat';
   nextPutAll: ' Comes Back';
   contents
```

Result

```
 'Cat in the Hat Comes Back'
```

Streams can be on any sequential collection

```
(WriteStream on: Array new)
   nextPut: $A;
   nextPutAll: 'Cat-in-the-Hat';
   nextPutAll: 'Comes-Back';
   contents
```

Result

```
 #($A $C $a $t $- $i $n $- $t $h $e $- $H $a $t $C $o $m $e $s $- $B
$a $c $k)
```

WriteStream Examples Continued

Note the difference between adding a string and an array containing a string

```
(WriteStream on: Array new)
   nextPut: $A;
   nextPutAll: #('Cat in the Hat');
   nextPutAll: #('Comes Back');
   contents
```

Result

```
 #($A 'Cat in the Hat' 'Comes Back')
```

```
(WriteStream on: Array new)
   nextPut: $A;
   nextPutAll: 'Cat-in-the-Hat';
   nextPutAll: 'Comes-Back';
   contents
```

Result

```
 #($A $C $a $t $- $i $n $- $t $h $e $- $H $a $t $C $o $m $e $s $- $B
$a $c $k)
```

ReadStream Examples

```
| x |
x := ReadStream on: 'Cat-in-the-Hat-Comes-Back'.
```

Transcript	"Result in Transcript"
print: x next; cr;	C
print: x peek; cr;	a
print: x next; cr;	a
show: (x upTo: $e); cr;	t-in-th
show: (x upToAll: 'Comes'); cr;	-Hat-
show: x upToEnd; cr;	Comes-Back
show: x contents	Cat-in-the-Hat-Comes-Back

upTo: and upToAll:

Return up to the indicated element (collection)
upTo: sets the position after the element ($e)
upToAll: sets the position at the start of the indicated collection

ReadStream on an Array

```
| x |
x := ReadStream on:
   #('Cat' 'in' 'the' 'Hat' 'Comes' 'Back' 'Again' 'by' 'Zeus').
```

Transcript	"Result in Transcript"
show: x next; cr;	Cat
show: x peek; cr;	in
show: x next; cr;	in
show: (x upTo: 'Comes'); cr;	#('the' 'Hat')
show: (x upToAll: #('Again' 'by')); cr;	#('Back')
show: x upToEnd	#('Again' 'by' 'Zeus')

Comment on ReadStream Example

The elements returned by the stream are elements in the underlying collection

upTo: requires elements of the underlying collection

upToAll: requires a collection of elements of the underlying collection

next returns an element of the underlying stream

ReadStream on: 'Cat-in-the-Hat-Comes-Back'.

The read stream is on a string

next returns a character

upToAll: requires a collection of characters

ReadStream on: #('Cat' 'in' 'the' 'Hat' 'Comes' 'Back' 'Again').

The read stream is on an array of strings

next returns strings

upToAll: requires a collection of strings

Files

Filename
  Main file class

Example

```
| name file fileWrite fileRead|
name := 'sampleFile'.
file := name asFilename.
fileWrite := file writeStream.
fileWrite
   nextPutAll: 'Hello world';
   nextPutAll: 'How are you?';
   cr;
   close.

fileRead := file readStream.
Transcript show: fileRead contents.
fileRead close.
fileAppend := file appendStream.
fileAppend
   nextPutAll: 'I am well';
   cr;
   close.
Transcript show: file contentsOfEntireFile
```

Creating

Filename named: 'filename'

'filename' asFilename

Both create a Filename object on a file

The filename string is a
file in the current directory or
Full path to the file

Writing to a File

Filename>>writeStream
- Opens a write stream on the file
- If file does not exits create the file
- If file does exist erase current contents

Filename>>appendStream
- Returns a write stream on the file
- If file does not exits create the file
- If file does exist the stream appends to the contents

Filename>>readStream
- Returns a read stream on the file
- File must exist
- Stream reads from the beginning of the file

Closing file streams

Always close streams on files

If you do not close the stream, the VM keeps the file open

Exceptions and Closing files

What happens when an exception is raised before you close a file?

```
| name file fileWrite |
name := 'sampleFile'.
file := name asFilename.
fileWrite := file writeStream.
1 /0.
fileWrite
   nextPutAll: 'Hello world';
   nextPutAll: 'How are you?';
   cr;
   close.
```

Using Ensure to close files

```
| name file fileWrite |
name := 'sampleFile'.
file := name asFilename.
[fileWrite := file writeStream.
1 /0.
fileWrite
   nextPutAll: 'Hello world';
   nextPutAll: 'How are you?';
   cr.]

   ensure: [fileWrite close].
```

Some File Operations in Filename

isDirectory	Returns true if Filename object is a directory
fileSize	Returns size of the file represented by filename object
delete	Delete the file or directory represented by filename object
directoryContents	Returns the contents of a filename object that represents a directory
makeDirectory	Make the filename object a directory.