CS 683 Emerging Technologies: Embracing Change
Spring Semester, 2001
Doc 11 Streams & Files
Contents

Streams 2
Examples 4
WriteStream 4
ReadStream Examples 7
Files 9
FileDirectory 10
FileStream 14

References

Squeak source code

Object-Oriented Design with Smalltalk — a Pure Object Language and its Environment, Ducasse, University of Bern, Lecture notes 2000/2001, http://www.iam.unibe.ch/~ducasse/WebPages/Smalltalk/ST00_01.pdf

Copyright©, All rights reserved. 2001 SDSU & Roger Whitney, 5500 Campanile Drive, San Diego, CA 92182-7700 USA. OpenContent (http://www.opencontent.org/opl.shtml) license defines the copyright on this document.

Streams

Iterate or traverse over
- Sequenceable Collections
- File contents

Maintain pointer to current position in collection

Basic Streams

Stream
  PositionableStream
    ReadStream
    WriteStream
      ReadWriteStream
        FileStream
      TextStream
      Transcription Stream

Some Other Streams

DataStream
GZipReadStream
GZipWriteStream
HtmlTokenizer
ObjectSocket
ZipEncoder

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
position	Returns current position
atEnd	true if at the end of the collection

PeekableStream Methods

skip: n	Increases the position by n
skipTo: anElement	Increases the position to after anElement
position: anInteger	Set position to given index
reset	Set position to 0
peek	Return next element, position not changed
peekFor: anObject	Return true if next element = anObject

Examples
WriteStream

The stream grows the underlying collection if needed

```
| 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'

Showing some 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')
```

with: allows us to start with a collection with elements

```
(WriteStream with: 'A Cat in the Hat')
   nextPutAll: ' Comes Back';
   contents
```

Result

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

ReadStream Examples

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

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

upTo: and upToAll:

Return up to the indicated element (collection)

Next return element is after indicated element (collection)

```
| 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	#('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

Important Classes

- FileDirectory
- FileStream

Platform dependent versions do exist, but do not use directly

The above classes will use the proper classes for the current platform

FileDirectory

Represents a directory in the underlying platform's file system

It knows the fully qualified path name for itself

Can enumerate the files and directories within itself

For each file and subdirectory it stores:

#(<name> <creationTime> <modificationTime> <dirFlag> <fileSize>)

Class Methods

instance creation

default	forFileName:
on:	root

name utilities

baseNameFor:	changeSuffix
checkName:fixErrors:	dirPathFor:
extensionFor:	fileName:extension:
isLegalFileName:	localNameFor:
searchAllFilesForAString	splitName:to:

create/delete file

deleteFilePath:

platform specific

dot	extensionDelimiter
isCaseSensitive	maxFileNameLength
pathNameDelimiter	slash

Instance Methods

path access

fullPathFor:	on:	pathName
pathNameDelimiter	pathParts	slash

File stream creation

fileNamed:	forceNewFileNamed:	newFileNamed:
oldFileNamed:	oldFileOrNoneNamed:	readOnlyFileNamed:

enumeration

containingDirectory	directoryNamed:	directoryNames
entries	fileAndDirectoryNames	fileNames
fullNamesOfAllFilesInSubtree	keysDo:	localName
statsForDirectoryTree:		

testing

directoryExists:	fileExists:	fileOrDirectoryExists:
includesKey:	isCaseSensitive	

File operations

copyFile:toFile:	copyFileNamed:toFileNamed:
createDirectory:	deleteDirectory:
deleteFileNamed:	deleteFileNamed:ifAbsent:
getMacFileTypeAndCreator:	mimeTypesFor:
putFile:named:	putFile:named:retry:
rename:toBe:	setMacFileNamed:type:creator:

File name utilities

checkName:fixErrors:	fileNamesMatching:	fullNameFor:
isLegalFileName:	nextNameFor:extension:	realUrl
	splitNameVersionExtensionFor:	url

File directory

assureExistance	assureExistanceOfPath:	localNameFor:
sleep	wakeUp	

searching

filesContaining:caseSensitive:
withAllFilesDo:andDirectoriesDo:

Sample Program

"Prompt user for a start directory
Find all files ending in .html in all subdirectories recursively"

```
| path |

path := FillInTheBlank request: 'Enter a path'.
(FileDirectory isLegalFileName: path) ifFalse:[^nil].
startLocation := FileDirectory on: path.
startLocation fullNamesOfAllFilesInSubtree
   select: [:each | each endsWith: '.html']
```

FileStream

Represents a read & write stream on a file

Class Methods for Instance Creation

fileNamed:	fullName:	isAFileNamed:
new	newFileNamed:	oldFileNamed:
oldFileOrNoneNamed:	readOnlyFileNamed:	

Instance Methods

accessing

contentsOfEntireFile	next	next:
nextPut:	nextPutAll:	size

testing

atEnd

positioning

position	position:	reset
setToEnd	skip:	

file open/close

close	closed	flush	reopen

Make sure you close your files

file modes

ascii	binary	readOnly
readWrite	text	

Some Simple File Actions

```
| file |
file := FileStream fileNamed: 'Roger'.
file
   nextPutAll: 'This is a string';
   nextPut: Character cr;
   nextPutAll: 'A Second line'.
file contentsOfEntireFile
```

contentsOfEntireFile closes the file

```
| file |
file := FileStream fileNamed: 'Roger'.
[file atEnd] whileFalse:
   [Transcript show: file next].
file close
```