CS 535 Object-Oriented Programming & Design
Fall Semester, 2008
Doc 9 Exceptions, Streams, Files
Sept 25 2008

References

Cincom Smalltalk Basic Libraries Guide, doc/BasicLibraries.pdf, Chapter 2 Streams

Cincom Smalltalk Application Developer's Guide, doc/AppDevGuide.pdf, Chapter 13 Files, Chapter 15 Exception and Error Handling

Reading

Smalltalk by Example, Alex Sharp,
Chapter 13 Streams
Chapter 14 Files

Exceptions

Basic Issues

How are exceptions raised (started)
How to handle exceptions
What can one do when handling exceptions
How is the correct handler found for an exception

Basic Handling of Exceptions

```
[ProtectedBlock]
    on: ExceptionList
    do: [:exception | HandlerBlock]
```

```
[numerator := 5.
denominator := 0.0.
numerator / denominator]
  on: ZeroDivide
  do:
    [:exception |
    Transcript
      show: exception description;
      cr]
```


Unlike Java, in Smalltalk zero divide by both integer and floats cause a zero divide exception to be raised

Catching Multiple Exceptions

Use a comma or ExceptionSets

```
[1/0]
   on: Warning , ZeroDivide
   do: [:exception | code here]
```

```
 | exceptions |
exceptions := ExceptionSet with: Warning with: ZeroDivide.
[1/0]
   on: exceptions
   do: [:exception | code here]
```

ensure:

[block] ensure: [clean up block]

Ensure that the clean up block will be done

If block ends due to an exception
    Execute handler for exception
Execute clean up block

Example

```
[[10/0] ensure: [Transcript show: 'In ensure'; cr]]
   on: ZeroDivide
   do: [:exception | Transcript show: 'In handler';cr ]
```

Output in Transcript

```
In handler
In ensure
```


Your code should not depend on the order of execution of the handler and clean up block

ifCurtailed:

[block] ifCurtailed: [clean up block]

Clean up block is done only if [block] ends abnormally

Raising Exceptions

Implicitly Raised Exceptions

12 / 0

Explicitly Raised Exceptions

Send message to an exception class

Warning raiseSignal: 'This string is the signal description'
Error raiseSignal
Error raiseSignal: 'Problem here'

Object Methods That Raise Exception

self error: 'Error message'

Simplest way to raise an exception

self halt

self halt: 'Message"

Raises Halt exception.

Allows user to invoke debugger or resume

self shouldNotImplement

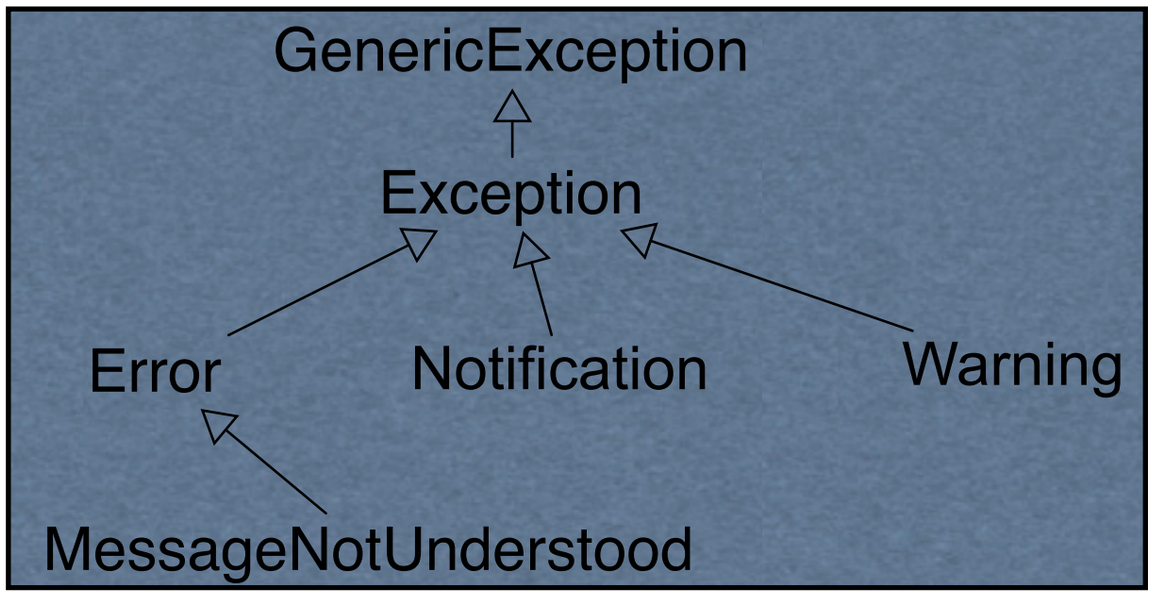
Used in subclasses in inherited methods that do not belong in the subclass

self subclassResponsibility

Used in methods to declare them abstract

Indicated subclasses must implement this method

Exceptions are Classes



Error

Normal Exception behavior

Your exceptions should subclass Error

Notification

Something interesting has occurred

If it is not handled, it will pass by without effect

Warning

An unusual event the user needs to know about

Asks the user if the program should continue

MessageNotUnderstood

A method was sent to an object that does not implement it

Exception Default Action

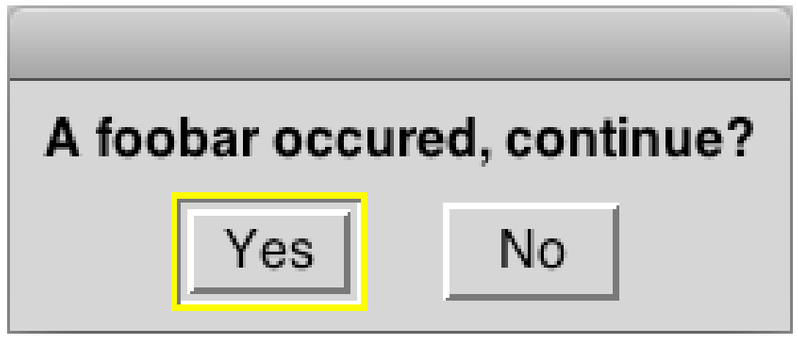
All exceptions have default action

What happens if exception is not caught in on:do:

Warning Default Behavior

Warning raiseSignal: 'An error occured, continue?'.

Result



Warning with handler

```
[Warning raiseSignal: 'Hi Mom'.
Transcript show: 'End']
   on: Warning
   do: [:exception | Transcript show: 'Handler']
```

Output in Transcript

Handler

No dialog window opens as the default behavior is not done when a warning is raised in a on:do: block

Finding the Exception Handler

When an exception is raised the enclosing handlers are searched

- Start with the code that raised the exception
- Search the "closest" enclosing handler first
- Continue searching the enclosing handers

The first handler that deals with the exception is used

If no handlers handle the exception the exception's default action is done

```
[[1/0]
   on: ZeroDivide
   do: [:exception | Transcript show: 'First']]
      on: ZeroDivide
      do: [:exception | Transcript show: 'Second']
```

Inheritance and Exception

All subexceptions are caught by an exception in on:do:

ZeroDivide is a subclass of Error

The ZeroDivide exception will be caught in the following

```
[1/0]
   on: Error
   do:
      [:exception |
      Transcript
         show: exception description;
         cr]
```

Resumable Exceptions

```
| result |
[result := 10 / 0 + 5.
Transcript show: result printString]
      on: ZeroDivide
      do: [:exception | exception resume: 1]
```

Output in Transcript

6

```
| result |
[result := 10/0.
Transcript show: result printString]
   on: ZeroDivide
   do:
      [:exception |
      exception resume ]
```

Output in Transcript

nil

retry

```
| x y result |
x := 10.
y := 0.
[result := x / y.
Transcript show: result printString]
   on: ZeroDivide
   do:
      [:exception |
      y := 1.
      exception retry ]
```

Output in Transcript

10

Creating Your Own Exceptions

Subclass the correct existing Exception

Almost always Error

If you want the exception to be resumable

Make method isResumable return true

If you want non-standard default behavior

Override the method defaultAction

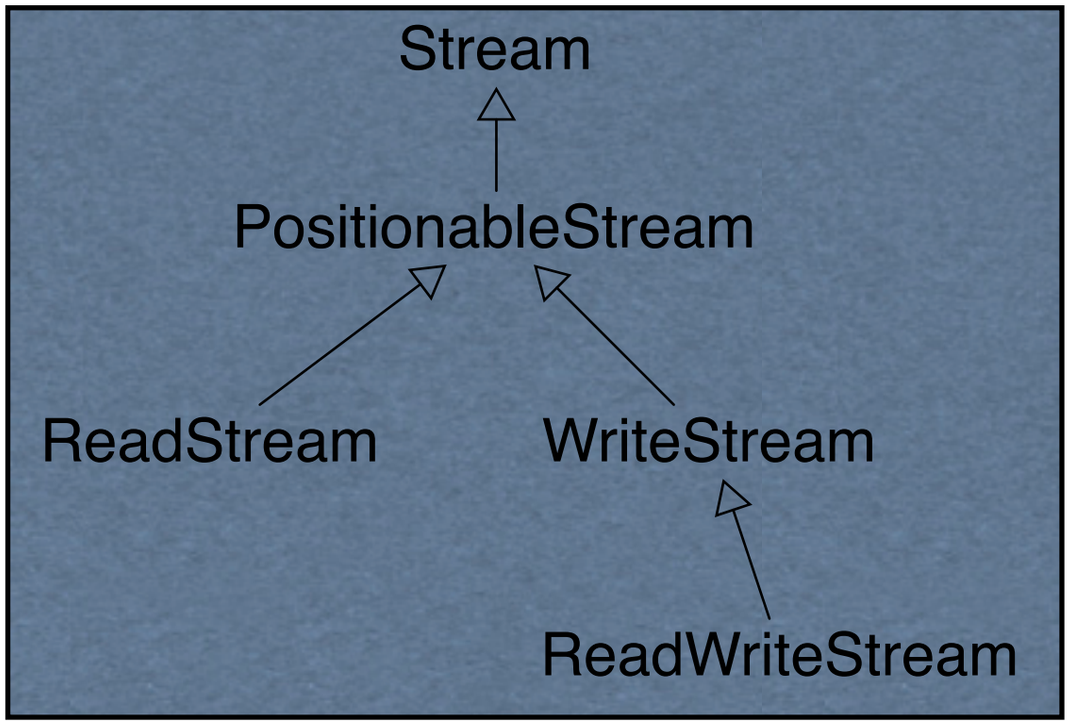
Streams

Streams

Iterate or traverse over

- Sequenceable Collections
- File contents

Maintains pointer to current position in collection



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

If these look familiar the Transcript is a 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

WriteStream Examples

```
| 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'

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

Result

#(5 'cat' $a)

nextPut: & nextPutAll:

```
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:

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

Explain This

```
| x |
x := WriteStream on: Array new.
x
   nextPut: 'cat';
   nextPut: 'in';
   nextPut: 'hat'.
x contents
```

Result

#('cat' 'in' 'hat')

```
| x |
x := WriteStream on: Array new.
x
   nextPutAll: 'cat';
   nextPutAll: 'in';
   nextPutAll: 'hat'.
x contents
```

Result

#($c $a $t $i $n $h $a $t)

Repositioning of the stream

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

Result

'Cat Comes Back'

ReadStream Examples

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

Note that in the output we do not see the $e from the x upTo: $e, but do see the 'Comes' in the output from the x upToAll: 'Comes'. upTo: sets the position after the element ($e). upToAll: sets the position at the start of the indicated collection

ReadStream on an Array

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

ReadStream

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

Most uses have String as underlying collection

Files

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
```

File Objects

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

Close your Files

Always close streams on files

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

```
| 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.