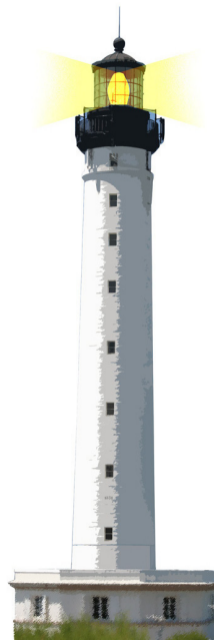


Using well asString and printString

the stream interplay

S. Ducasse



Goal

- Think about **intermediary object creation**
- How to avoid creating spurious objects
- `asString` **and** `printString` **inside** `printOn`:



printString: setting the stage

```
Object >> printString  
^ self printStringLimitedTo: 50000
```

```
Object >> printStringLimitedTo: limit  
^ self printStringLimitedTo: limit using: [:s | self printOn: s]
```

printStringLimitedTo:using:

streamContents: [:s | self printOn: s] creates a stream and gather object textual representation

```
Object >> printStringLimitedTo: limit
| limitedString |
limitedString := String
                    streamContents: [ :s | self printOn: s ]
                    limitedTo: limit.
limitedString size < limit ifTrue: [ ^ limitedString ].
^ limitedString , '...etc...'
```

The case of displayStringOn:

- When we get a stream, better use it directly

```
MessageTally >> displayStringOn: aStream  
  self displayIdentifierOn: aStream.  
  aStream  
    nextPutAll: '(';  
    nextPutAll: self tally printString;  
    nextPutAll: ')'
```

self tally printString

- Creates a new stream
- Get its contents to be able to put in the first stream



Better

```
MessageTally >> displayStringOn: aStream  
self displayIdentifierOn: aStream.  
aStream  
  nextPutAll: '(';  
  print: self tally;  
  nextPutAll: ')'
```

```
Stream >> print: anObject  
"Have anObject print itself on the receiver."  
  
anObject printOn: self
```

- No creation of intermediary streams



Another case of misuse

```
printProtocol: protocol sourceCode: sourceCode
```

```
^ String streamContents: [ :stream |  
  stream  
  nextPutAll: "'protocol: '";  
  nextPutAll: protocol printString;  
  nextPut: $"; cr; cr;  
  nextPutAll: sourceCode ]
```

protocol printString

- Creates a new stream
- Get its contents to be able to put in the first stream



Better use print:

```
printProtocol: protocol sourceCode: sourceCode
```

```
^ String streamContents: [ :stream |  
  stream  
    nextPutAll: "'protocol: '  
    print: protocol;  
    nextPut: $"; cr; cr;  
    nextPutAll: sourceCode ]
```


About asString

asString has the similar issues than printString

```
Object >> asString
```

```
"Answer a string that represents the receiver."
```

```
^ self printString
```

- asString should be used when we convert an object to its string representation
- now check before calling it inside a streamContents

Conclusion

- Check protocols (`printString`, `printOn:`, `asString`)
- Read code around
- Streams are powerful containers



A course by

S. Ducasse, L. Fabresse, G. Polito, and Pablo Tesone



Except where otherwise noted, this work is licensed under CC BY-NC-ND 3.0 France
<https://creativecommons.org/licenses/by-nc-nd/3.0/fr/>