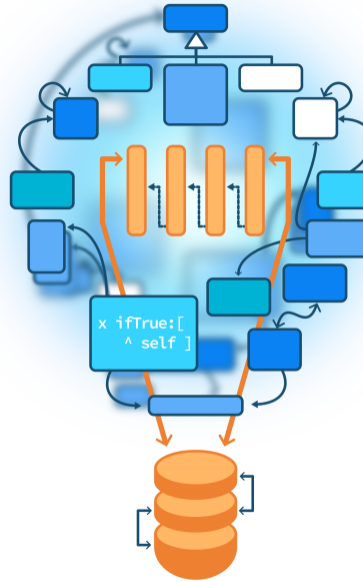


Using well asString and printString

A Pharo code idiom

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



Goal

- Think about **intermediary object creation**
- Avoid creating spurious objects
- Use `asString` and `printString` **inside** `printOn`:



printString: setting the stage

- Can get a textual representation of any object

```
> Apple new printString  
'an Apple'
```

```
> Date today printString  
'2 April 2023'
```



printString: implementation

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

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

- `streamContents: [:s | self printOn: s]` creates a stream and gather object textual representation
- Each object can place text inside the stream



Some suboptimal use: The case of displayStringOn:

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

- When we get a stream, better use it directly
- self tally printString
 - creates a **new** stream
 - and gets its contents 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;  
  cr; cr;  
  nextPutAll: sourceCode ]
```

protocol printString

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



Better use print:

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

```
^ String streamContents: [ :stream |  
  stream  
    nextPutAll: 'protocol: '  
    print: protocol;  
    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
- Check before calling it inside a streamContents



Conclusion

- Check protocols (`printString`, `printOn:`, `asString`)
- Read code around
- Streams are powerful containers
 - can be passed around
 - no need to create intermediary streams for basic sub processes



Produced as part of the course on <http://www.fun-mooc.fr>

Advanced Object-Oriented Design and Development with Pharo

A course by

S.Ducasse, L. Fabresse, G. Polito, and P. 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/>