About Null Check
The case of lazy initialization
S. Ducasse, L. Fabresse, G. Polito, and P. Tesone

http://www.pharo.org
Goals

- Think about object initialization
- Present **Lazy Initialization**
- Complement to ’Avoid Nil’ Lectures
Problem

- Need to reduce startup time
- How can we do less at the beginning?
- Sometimes you do not want to be forced to initialize all the state at instance creation time
Solution

- **Only** perform initialization if the state is used
- Delay initialization until needed
Lazy initialization

- Let nil value in instance variable
- Do not initialize instance variable at *instantiation time*
- Do not expose instance variable nil
  - Do not access instance variable *directly*
- Only access instance variable via a *lazy accessor*
Lazy accessor

MyObject >> x
^ x ifNil: [ x := 0]
Example of Lazy Initialization

You defer the initialization of the variable to its first use

```
FreeTypeFont >> descent
^ cachedDescent ifNil: [
  cachedDescent := (self face descender * self pixelSize //
                   self face unitsPerEm) negated ]
```

- This is only when the method descent is executed that cachedDescent will be initialized
Solution: Use Lazy Initialization when Necessary

- **Defer** initialization and caches the result
- Pay attention you should NOT access directly an instance variable used in a lazy setting
- You should **always use the lazy accessor**
- Else you expose to *nil* value and will force client to check
Pros/Cons

- Lazy initialization trade execution at instance creation time for a check at each execution `ifNil:`
Conclusion

- Lazy initialization is another tool at hand
- Don’t overuse it
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/