

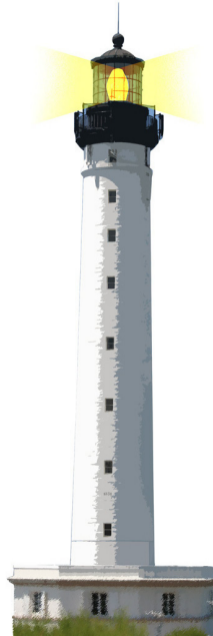
Advanced Object-Oriented Design

Class Methods At Work

S. Ducasse and L. Fabresse



<http://www.pharo.org>



What you will learn

- In Pharo class methods are normal virtual methods
 - methods are looked up dynamically
- Most class methods create new instances
 - but they can be used for other things



Example: Creating the right document elements from lines

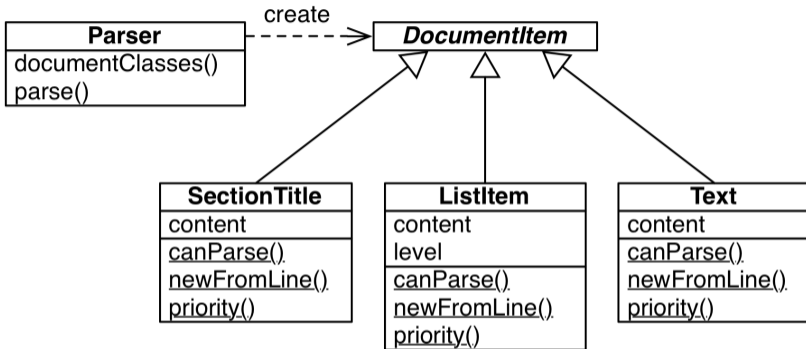
Imagine we want to parse the following and create the corresponding objects

```
!Section Title
- list item
-- subitem

Any text here
```



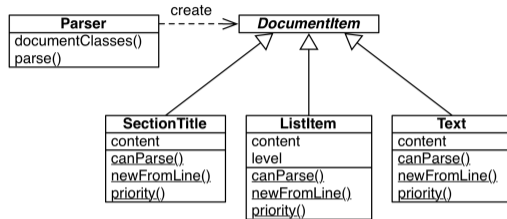
A possible design



Document item **classes** know

- if they can parse a line (`canParse()`)
- how to create instances (`newFromLine()`)

Parsing lines



```
Parser >> documentClasses
  ^ DocumentItem allSubclasses
    sorted: [ :class1 :class2 | class1 priority < class2 priority ]
```

```
Parser >> parse: line
  self documentClasses
  detect: [ :subclass |
    (subclass canParse: line)
    ifTrue: [ ^ subclass newFromLine: line ] ]
```

The command-line handler

- The Pharo command-line interface (CLI) uses the same approach
- each subclass of `CommandLineHandler` knows how to deal with one command
- the correct subclass is selected by sending messages to the class

```
$ pharo Pharo.image eval "10 factorial"  
3628800
```



The command-line handler

```
CommandLineHandler class >> isResponsibleFor: arguments  
  ^ arguments includesSubCommand: self commandName
```

```
EvaluateCommandLineHandler class >> commandName  
  ^ 'eval'
```

```
CommandLineHandler class >> allHandlers  
  ^ self allSubclasses  
    reject: [ :handler | handler isAbstract ]
```

```
CommandLineHandler class >> handlersFor: arguments  
  ^ self allHandlers  
    select: [ :handlerClass |  
      handlerClass isResponsibleFor: arguments ]
```



Pay attention

- This is **costly** to check all subclasses all the times
- Do you need such a dynamic behavior?
- Are you loading that many different classes?
 - In pillar this is not really needed
 - For the command line, each application may define its own commandes



Conclusion

- Classes are objects and can be sent messages
- Method lookup is exactly the same as for all objects:
 - go to the class of the receiver
 - follow inheritance chain
- Pharo makes it easy to iterate over subclasses
 - but this is **costly**
 - check next Lectures on registration



A course by

S. Ducasse, 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/>