

Class Methods At Work

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



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



Case study: parsing a string

Imagine we want to parse the following string:

!Section Title

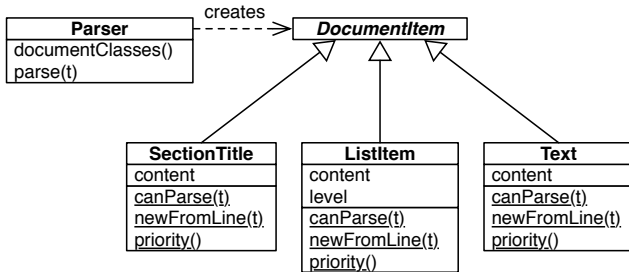
– list item

-- subitem

Any text here

and create the corresponding objects.

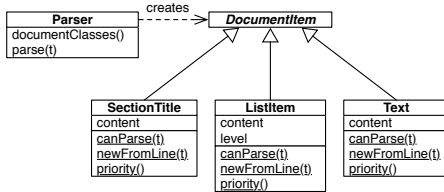
A possible design



Each **DocumentItem** subclass knows

- if it can parse a line (`canParse:`)
- how to create an instance of itself (`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 Pharo command-line interface (CLI)

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

- it uses the same approach
- each subclass of `CommandLineHandler` processes one type of command
- the correct subclass is selected by sending messages to the class

The command-line handler

```
CommandLineHandler class >> handlersFor: arguments
```

```
  ^ self allHandlers  
    select: [ :handlerClass |  
      handlerClass isResponsibleFor: arguments ]
```

```
CommandLineHandler class >> allHandlers
```

```
  ^ self allSubclasses  
    reject: [ :handler | handler isAbstract ]
```

```
CommandLineHandler class >> isResponsibleFor: arguments
```

```
  ^ arguments includesSubCommand: self commandName
```

```
EvaluateCommandLineHandler class >> commandName
```

```
  ^ 'eval'
```



Evaluation

Pros:

- Modular design
- Extensible

Cons:

- Checking all subclasses all the times is **costly**
- Do you need such a dynamic behavior?
 - For the command line, each application may define its own commands



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
 - it enables modular and extensible design
 - but this is **costly**
- Related to the lecture on *Registration*



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/>