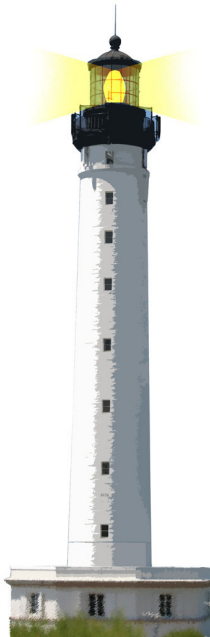
Learning Object-Oriented
Programming and Design with TDD

Class Methods At Work

Stéphane Ducasse
http://stephane.ducasse.free.fr



Pharo
http://www.pharo.org

W6S03

What You Will Learn

- Class methods are normal methods
- Most class methods create new instances
  - but they can be used for other things

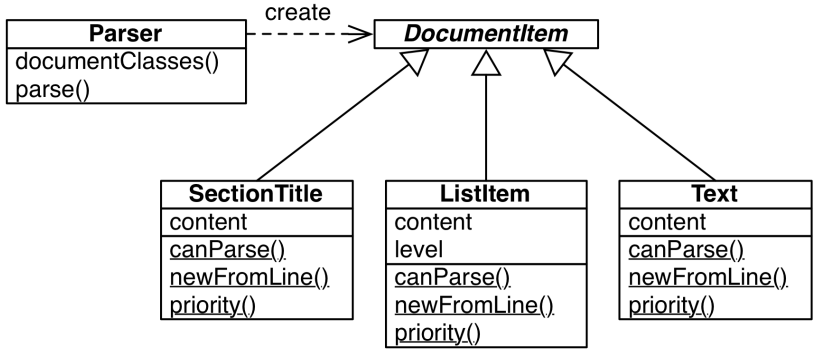
Parsing Lines

Imagine we want to parse

```
!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:)

A Possible Design: canParse:

```
SectionTitle class >> canParse: aLine
  aLine size > 1 ifFalse: [ ^ false ].
  ^ aLine beginsWith: '!'
```

```
ListItem class >> canParse: aLine
  aLine size > 1 ifFalse: [ ^ false ].
  ^ aLine beginsWith: '–' or: [ aLine beginsWith: '––' ]
```

```
Text class >> canParse: aLine
  ^ true
```

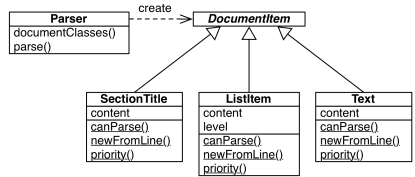
A Possible Design: newFromLine:

```
SectionTitle class >> newFromLine: aLine
  ^ self new contents: aLine allButFirst; yourself
```

```
ListItem class >> newFromLine: aLine
  aLine beginsWith: '–'
    ifTrue: [ ^ ListItem new level: 1; contents: aLine allButFirst ].
  aLine begingsWith: '––'
    ifTrue: [ ^ ListItem new level: 3; contents: aLine allButFirst: 2]
```

```
Text class >> newFromLine: aLine
  ^ Text new contents: aLine
```

Parsing Lines




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

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

Stepping back

- Class methods are dynamically resolved
- Classes are used as a dynamic registration system
- Adding a new class adds a new functionality modularly
- Can be incrementally defined (in different packages)

About class method lookup

- Method lookup is **exactly** the same as for all objects:
  - go to the class of the receiver
  - follow inheritance chain
- In Pharo there is ONLY ONE method lookup.
- More during the lecture *Understanding Metaclasses*
- Pharo makes it easy to iterate over subclasses

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 Eval Command-Line Handler

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

EvaluateCommandLineHandler class >> description
  ^ 'Directly evaluates passed in one line scripts'
...

EvaluateCommandLineHandler >>evaluateArguments
  | argumentString |
  argumentString :=  self argumentString.
  argumentString ifEmpty: [ ^ self ].
  self evaluate: argumentString
```

The Command-Line Handler

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

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

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

Stepping back

To add a new command

- add a class with
  - a command name
  - new activation condition

Mechanics:

- Automatically registered entities
- Dynamic discovery of services
- Methods dynamically resolved

In Java

- Static methods are not looked up dynamically.
- Need to implement
  - a registration mechanism with one instance of each class
  - a singleton

Conclusion

- Class methods are dynamically resolved
- Classes are used as a dynamic registration system
- Adding a new class adds a new functionality modularly
- Can be incrementally defined (in different packages)
- Classes are objects and can be sent messages

Resources

- Pharo mooc - Videos W4S06: `http://mooc.pharo.org`
- Pharo by Example: `http://books.pharo.org`

A course by Stéphane Ducasse

http://stephane.ducasse.free.fr

Reusing some parts of the Pharo Mooc by

Damien Cassou, Stéphane Ducasse, Luc Fabresse

http://mooc.pharo.org