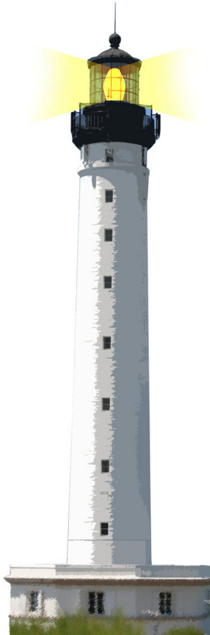
Advanced Object-Oriented Design

About Registration

When dynamic class registration is too much

S. Ducasse



http://www.pharo.org

Goal

- Thinking about system dynamics
- Alternatives to class methods as registration mechanism
- Impact of dynamic registration

Using class methods as registration

- Class is a real object
- We can send a message to a class
- Each class can answer specifically

```
Object allSubclasses collect: [ :each | each foo ]
```

Remember the Pharo Mooc

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

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

Registration for 'Free'

Pros:

- Each time a new class is loaded it is taken into account

Cons:

- We do it **all the time for nothing**
- We are querying the system for nothing!
- It is expensive

Solution 1: Explicit static list

```
PillarParser >> documentClasses
  ^ { Section . List . Paragraph }
    sorted: [ :class1 :class2 | class1 priority < class2 priority ]
```

Statically sorting the list

In fact we could precompute priority too

```
PillarParser >> documentClasses
  ^ { Section . Paragraph . List }
```

Pros:

- Do not have to query all the classes all the time

Cons:

- Watch out because we may not want to list explicitly class to avoid dependencies to other packages

Solution 2: Explicit registration mechanism

E.g., classes can explicit register to the parser

```
Section class >> initialize
  PillarParser registerClass: self
```

```
List class >> initialize
  PillarParser registerClass: self
```

```
PillarParser >> documentClasses
  ^ RegisteredClasses
```

A registration mechanism supports extension

```
Extra class >> initialize
  PillarParser registerClass: self
```

- External classes can also register
- Without introducing unwanted dependency
- Without scanning all the classes of the system

Unregistration

With explicit registration, the unregistration can be also a concern.

- The registration holder (here PillarParser) should offer way to cancel a registration
- Registered classes have the responsibility to unregister themselves.

Conclusion

- XXX subclasses is a cool pattern
- But it has a cost!
- Design is about tradeoffs

A course by

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