



# Conception orientée à objets

Stéphane Ducasse

[Stephane.Ducasse@univ-savoie.fr](mailto:Stephane.Ducasse@univ-savoie.fr)

<http://www.iam.unibe.ch/~ducasse/>

# Conception...

- Difficile
  - car pleins de pours et de contres
  - basé sur l'expérience
  - rien de tout blanc ou tout noir
  
- Mais quelques grandes lignes...
- Une opportunité pour réfléchir et prendre de la distance



# Objectifs

- **Concepts objets**
  - Instanciation
  - Heritage
  - Lookup self/super
- **Reflections sur la conception**
  - Methodes = unités de réutilisation
  - Principes
  - Nommage
  - Law of Demeter
  - Principe d'hollywood
  - Design patterns

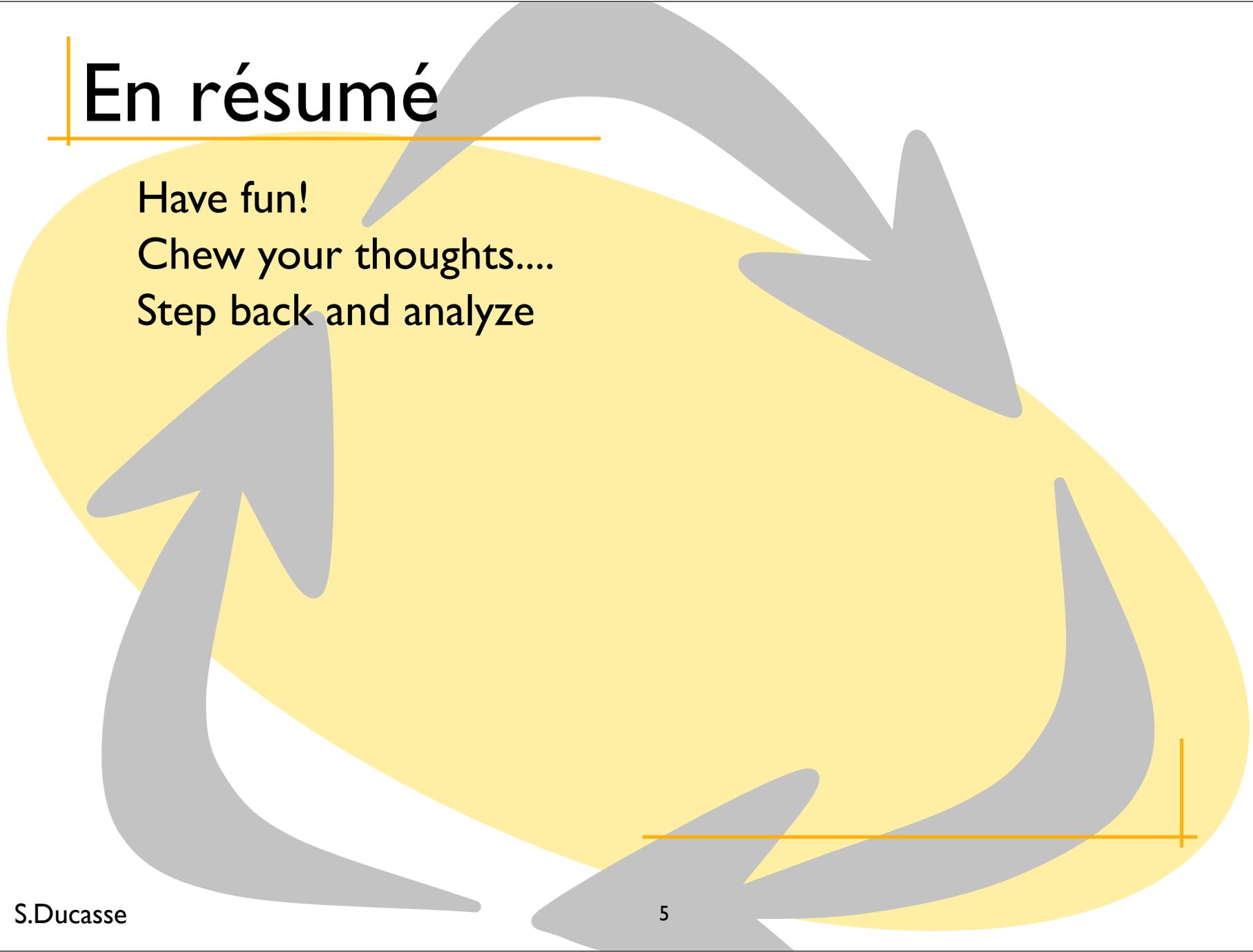


# Smalltalk

- Simple mais pas débile
- Utilisé dans la modélisation de systèmes **complexes**
  - Banques
  - UPS tracking system
  - AMD weaving chain
  - Automation system (frites)
- Smalltalk
  - pure, simple, puissant
  - Java en plus élégant, plus simple et plus puissant
  - inventeur a reçu deux prix Nobel en 2004



# En résumé



Have fun!  
Chew your thoughts....  
Step back and analyze

# Bibliographie

***Smalltalk by Example***, Sharp

***Object-Oriented Design Heuristics***, Riel

***Smalltalk Best Design Patterns***, Beck

***Smalltalk Design Pattern Companion***, Brown,  
Alpert, Woolf

***Refactoring: Improving the Design of Existing  
Code*** Fowler, Beck, Brant, Roberts

***Refactoring to Patterns***, Keriesky

# Questions?

## **How do static and dynamic types interact?**

```
class A {  
    void m(A a) { println("A.m(A)"); }  
}  
class B extends A {  
    void m(B b) { println("B.m(B)"); }  
}
```

```
B b = new B(); A a = b;
```

## *What are the results of the invocations? and why?*

```
a.m(a);  
a.m(b);  
b.m(a);  
b.m(b);
```

