Pharo : Modèle d'Objet

Réalisé par :
Edouard Guegain
Cheikh A. Bamba Lô
Sasandy Andrianasolo T.

Plan

Introduction

I. Interactions entre les objets
   a. Tout est objet
   b. Attributs privés, Méthodes publiques
   c. Messages
II. Comportement des classes
   a. Héritage
   b. Recherche de méthode
   c. Les classes sont des objets

Conclusion

Introduction

Règle 1. Tout est objet

Règle 2. Tout objet est instance d’une classe.

Règle 3. Toute classe a une super-classe.

Règle 4. Tout se passe par envoi de messages.

Règle 5. La recherche des méthodes suit la chaîne d’héritage.

I. Interactions entre les objets

Tout est Objet

Règle 1 : Tout est objet

- entiers, booléen, caractères, tableaux, ...
- classes et messages

Les objets reçoivent des messages.

Pas de Type de base ➜ pas de Wrapper

Tout est Objet

3 types d’objets:

- objet ordinaire avec les variables d’intances passés par réference
- objet avec les plus petits entiers passés en valeur
- objet indexés comme les tableaux

3 + 4
>>> 7

20 factorial
>>> 2432902008176640000

Attributs privés, Méthodes publiques

Méthodes publiques :

- pas de mot clé : ~~public~~, ~~private~~
- virtuellement liées: elles sont recherchées dynamiquement

Méthodes regroupées en protocoles

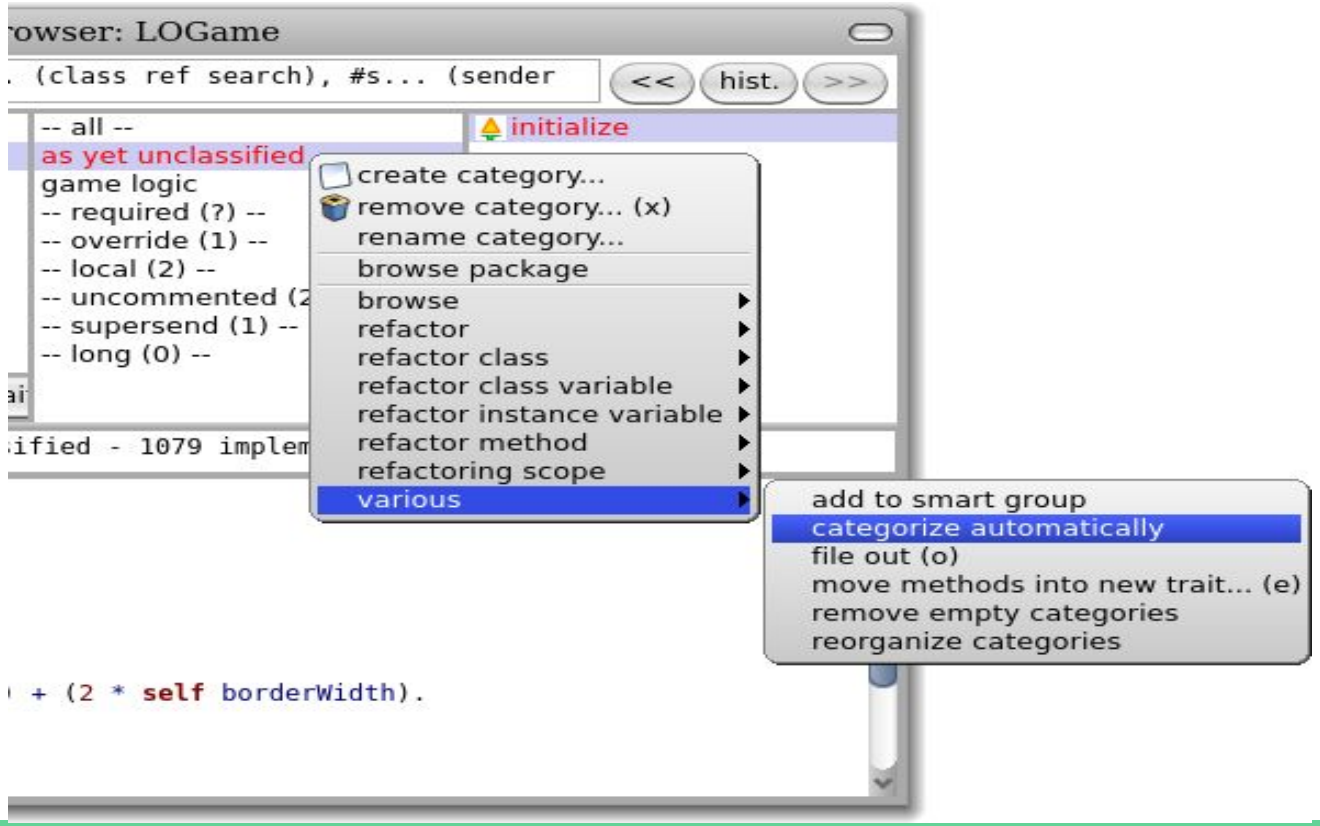
- indique l'intention
- convention de nommage (“accessing”, “initialization”)
- un protocole "private"

Attributs privés, Méthodes publiques

Attributs privés :

- Tout objet à des attributs privés (variables d’instances)
- Protégés
- Accessible uniquement par:
  - L’ objet lui même
  - ses sous-classes
  - ses méthodes

Attributs privés, Méthodes publiques

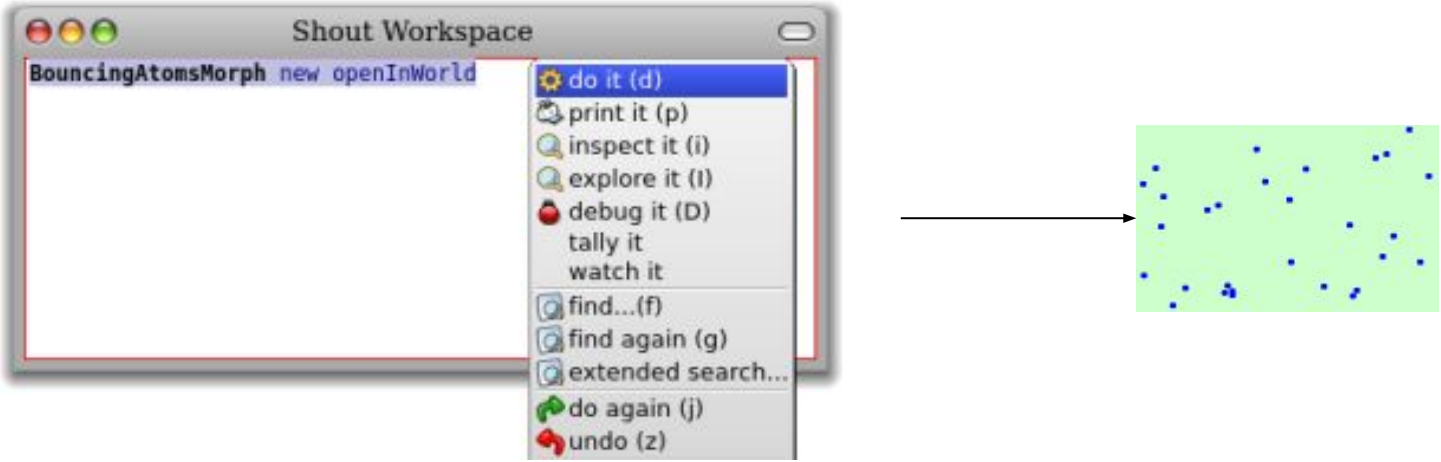


Messages

Tout se passe par envoi de messages

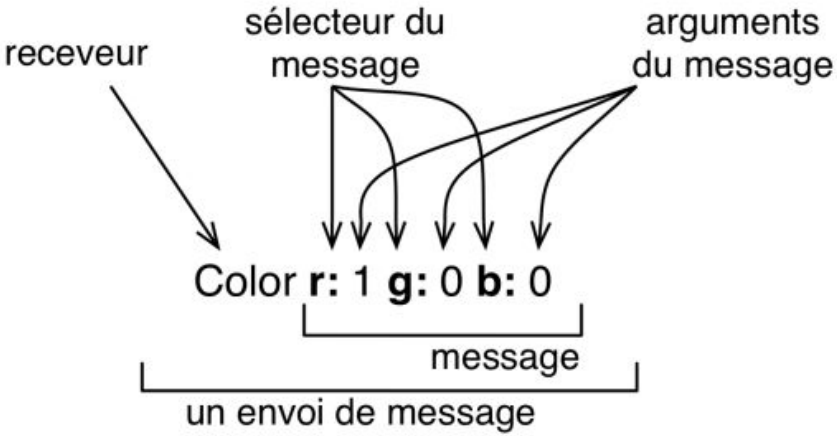
"envoyer un message" : "faire appel à une opération" ou "invoquer une méthode"

↔ demander à un objet de faire quelque chose



Messages

Syntaxe :



Messages

3 types de messages :

- Messages unaires : messages sans argument

```
1 factorial
```

- Messages binaires : un argument, formés avec des opérateurs

```
1 + 2
```

- Messages à mots-clés : un nombre arbitraire d’arguments, formés avec plusieurs mots-clés, chacun d’entre eux se finissant par deux points (:) et prenant un paramètre

```
30 between:10 and:100
```

Messages

Priorité des messages :

- messages unaires -> messages binaires -> messages à mots-clés

```
2 raisedTo: 1 + 3 factorial −→ 2 raisedTo: (1+(3 factorial)) −→ 128
```

- Évaluation de gauche à droite si de même type (changement d'ordre par utilisation des parenthèses)

```
1 + 2 * 3 −→ 9
1 + (2 * 3) −→ 7
```

- Utilisation de [] : fermeture lexicale ou bloc, lorsque vous ne savez pas combien de fois une expression peut être évaluée

```
[ x isReady ] whileTrue: [ y doSomething ]
```

Messages

Envoi des messages :

- Séquence: suite d’expressions séparées par des points, receveur respective

```
Transcript cr.
Transcript show: 'Bonjour le monde'.
Transcript cr
```

- Cascade : receveur, spécifié une seule fois et la suite des messages, séparée par des points virgules

```
Transcript cr;
          show: 'Bonjour le monde';
          cr
```

II. Comportement des classes

Héritage

Règle 3. Chaque classe a une superclasse.

SmallInteger superclass	⟶	Integer
Integer superclass	⟶	Number
Number superclass	⟶	Magnitude
Magnitude superclass	⟶	Object
Object superclass	⟶	ProtoObject

=> Héritage simple

(pas d'héritage multiple)

* Utilisation des traits (collection de méthodes)

```
Trait named: #TExample
     uses: { }
     category: 'PBE-LightsOut'

TExample»MethodeExample
     "Returns MethodeExample result"
     ↑'test'

SuperClassExample subclass: #ClassExample
     uses: TExample

ClassExample new MethodeExample -→ 'test'
```

Recherche de Méthode

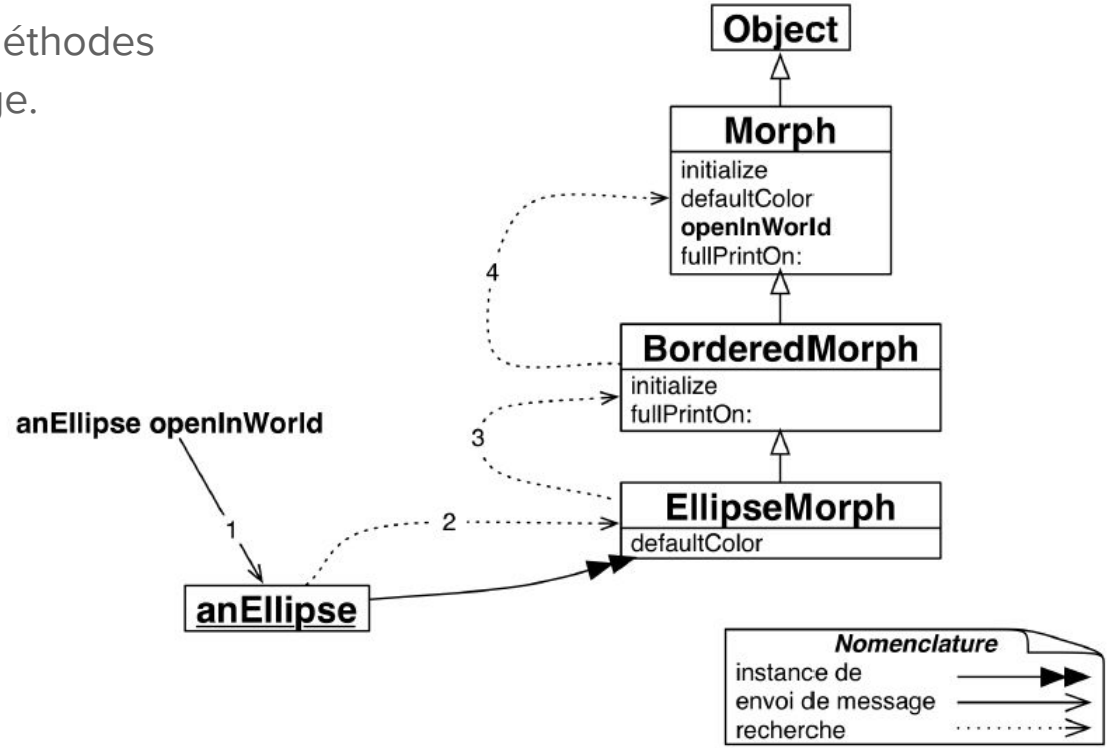
Règle 5. La recherche des méthodes suit la chaîne d'héritage.

Surcharge de méthode

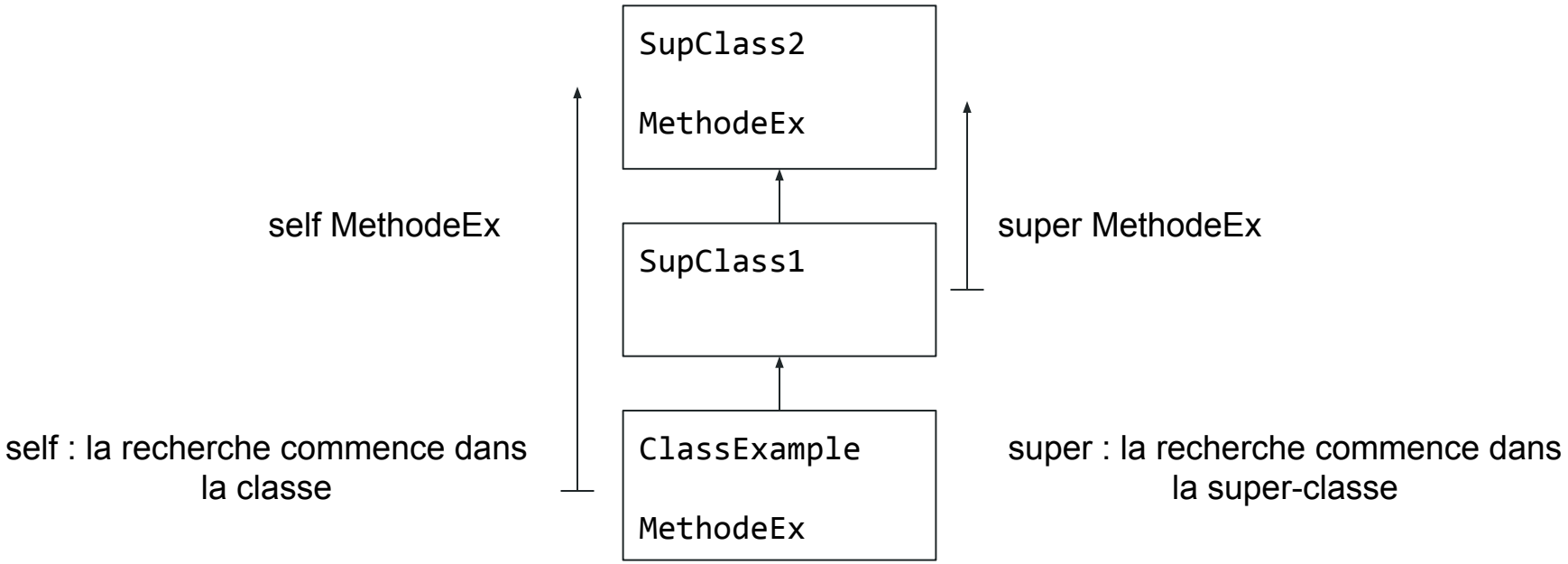
```
EllipseMorph»initialize
     self defaultColor
```

Extension avec super

```
EllipseMorph»initialize
     super initialize.
     self defaultColor
```



Recherche de Méthode



Les classes sont des objets

Règle 6. Toute classe est l’instance d’une méta-classe.

Règle 7. La hiérarchie des méta-classes est parallèle à celle des classes.

Règle 8. Toute méta-classe hérite de Class et de Behavior.

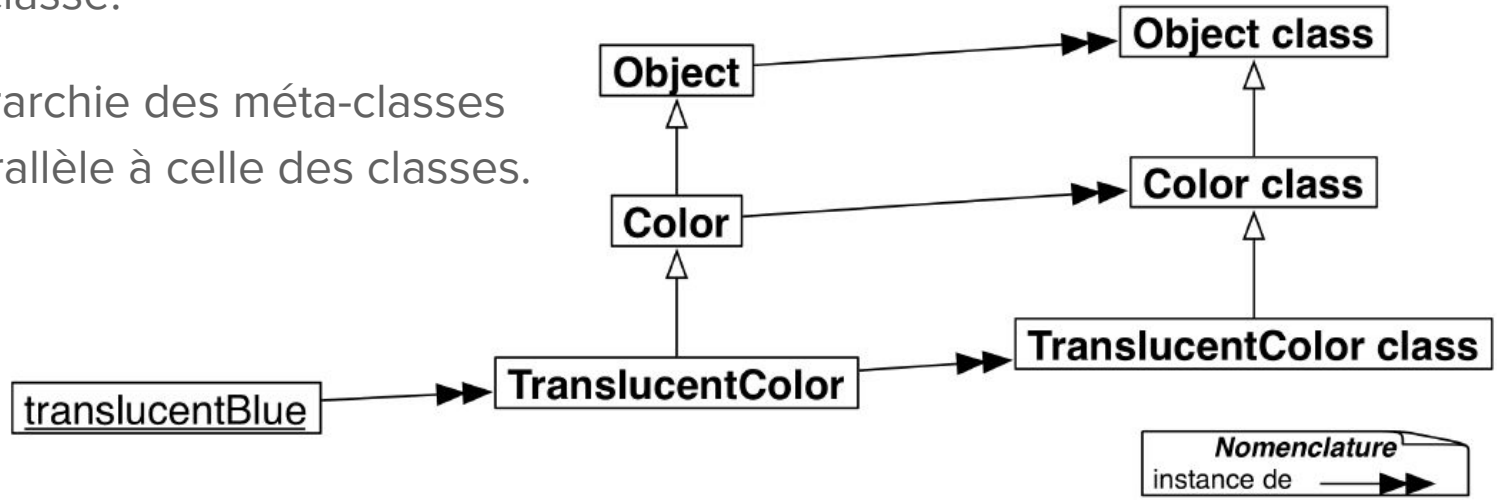
Règle 9. Toute méta-classe est une instance de Metaclass.

Règle 10. La méta-classe de Metaclass est une instance de Metaclass.

Les classes sont des objets

Toute classe est l'instance d'une méta-classe.

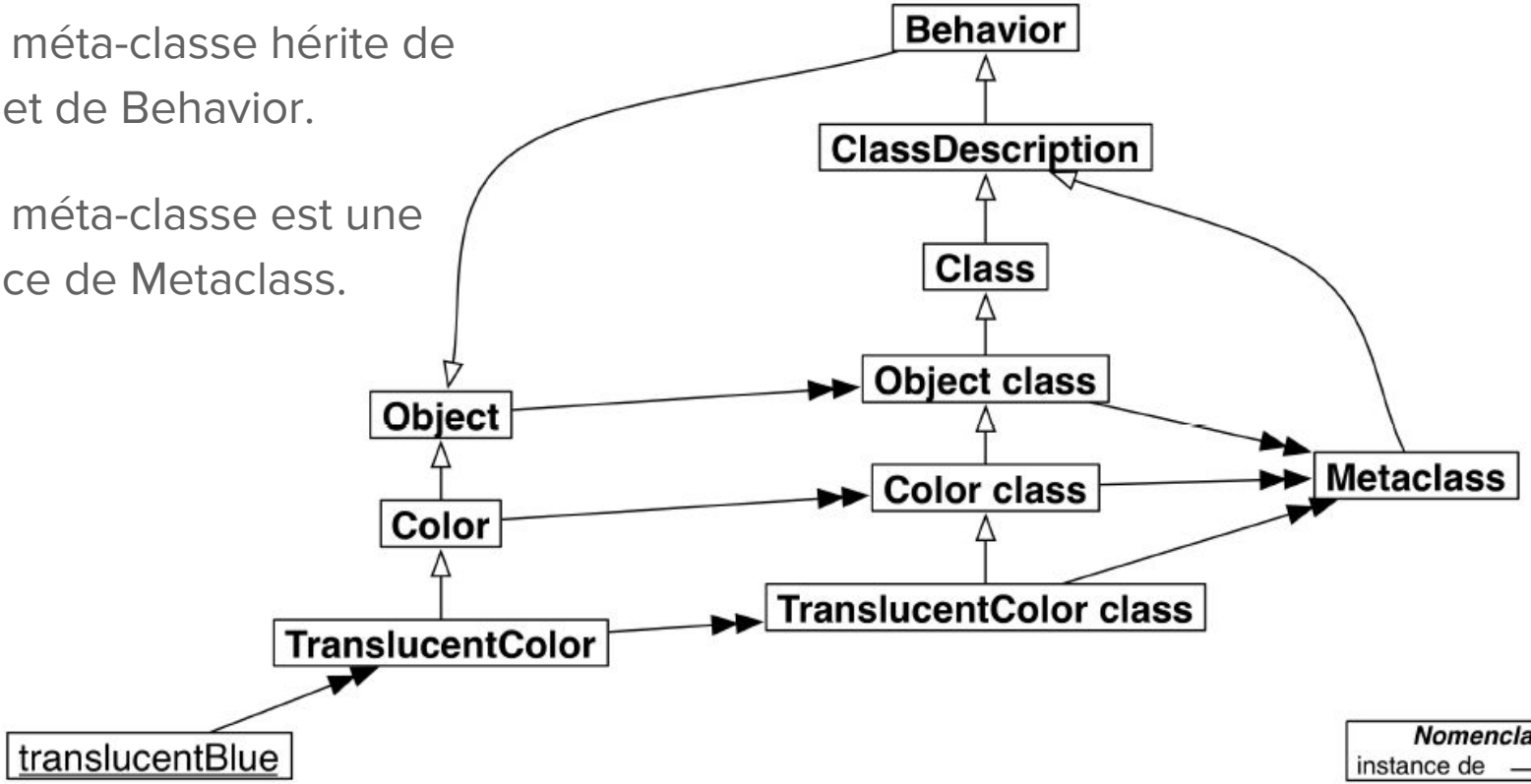
La hiérarchie des méta-classes est parallèle à celle des classes.



Les classes sont des objets

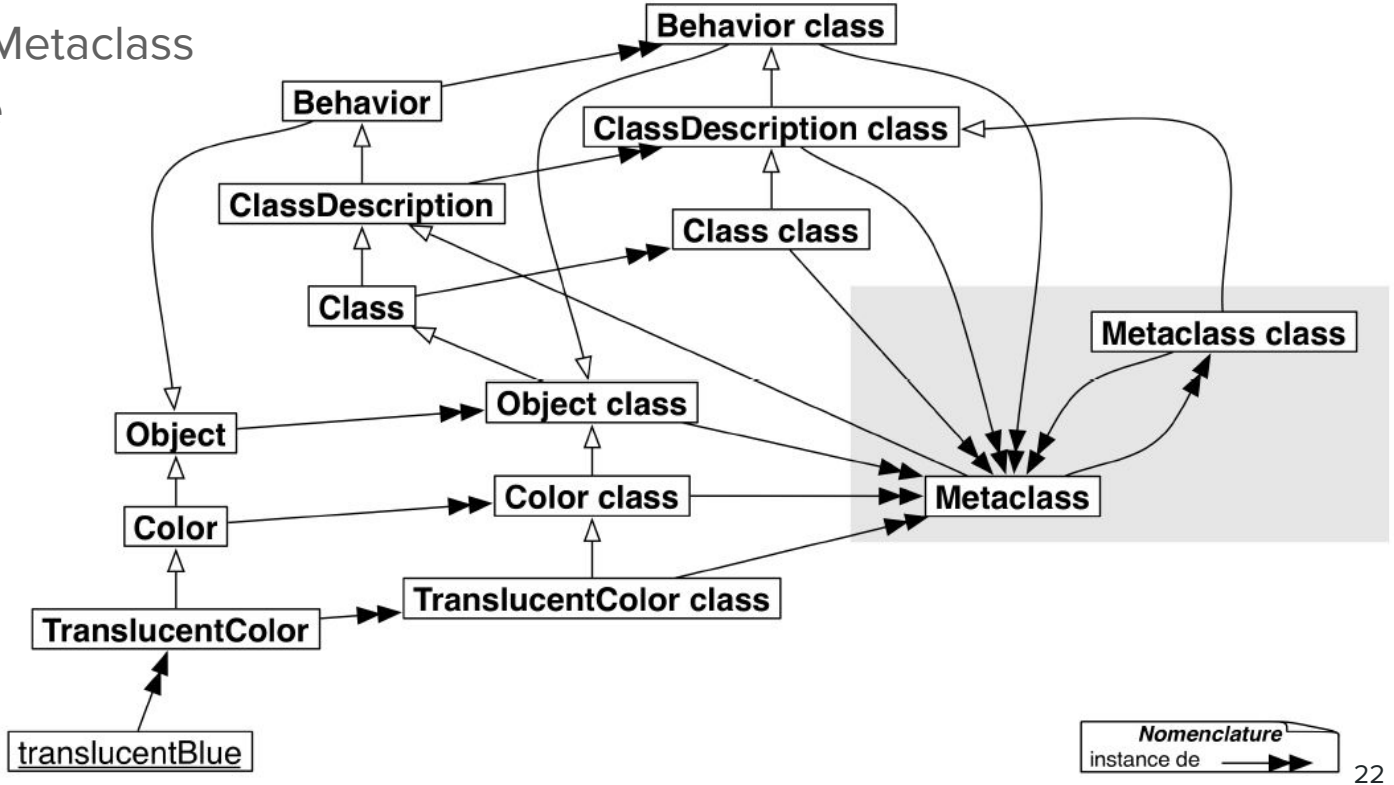
Toute méta-classe hérite de Class et de Behavior.

Toute méta-classe est une instance de Metaclass.



Les classes sont des objets

La méta-classe de Metaclass est une instance de Metaclass.



Conclusion

Conclusion

- Règles simples
- Règles uniformes

Sources :

Andrew BLACK, Stéphane DUCASSE, Oscar NIERSTRASZ, Damien POLLET, *Pharo par l'Exemple*, 2011

The Pharo Object Model, inria.fr