Programmation Orientée Objet : le polymorphisme

1. Redéclaration de méthodes

a) **Surcharge** = même nom, liste de paramètres différente

- dans la **même classe** ou dans les **classes dérivées** ex : courir, jouer
- activation **en fonction des paramètres** (recherche ascendante dans la hiérarchie de classes)
- les différentes versions **coexistent**

b) **Redéfinition =** même nom, même liste de paramètres, même type de retour

- uniquement dans les **classes dérivées** avec un niveau de visibilité supérieur ou égal ex : afficher, manger
- activation **en fonction de la classe de l'objet appelant** (recherche ascendante dans la hiérarchie de classes)
- la méthode héritée est « **cachée** » (ou « masquée »)

2. Notion de variables génériques

a) Propriété

Si C' est une classe dérivée de C alors **tout objet de C' est aussi un objet de C**

Attention : la réciproque est fausse

Ex : laLionne, leLion, leCanari sont des animaux
tout animal n'est pas un mammifère
unAnimal =leLion ; ok
leLion = unAnimal ; pas ok
leLion.courir(laLionne) ; ok
tabAnimaux[0]=leLion ; ok

b) Limites

Problème n°1 : une variable générique (polymorphisme)
une méthode redéfinie dans la classe dérivée

⇨ Quelle est la méthode activée ? Animal ou Mammifère ?

Problème n'2 : une variable générique (polymorphisme)
une méthode déclarée uniquement dans la classe dérivée

⇨ Comment appeler la méthode ?

c) Liaison de méthodes

- liaison statique = activation de la méthode correspondant au type **déclaré** de l'objet **(compilation)**
- liaison dynamique = activation de la méthode correspondant au type **réel_**de l'objet **(exécution)**

en Java = **toujours liaison dynamique**

d) Transtypage explicite

```
((Mammifere) unAnimal).crier ( ) ;
```

Règles d'application :

- unAnimal doit être de type réel Mammifere ou classe dérivée
- crier doit être déclaré dans Mammifere