

Mission 9 : Matière à apprendre

La programmation orientée objets :
masquage d'information, variables et méthodes de classe, héritage

Objects

- 1 - Classes and objects - Basics
- 2 - Classes and objects - Advanced
- 3 - Even more object-oriented programming
- 4 - Collections of objects
- 5 - Inheritance
- 6 - Linked lists
- Appendix - Source code of phone class
- Appendix - Source code of card game
- Appendix - Source code of linked lists
- Appendix - Worked out example: accounts

masquage d'information

attributs privés

méthodes accesseurs

variables de classe

méthodes de classe

héritage

self

super()



Exemple: un compte en banque

```
class Compte :  
  
    def __init__(self, titulaire):  
        self.titulaire = titulaire  
        self.solde = 0
```

Compte
titulaire
solde

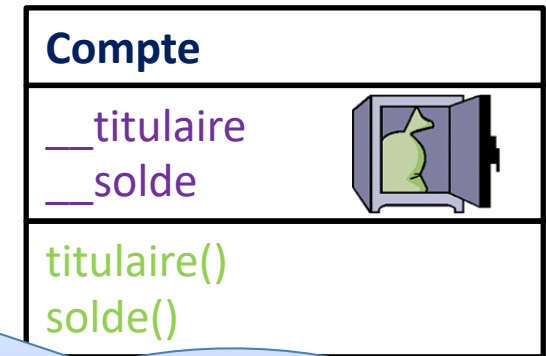
```
>>> a = Compte('kim')  
>>> a.titulaire  
'kim'  
>>> a.solde  
0  
>>> a.solde += 1000  
>>> b.solde -= 1000
```

comment protéger ?



Masquage d'information

```
class Compte :  
    def __init__(self, titulaire) :  
        self.__titulaire = titulaire  
        self.__solde = 0  
  
    def titulaire(self):  
        return self.__titulaire  
  
    def solde(self):  
        return self.__solde
```



attributs
privés

méthodes
accesseurs

```
>>> a = Compte('kim')  
>>> a.__titulaire  
AttributeError: 'Compte' object has no attribute '__titulaire'  
>>> a.__solde  
AttributeError: 'Compte' object has no attribute '__solde'  
>>> a.titulaire()  
'kim'  
>>> a.solde()  
0
```

Méthode `__str__`

```
class Compte :
    def __init__(self, titulaire) :
        self.__titulaire = titulaire
        self.__solde = 0

    def titulaire(self):
        return self.__titulaire
    def solde(self):
        return self.__solde
```

MIEUX :
Utiliser les
accesseurs

```
def __str__(self) :
    return "Compte de " + self.__titulaire \
        + " : solde = " + str(self.__solde)
```

méthode
`__str__`


```
>>> print(c)
Compte
```

```
def __str__(self) :
    return "Compte de " + self.titulaire() \
        + " : solde = " + str(self.solde())
```

Méthodes mutateurs

```
class Compte :  
  
    ...  
  
    def deposer(self, somme):  
        self.__solde += somme  
        return self.solde()  
  
    def retirer(self, somme):  
        if self.solde() >= somme :  
            self.__solde -= somme  
            return self.solde()  
        else :  
            return "Solde insuffisant"
```

```
>>> a = Compte('kim')  
>>> a.deposer(100)  
100  
>>> a.retirer(90)  
10  
>>> a.retirer(50)  
Solde insuffisant
```

Compte	
__titulaire	
__solde	
titulaire() solde() deposer(somme) retirer(somme)	

Variable de classe

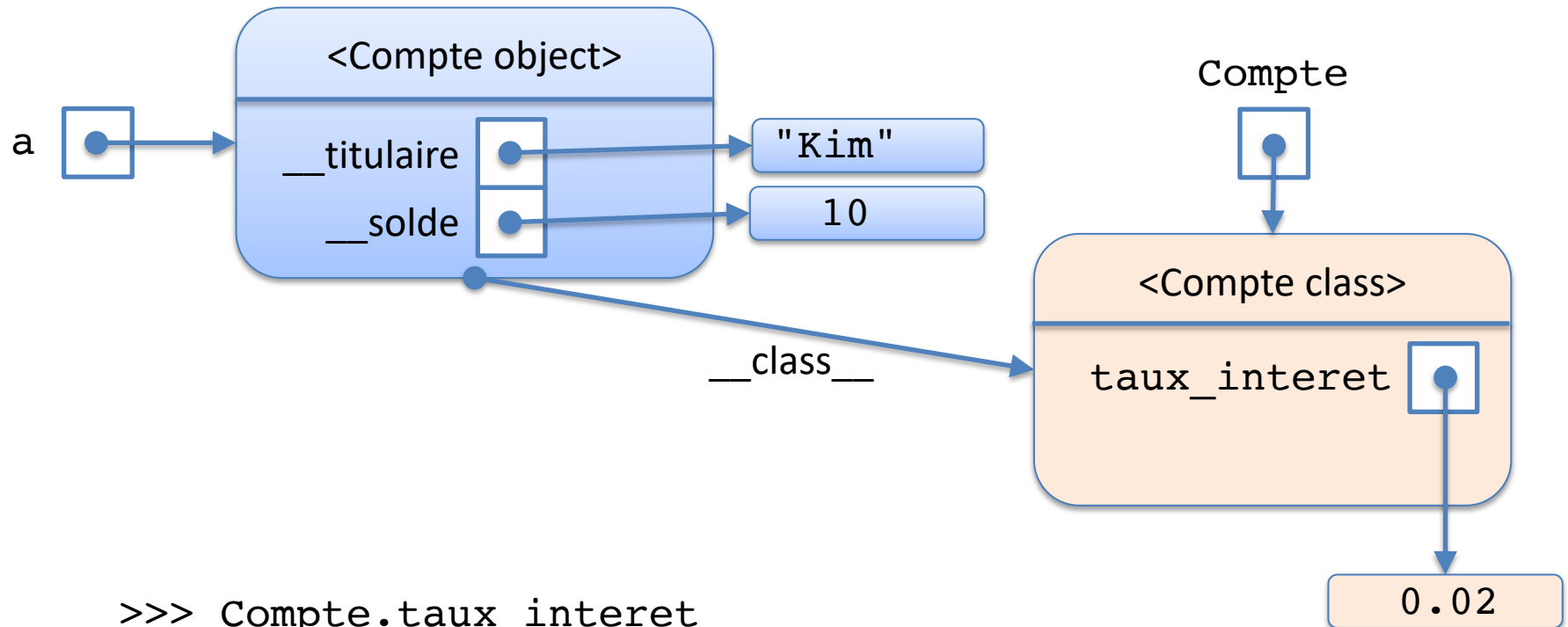
```
class Compte :  
    taux_interet = 0.02  
    ..
```

une variable
pour la classe

```
>>> Compte.taux_interet  
0.02  
>>> a = Compte("Kim")  
>>> a.taux_interet  
0.02  
>>> b = Compte("Siegfried")  
>>> b.taux_interet  
0.02  
>>> Compte.taux_interet = 0.04  
>>> a.taux_interet  
0.04  
>>> b.taux_interet  
0.04
```

valeur partagée par **toutes**
les instances de la classe

Variable de classe

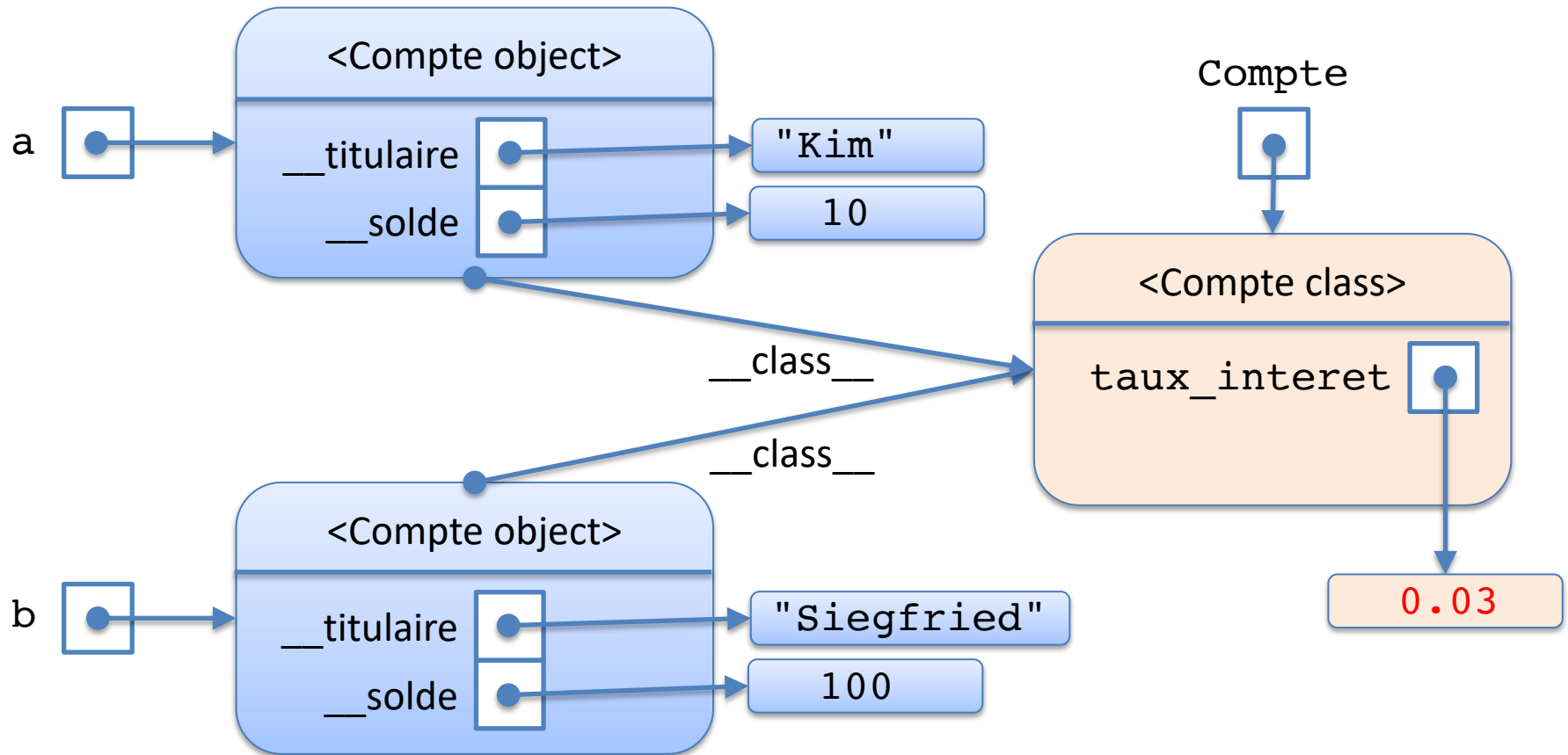


```
>>> Compte.taux_interet  
0.02
```

```
>>> a.taux_interet  
0.02
```

Si pas trouvé dans
l'objet, Python va chercher
dans la classe...

Variable de classe



```
>>> Compte.taux_interet = 0.03
```

```
>>> a.taux_interet
```

```
0.03
```

```
>>> b.taux_interet
```

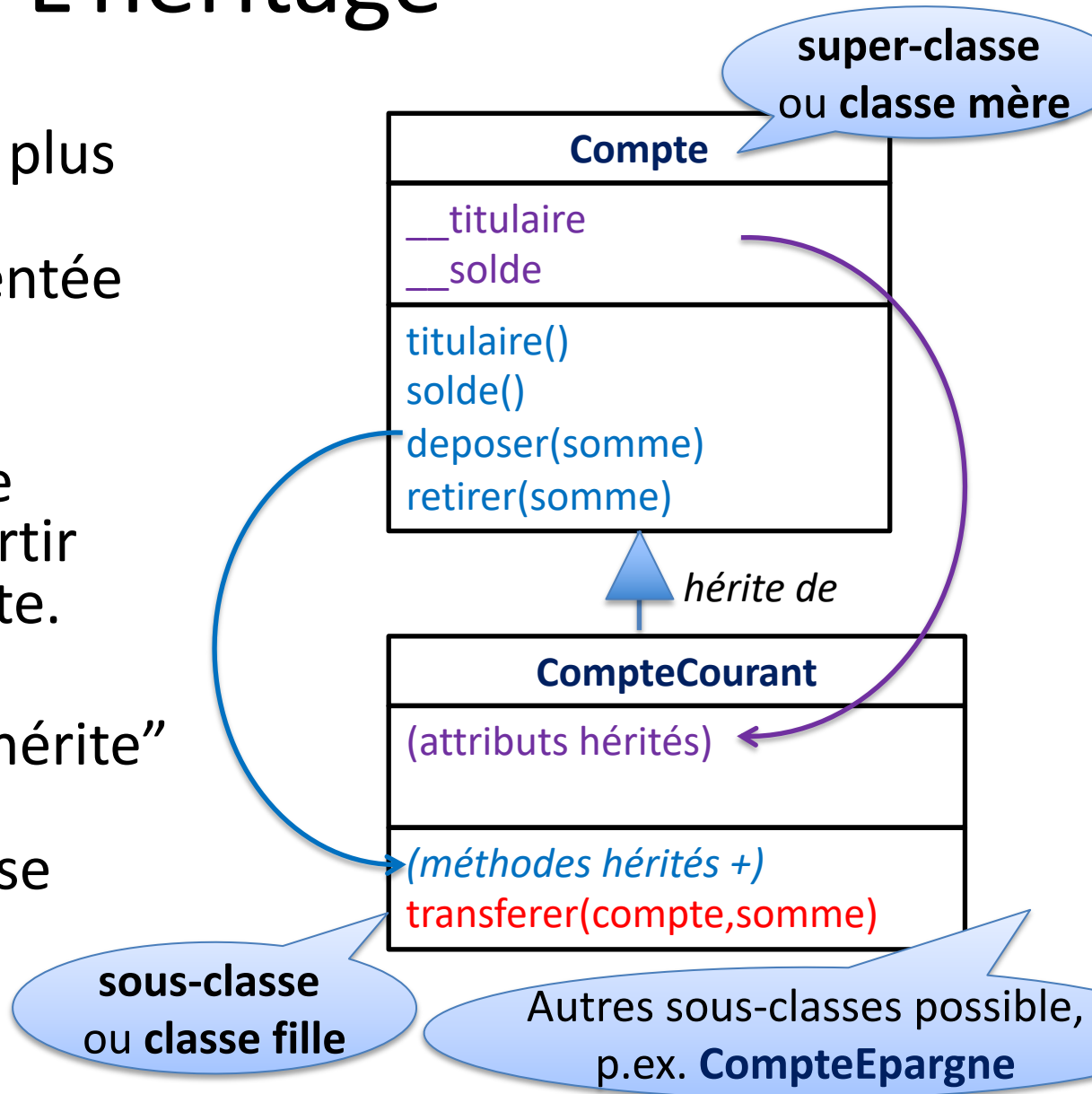
```
0.03
```


L'héritage

Un des concepts les plus importants de la programmation orientée objet.

Permet de créer une nouvelle classe à partir d'une classe existante.

La nouvelle classe "hérite" tous les attributs et méthodes de la classe existante.



Héritage

sous-classe

```
class CompteCourant(Compte) :  
  
    def transferer(self, compte, somme) :  
        res = self.retirer(somme)  
        if res != "Solde insuffisant"  
            compte.deposer(somme)  
        return res
```

classe mère

méthode
ajoutée

appel à une
méthode héritée

appel à une méthode
sur un autre objet

```
compte_kim = CompteCourant("Kim")  
compte_charles = CompteCourant("Charles")  
compte_kim.deposer(100)  
compte_kim.transferer(compte_charles, 50)
```

```
compte_kim.solde() → 50
```

```
compte_charles.solde() → 50
```

```
compte_kim.transferer(compte_charles, 60)
```

```
→ Solde insuffisant
```

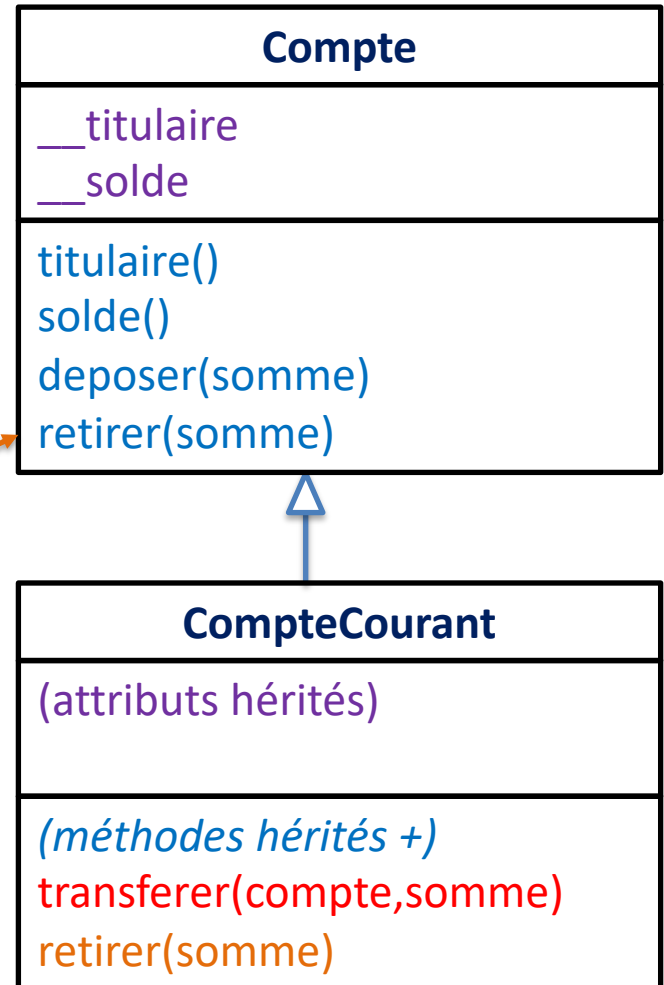
Redéfinition

Avec l'héritage on peut aussi **redéfinir** les attributs ou méthodes d'une super-classe

```
class Comptecourant(Compte) :
    __frais_retirer = 0.10
    def transferer(self,...) :
        ...
    def retirer(self, somme):
        frais = self.__frais_retirer
        return super().
            retirer(somme+frais)
```

L'héritage
c'est super!

```
compte_kim = Comptecourant("Kim")
compte_kim.deposer(1000)    → 1000
compte_kim.retirer(10)     → 989.9
compte_kim.retirer(10)     → 979.8
```



super()

```
class Comptecourant(Compte) :
    __frais_retirer = 0.10
    def __init__(self, titulaire, banque) :
        super().__init__(titulaire)
        self.__banque = banque
    def __str__(self) :
        return super().__str__() +
            "; banque = " + self.__banque
    ...
```

`super()` permet de référer à une (méthode dans la) classe mère sans devoir la nommer explicitement.

Souvent utilisé pour étendre une méthode de la super-classe, par exemple:

`retirer(somme)`, `__init__`, `__str__`

Description	prix HTVA	TVA	prix TVAC
Mission 9			
laptop 15" 8GB RAM	49.99	156.20	899.99
installation windows		13.88	79.99
installation wifi	45.22	9.50	54.72
carte graphique	119.49	25.09	144.58
1 * disque dur 350 GB @ 49.99	49.99	10.50	60.49
3 * souris bluetooth @ 15.99	47.97	10.07	58.04
Réparation (0.75 heures)	46.25	9.71	55.96
5 * adaptateur DVI - VGA @ 12.00	60.00	12.60	72.60
2 * Java in a Nutshell @ 24.00	48.00	2.88	50.88
5 * souris bluetooth @ 15.99	79.95	16.79	96.74
T O T A L	1306.77	267.22	1573.99

Objectifs :

Héritage

Problème :

Facturation & Livraison

Livraison - Facture No 1 : PC Store - 22 novembre

Description	poids/pce	nombre	poids
1 * disque dur 350 GB @ 49.99 (!)	0.355kg	1	1.000kg
3 * souris bluetooth @ 15.99	0.176kg	3	3.000kg
5 * adaptateur DVI - VGA @ 12.00	0.000kg	5	5.000kg
2 * Java in a Nutshell @ 24.00	0.321kg	2	2.000kg
5 * souris bluetooth @ 15.99	0.176kg	5	5.000kg
5 articles		16	2.405kg

(!) *** livraison fragile ***