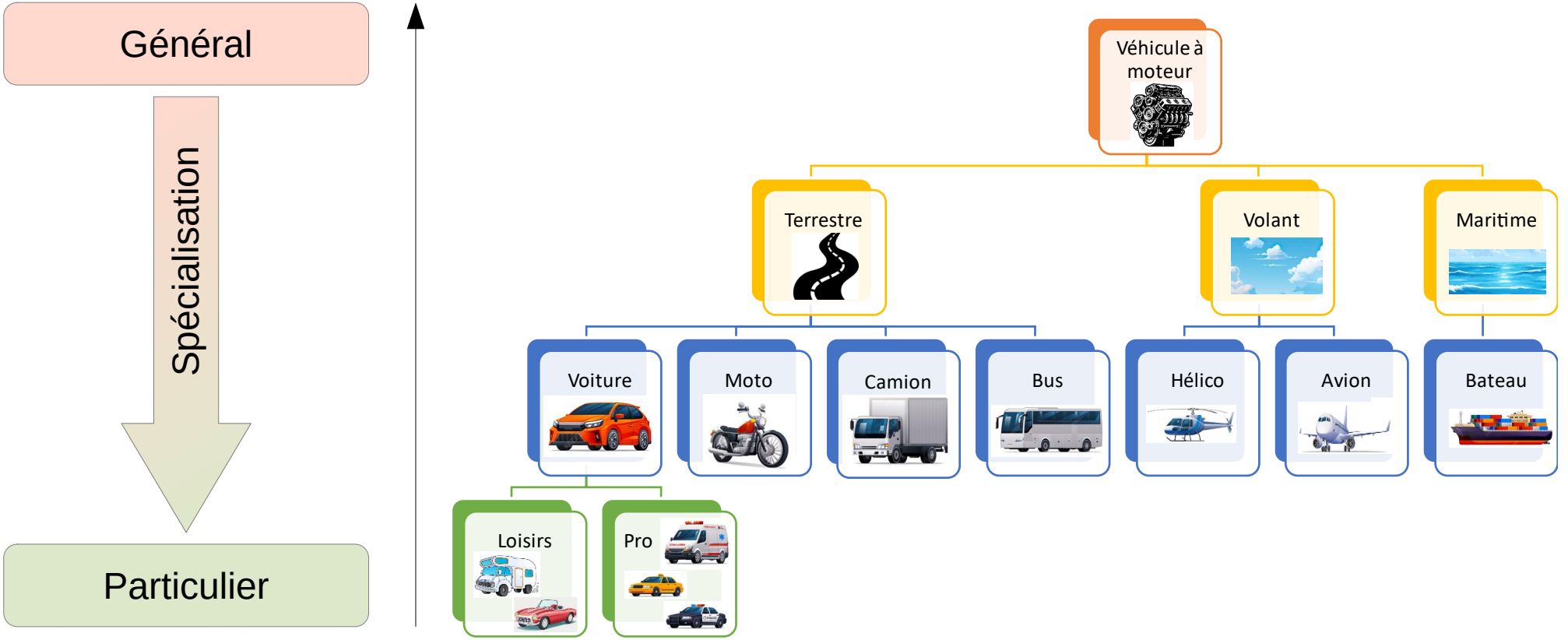
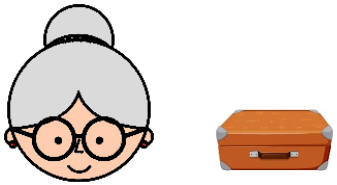
L'héritage

Qu'est-ce que l'héritage ?



Qu'est-ce que l'héritage ?



Classe mère

Classe fille

- La classe fille hérite de **tous les attributs** et **toutes les méthodes** de la classe mère.
- La classe fille peut **avoir des attributs et/ou des méthodes** qui lui sont **spécifiques**.
- La classe fille peut **spécialiser des méthodes** de la classe mère (=> polymorphisme d'héritage)

Dérivation de classe

```
class ClasseFille(ClasseMère):

    def __init__(self, ...):

        ClasseMère.__init__(self, ...)
        self.attribut_ = ...
```

Définition d'une **sous-classe** (ou classe héritée / dérivée)

Méthode **constructeur** de la classe mère

Dans la méthode constructeur de la classe fille, **il est obligatoire d'appeler la méthode constructeur de la classe mère**. Cela permet la création et l'accès à l'ensemble des attributs et des méthodes de la classe mère.

Tous les **attributs supplémentaires** de la classe fille doivent être créés dans la **méthode constructeur** de la classe fille, sans modification de la classe mère.

Dans l'appel du constructeur de la classe mère, il ne faut pas oublier le paramètre **self**. L'information ClasseMère devant le point (.) indique que la méthode constructeur qu'il faut utiliser est bien celle de la classe mère.

Dérivation de classe - Exemple

On souhaite créer un catégorie particulière d'élèves qui se sont spécialisés dans un parcours. Leurs spécificités sont :

- Ils appartiennent tous à la promo "2A "
- Ils sont attachés à une option.
- Leurs notes sont associées à des coefficients et stockées sous la forme d'une liste de doublets (note, coefficient)

Créer une **classe** EleveParcours qui **hérite** de la classe Elève, avec sa méthode **constructeur**.

Dans le programme principal, faire une instanciation de la classe EleveParcours.

Dérivation de classe - Exemple

```
class EleveParcours(Eleve):

    def __init__(self, nom, option, lnotes = None):

        Eleve.__init__(self, nom, '2A', lnotes)
        self.option_ = option
```

```
# Programme Principal
jarjar = EleveParcours('Binks' , 'Commerce')
```

Polymorphisme d'héritage

Le **polymorphisme d'héritage** consiste à utiliser le même nom pour des méthodes qui ont le même usage dans une **sous-classe** et sa **classe mère**. Il s'agit alors d'une spécialisation de la méthode.

Il est possible d'appeler la méthode de la classe mère dans la méthode spécialisée de la classe fille :

```
class ClasseFille(ClasseMère):

    ...

    def methode(self, ...):

         ClasseMère.methode(self, ...)
        ...
```

Polymorphisme d'héritage - Exemple

Dans la classe EleveParcours :

- Spécialiser la méthode moyenne.
- Spécialiser l'affichage (surcharge de print) pour indiquer en plus l'option de l'élève.

Dans le programme principal :

- Créer une instance de la classe EleveParcours. Afficher l'instance, ajouter une note à l'élève puis réafficher l'instance.

Polymorphisme d'héritage - Exemple

```
class EleveParcours(Eleve):

    def __init__(self, nom, option, lnotes = None):

        Eleve.__init__(self, nom, '2A', lnotes)
        self.option_ = option

    def calculer_moyenne(self):

        if self.lnotes_ == []:
            self.moyenne_ = None

        else:
            somme = 0
            n = 0

            for note, coeff in self.lnotes_:
                somme += note * coeff
                n += coeff
            self.moyenne_ = somme / n

    def __repr__(self):

        txt = Eleve.__repr__(self)
        return txt + ' option : ' + self.option_
```

```
# Programme Principal
jarjar = EleveParcours('Binks' , 'Commerce')

print(jarjar)
jarjar.ajouter_note((14, 2))
print(jarjar)
```

Utilisation de la méthode ajouter_note de la classe mère, aucune modification n'est nécessaire Dans l'appel, par contre, la note doit être fournie avec son coefficient sous la forme d'un doublet.

Réutilisation de la méthode __repr__ de la classe mère. Ceci est préférable à un copié-collé.

Polymorphisme ad hoc et d'héritage

Appel d'une méthode faisant l'objet de polymorphisme (i.e. définie dans plusieurs classes).

`instance.methode(params, …)`

instance est un objet de la classe B

Si la méthode est définie dans la classe B, je l'utilise.

Les classe B et A sont indépendantes :
Polymorphisme ad hoc

instance est un objet de la classe A

Si la méthode est définie dans la classe A, je l'utilise.

La classe A hérite de la classe A0

La classe A hérite de A0 :
Polymorphisme d'héritage

Sinon, je cherche la méthode dans la classe A0, classe mère de A.

Héritage ou Utilisation ?

Une classe peut utiliser des objets d'une autre classe sans pour autant qu'il y ait une relation d'héritage !