

Support de cours

Cours:

Introduction à la programmation orientée objet (en C++)

Vidéo:

W16-03-heritmultvirtuelles-CPP-pt3

Concepts (extraits des sous-titres générés automatiquement) :

Classes directes. Héritage virtuel. Constructeur de la sous-classe. Sous-sous-classe ovovivipare. Appel explicite. Première chose. Constructeurs. Chose suivante. Classe virtuelle. Déclaration des constructeurs des sous-classes. Ordre de déclaration d'héritage. Classe animal. Super-super-classe. Constructeur d'ovipare. Forte contrainte.



[vers la recherche de séquences vidéo](#)

(dans Introduction à la programmation orientée objet (en C++).)



[vers la vidéo](#)

Center for Digital Education. Plus de matériel de soutien pédagogique ici :

<https://www.epfl.ch/education/educational-initiatives/cede/educational-technologies-gallery/boocs-en/>

Classes virtuelles

(Partie 3)

Introduction à la programmation orientée objet (en C++)

Jean-Cédric Chappelier, Jamila Sam et Vincent Lepetit

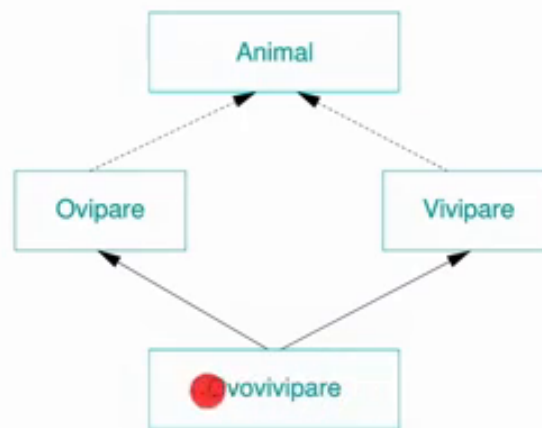
...

notes

résumé

0m 0s





Un seul objet de la super-classe **Animal** est hérité par l'héritage commun des sous-classes **Ovipare** et **Vivipare**.

de cette super-super classe virtuelle incombe à toute les sous-classes dont on veut créer une instance. Donc toutes ces sous-classes, sous-sous-classes, etc., qui héritent indirectement d'une super-super-super-classe virtuelle et qui ne sont pas abstraites, (c'est à dire que l'on peut donc en créer des instances), devront faire un appel **explicitite** au constructeur de la super-super-super-classe qui est virtuelle. Ca c'est donc une forte contrainte dans la déclaration des constructeurs des sous-classes Voyons ça de façon concrète sur l'exemple des ovovivipares.

notes

résumé

0m 37s



```
Ovovivipare::Ovovivipare(string nom, Habitat habitat, Regime regime,
                        unsigned int nb_oeufs,
                        unsigned int gestation,
                        bool rarete = false)
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{ }
```

Dans ce cas là, nous avons la sous-sous-classe Ovovivipare qui hérite de la sous-classe Ovipare et Vivipare lesquelles héritent virtuellement de la classe Animal. On a donc ici une classe virtuelle Animal et donc il incombe à la sous-classe Ovovivipare, dans ses constructeurs, d'appeler directement le constructeur d'Animal. On devra donc écrire la chose suivante dans le constructeur d'Ovovivipare, avoir un appel explicite au constructeur d'Animal on sera obligé d'écrire ceci. Supposons qu'il prenne ces trois paramètres, avant d'écrire l'appel au constructeur usuel des super-classes directes dont dépend Ovovivipare. Donc ça c'est une contrainte assez forte au niveau des classes virtuelles.

notes

résumé

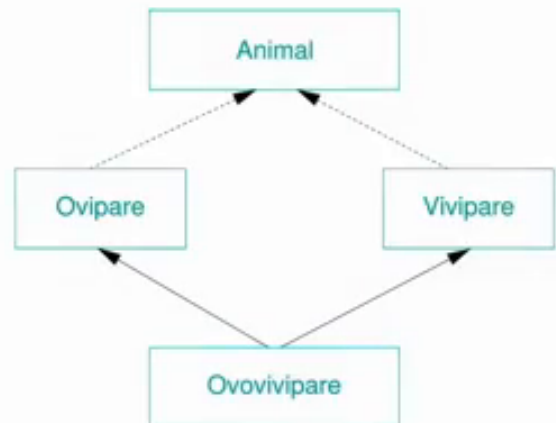
1m 13s



Comment sont gérés les appels au constructeur de la super-classe virtuelle ?

```
Ovovivipare::Ovovivipare(// ...
                        )
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{}
```

```
Ovovivipare o(...);
```



Ce qui se passe concrètement, lorsqu'on déclare donc une instance,

notes

résumé

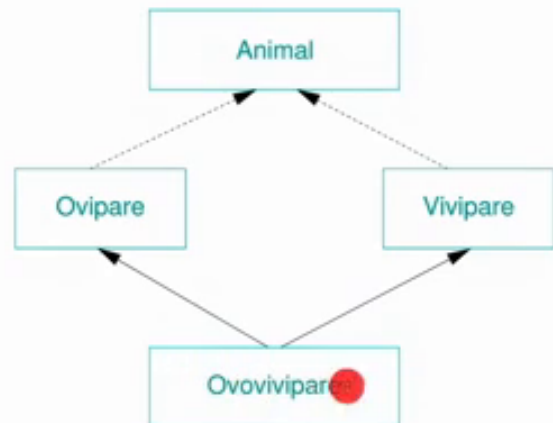
1m 49s



Comment sont gérés les appels au constructeur de la super-classe virtuelle ?

```
Ovovivipare::Ovovivipare(// ...
                        )
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{}
```

```
Ovovivipare o(...);
```



Comme ça, o de Ovovivipare donc appel au constructeur d'Ovovivipare, c'est que la première chose qui va se faire, c'est d'appeler la construction d'un Animal. Donc quand on construit, comme ça, une sous-sous-sous classe,

notes

résumé

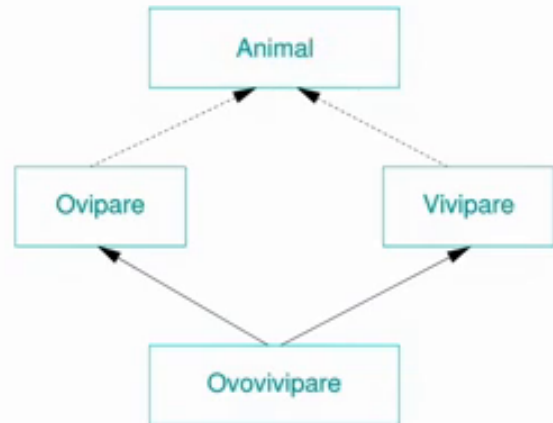
1m 52s



Comment sont gérés les appels au constructeur de la super-classe virtuelle ?

```
Ovovivipare::Ovovivipare(// ...
)
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{}
```

```
Ovovivipare o(...);
```



qui dépend d'une classe virtuelle, la première chose que l'on fait, c'est de créer la classe virtuelle. il n'y en a qu'une, justement, c'est le but de l'héritage virtuel et des classes virtuelle de n'en avoir qu'une et de ne pas en avoir plusieurs, au travers de tout les chemins d'héritage. Et donc comme en n'a qu'une, on est obligé de la construire en premier

notes

résumé

2m 7s

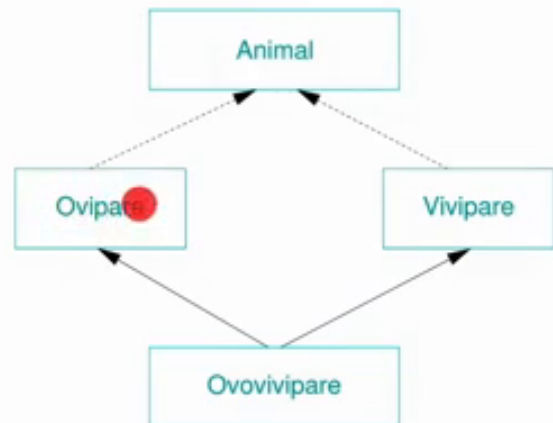


Classes virtuelles : appel des constructeurs

Comment sont gérés les appels au constructeur de la super-classe virtuelle ?

```
Ovovivipare::Ovovivipare(// ...
)
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{}
```

```
Ovovivipare o(...);
```



lequel constructeur d'Ovipare

notes

résumé

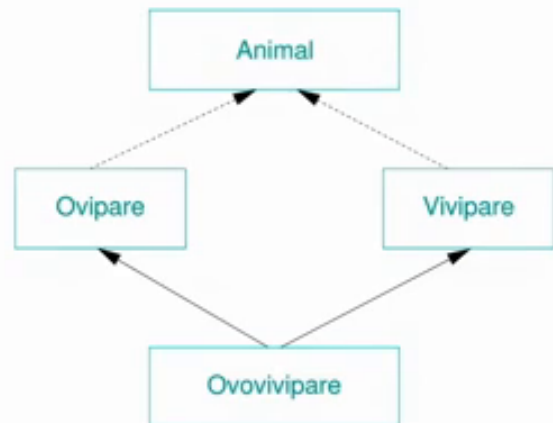
2m 37s



Comment sont gérés les appels au constructeur de la super-classe virtuelle ?

```
Ovovivipare::Ovovivipare(// ...
)
: Animal(nom, habitat, regime),
  Ovipare(nb_oeufs),
  Vivipare(gestation),
  espece_rare(rarete)
{}
```

Ovovivipare o(...);



normalement, devrait appeler le constructeur de la super-classe, Mais ici, comme on a déjà initialisé la super-classe, cet appel au constructeur d'Animal dans le constructeur d'Ovipare va être ignoré. De même ensuite on appellera le constructeur de Vivipare, lequel va ignorer l'appel du constructeur d'Animal puisque ça a déjà été fait au départ. Pour résumer, ici, construction d'un Ovovivipare
Première chose, on appelle le constructeur d'Animal ; deuxième chose, on appelle le constructeur d'Ovipare; dans lequel on n'appelle plus le constructeur d'Animal;
Troisième chose, on appelle le constructeur de Vivipare,

notes

résumé

2m 39s



Lors de la création d'un objet d'une classe plus dérivée, son constructeur invoque directement le constructeur de la super-classe virtuelle.

Les appels au constructeur de la super-classe virtuelle dans les classes *intermédiaires* **sont ignorés**.

Si la super-classe virtuelle a un *constructeur par défaut*, il n'est pas nécessaire de faire appel à ce constructeur explicitement. Il sera directement appelé par le constructeur de la classe dont on crée une instance.

S'il n'y a pas d'appel explicite au constructeur de la super-classe virtuelle et si celle-ci n'a pas de constructeur par défaut, *la compilation signalera une erreur*.

dans lequel on n'appelle plus le constructeur d'Animal et puis on termine bien sûr, comme d'habitude en initialisant ses propres attributs. Donc pour résumer, lors de la création d'un objet d'une classe la plus dérivée, sous-sous-sous classe d'une classe virtuelle, c'est son constructeur, de cette sous-sous-classe, qui a la charge d'invoquer directement le constructeur de la superclasse virtuelle. Et les appels au constructeur de la super-classe virtuelle dans les classes intermédiaires sont simplement ignorés. Bien sur, si la super-classe virtuelle a un constructeur par défaut alors il n'est évidemment pas nécessaire de faire explicitement appel à son constructeur, mais de toute façon,

notes

résumé

3m 13s



Dans une hiérarchie de classes où il existe des super-classes virtuelles :

- ▶ le soin d'initialiser les super-classes virtuelles incombe à la sous-classe la plus dérivée
- ▶ les constructeurs des super-classes virtuelles *sont invoqués en premier*
- ▶ les appels explicites au constructeur de la super-classe virtuelle dans les classes intermédiaires *sont ignorés*.
- ▶ ceux des classes non-virtuelles le sont ensuite *dans l'ordre de déclaration de l'héritage*
- ▶ l'ordre d'appel des *constructeurs de copie* est identique
- ▶ l'ordre d'appel des *destructeurs* est *l'inverse* de celui des appels de constructeurs

il y aura un appel à ce constructeur par défaut, qui sera appelé dès la création de l'instance la plus dérivée. Et comme d'habitude, bien sûr, si l'appel au constructeur de la super-super-classe virtuelle est omis dans les sous-sous-classes et que cette super-super-classe virtuelle n'a pas de constructeur par défaut, alors évidemment la compilation va signaler une erreur, puisqu'elle voudra absolument appeler un constructeur qui n'existe pas. Ainsi dans une hiérarchie de classe, où il existe des super-classes virtuelles, le soin d'initialiser ces super-classes virtuelles

notes

résumé

3m 49s





incombe a toute les sous-classes que l'on veut créer, les constructeurs des super-classes virtuelles sont invoqués en tout premier, et les appels à ces constructeurs de super-classes virtuelles dans les classes intermédiaires sont ignorés. Pour ce qui est des sous-classes non virtuelles, elles sont ensuite initialisées dans l'ordre de déclaration d'héritage, comme d'habitude. L'ordre d'appel des constructeurs de copies vérifie aussi ces règles là. Et comme toujours, l'ordre d'appel des destructeurs est l'inverse de celui de l'ordre d'appel des constructeurs. Voilà ce qui conclue cette séquence sur ce sujet un petit peu délicat des classes virtuelles et donc de l'héritage virtuel. et donc de l'héritage virtuel.

notes

résumé

4m 25s

