

INF1010

Complément sur les appels de constructeurs

Voici une autre méthode pour trouver la liste d'appels des constructeurs lors de la création d'un objet. Cette technique se veut un complément des notes de cours. Elle comporte deux étapes, soit de construire un arbre et ensuite de le vider.

Étape 1 : construction de l'arbre

Chaque nœud de l'arbre va correspondre à un appel de constructeur. Chaque arc correspond à une relation (héritage ou composition). Ainsi, on commence avec le premier appel (celui de l'objet créé). Ensuite, on ajoute le constructeur de la classe de base (s'il y a lieu) et les constructeurs des attributs objet (s'il y a lieu), en s'assurant que le constructeur de la classe de base soit au dessus. De plus, si on a accès aux listes d'initialisation, on en profite pour spécifier dans chaque nœud le bon constructeur. On répète cette étape pour chaque feuille (nœud qui termine l'arbre), jusqu'à ce que toutes les feuilles soient des constructeurs de classes simples (classes qui ne sont ni dérivées ni composées).

Par exemple, si on prend le diagramme illustré à la Figure 1 et que l'on instancie la classe D, on obtient l'arbre illustré dans la Table 1.

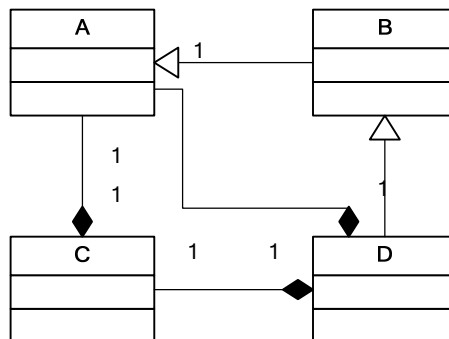
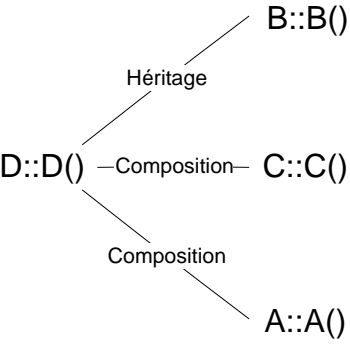
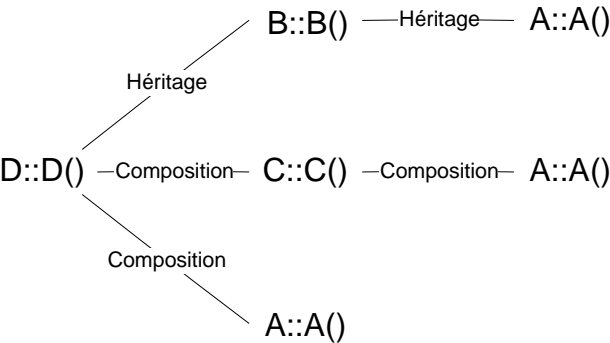


Figure 1: Exemple de diagramme de classe

Table 1: Étapes pour construire l'arbre

<p>D::D()</p>	<p>On débute l'arbre par l'appel du constructeur de l'objet instancié</p>
 <pre> graph LR DD[D::D()] -- Héritage --> BB[B::B()] DD -- Composition --> CC[C::C()] DD -- Composition --> AA[A::A()] </pre>	<p>On rajoute ensuite les appels du constructeur de la classe de base en premier ensuite les appels des constructeurs des attributs objet</p>
 <pre> graph LR DD[D::D()] -- Héritage --> BB[B::B()] DD -- Composition --> CC[C::C()] DD -- Composition --> AA1[A::A()] BB -- Héritage --> AA2[A::A()] CC -- Composition --> AA3[A::A()] </pre>	<p>On répète cette étape pour chaque feuille jusqu'à que toutes les feuilles soient des appels de constructeurs de classes simples. Ici, A est une classe simple, le travail est donc terminé.</p>

Étape 2 : Vider l'arbre

Une fois que l'arbre est construit, il faut le vider, dans un ordre spécifique, pour obtenir la liste des appels de constructeurs, dans le bon ordre. Pour s'y faire, on doit traiter les nœuds en se posant la question suivante :

Est-ce que le nœud possède au moins un fils?

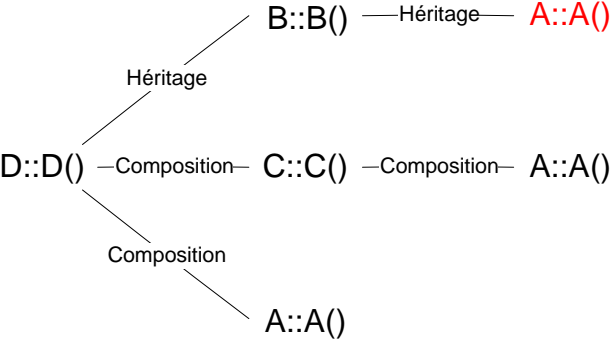
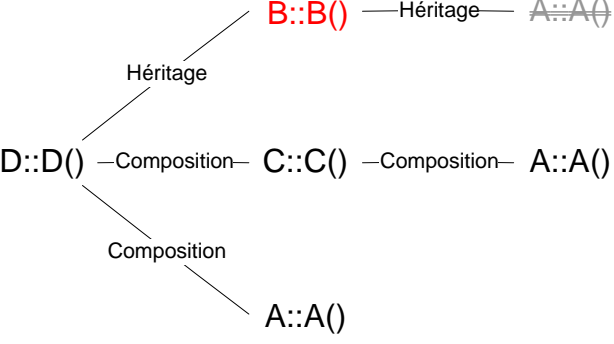
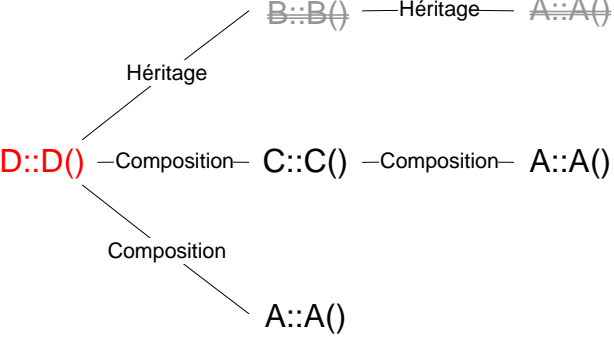
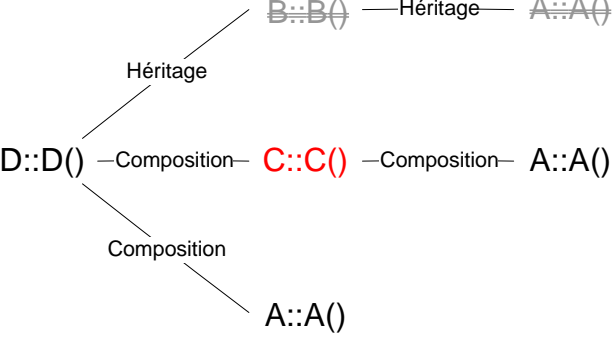
- Si la réponse est *oui*, on traite alors son fils le plus haut;
- Si la réponse est *non*, on retire alors le nœud, on l'ajoute à notre liste d'appels et on traite son parent.

On effectue cette opération tant et aussi longtemps que notre arbre n'est vide. On commence le traitement à la racine (ceci va nécessairement nous amener à retirer la feuille la plus haute en premier).

La Table 2 montre les différentes étapes pour retirer les nœuds de l'arbre trouvé à la Table 1.

Table 2: Étapes à effectuer pour retirer les nœuds

Arbre	Description	Liste
<pre> graph TD Root[D::D()] -- Héritage --- B[B::B()] Root -- Composition --- C[C::C()] Root -- Composition --- A[A::A()] B -- Héritage --- A C -- Composition --- A </pre>	On commence par la racine. Ce nœud possédant plusieurs fils, on passe à son fils le plus haut	(Vide)
<pre> graph TD Root[D::D()] -- Héritage --- B[B::B()] Root -- Composition --- A[A::A()] B -- Héritage --- A </pre>	Ce nœud possédant un seul fils, on passe à son fils	(Vide)

 <pre> graph TD DD["D::D()"] -- Héritage --> BB["B::B()"] DD -- Composition --> CC["C::C()"] DD -- Composition --> AA1["A::A()"] BB -- Héritage --> AA2["A::A()"] style AA2 fill:#f00 </pre>	<p>Ce nœud ne possédant pas de fils, on le retire et on passe à son parent</p> <p><i>remarque : le premier nœud retiré est toujours la feuille la plus haute</i></p>	<p>(Vide)</p>
 <pre> graph TD DD["D::D()"] -- Héritage --> BB["B::B()"] DD -- Composition --> CC["C::C()"] DD -- Composition --> AA1["A::A()"] BB -- Héritage --> AA2["A::A()"] style AA2 fill:#f00 </pre>	<p>Ce nœud ne possédant plus de fils (on a retiré son seul fils), on le retire et on passe à son parent</p>	<p>A::A()</p>
 <pre> graph TD DD["D::D()"] -- Héritage --> BB["B::B()"] DD -- Composition --> CC["C::C()"] DD -- Composition --> AA1["A::A()"] BB -- Héritage --> AA2["A::A()"] style AA2 fill:#f00 </pre>	<p>Ce nœud possédant encore des fils, on choisit le fils le plus haut</p>	<p>A::A() B::B()</p>
 <pre> graph TD DD["D::D()"] -- Héritage --> BB["B::B()"] DD -- Composition --> CC["C::C()"] DD -- Composition --> AA1["A::A()"] BB -- Héritage --> AA2["A::A()"] style AA2 fill:#f00 </pre>	<p>Ce nœud possédant un fils, on le choisit</p>	<p>A::A() B::B()</p>

<pre>graph TD; D["D::D()"] -- Héritage --> B["B::B()"]; D -- Composition --> C["C::C()"]; D -- Composition --> A["A::A()"]; B -- Héritage --> A2["A::A()"]; C -- Composition --> A3["A::A()"];</pre>	Ce nœud ne possédant pas de fils, on le retire et on passe à son parent	A::A() B::B()
<pre>graph TD; D["D::D()"] -- Héritage --> B["B::B()"]; D -- Composition --> C["C::C()"]; D -- Composition --> A["A::A()"]; B -- Héritage --> A2["A::A()"]; C -- Composition --> A3["A::A()"];</pre>	Ce nœud ne possédant plus de fils (on a retiré son seul fils), on le retire et on passe à son parent	A::A() B::B() A::A()
<pre>graph TD; D["D::D()"] -- Héritage --> B["B::B()"]; D -- Composition --> C["C::C()"]; D -- Composition --> A["A::A()"]; B -- Héritage --> A2["A::A()"]; C -- Composition --> A3["A::A()"];</pre>	Ce nœud possédant encore un fils, on le choisit pour le traiter.	A::A() B::B() A::A() C::C()
<pre>graph TD; D["D::D()"] -- Héritage --> B["B::B()"]; D -- Composition --> C["C::C()"]; D -- Composition --> A["A::A()"]; B -- Héritage --> A2["A::A()"]; C -- Composition --> A3["A::A()"];</pre>	Ce nœud ne possédant pas de fils, on le retire et on passe à son parent	A::A() B::B() A::A() C::C()

	<p>Ce nœud ne possédant plus de fils (on a retiré tous ses fils), on le retire</p>	<p>A::A() B::B() A::A() C::C() A::A()</p>
	<p>L'arbre étant maintenant vide, le travail est terminé</p>	<p>A::A() B::B() A::A() C::C() A::A() D::D()</p>

Résultat : La liste d'appels des constructeurs

La liste obtenue est alors :

A::A()
B::B()
A::A()
C::C()
A::A()
D::D()