Programmation Orientée Objet (C++) : Héritage

Jamila Sam

Laboratoire d'Intelligence Artificielle
Faculté I&C

EPFL

Vidéos, transparents et quiz

https ://www.coursera.org/learn/programmation-orientee-objet-cpp/home/week/4

☞ Semaine 4

EPFL

Progression

Objet	
Encapsulation et Abstraction Classes Héritage simple/multiple Polymorphisme Classes abstraites/virtuelles Résolution des collisions de noms	
Traitements	Données
Méthodes Constructeurs & Destructeurs Const Virtuelles (pures) Surcharge d'opérateurs(interne/externe)	Attributs Appels aux constructeurs des attributs (hérités) Statiques
Privés/protégés/publiques Hérités/cachés (: :)	

EPFL

Objectif de la leçon d'aujourd'hui

- rappel des concepts fondamentaux
- étude de cas : relation A-UN ou relation EST-UN
- complément non abordé par le MOOC

EPFL

Héritage

Spécifier un *lien d'héritage* :

```
class Sousclasse  : [public] SuperClass {...}
```

Droits d'accès : `protected` accès autorisé au sein de la hiérarchie

Masquage : un attribut/méthode peut être redéfini dans une sous-classe

Accès à un *membre caché* : `SuperClasse::membre`

Le constructeur d'une sous classe doit faire appel au *constructeur de la super classe* :

```
class SousClasse: SuperClasse
{
   SousClasse(liste de paramètres)
   : SuperClasse(Arguments),
    attribut1(valeur1), ..., attributN(valeurN) {...}
};
```


Concepts fondamentaux

- composition : **A-UN**

```
class Instrument {};

class Musicien {
private:
        // un musicien A-UN instrument
        Instrument instrument;
};
```

- héritage : **EST-UN**

```
class Instrument {};

// un violon EST-UN instrument
class Violon : public Instrument {};
```

  - à chaque fois que vous dites/pensez `A` est un `B`
  - « est-un » : au niveau conceptuel (sémantique) et au niveau du type

- ordre d'appel des constructeurs/destructeurs

Etude de cas

EST-UN ou **A-UN** ?

Restriction des accès lors de l'héritage

Les niveaux d'accès peuvent être **modifiés lors de l'héritage**

Syntaxe :

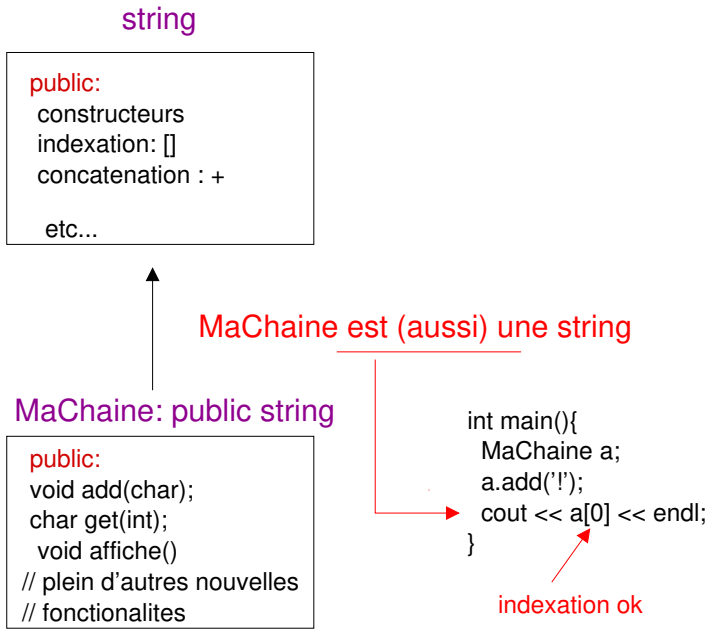
```
class ClasseEnfant: [accès] classeParente
{
   /* Déclaration des membres
      spécifiques à la sous-classe */
   //...
};
```

où **accès** est le mot-clé `public`, `protected` ou `private`. Les crochets entourant un élément `[ ]` indiquent qu'il est optionnel.

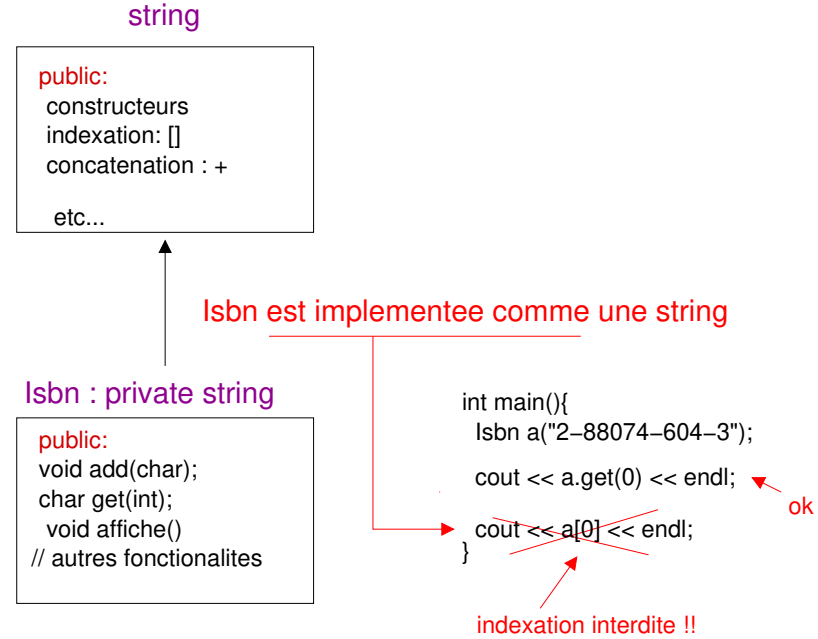
Les droits peuvent être conservés ou restreints, mais **jamais relachés** !

Par défaut, l'accès est privé.

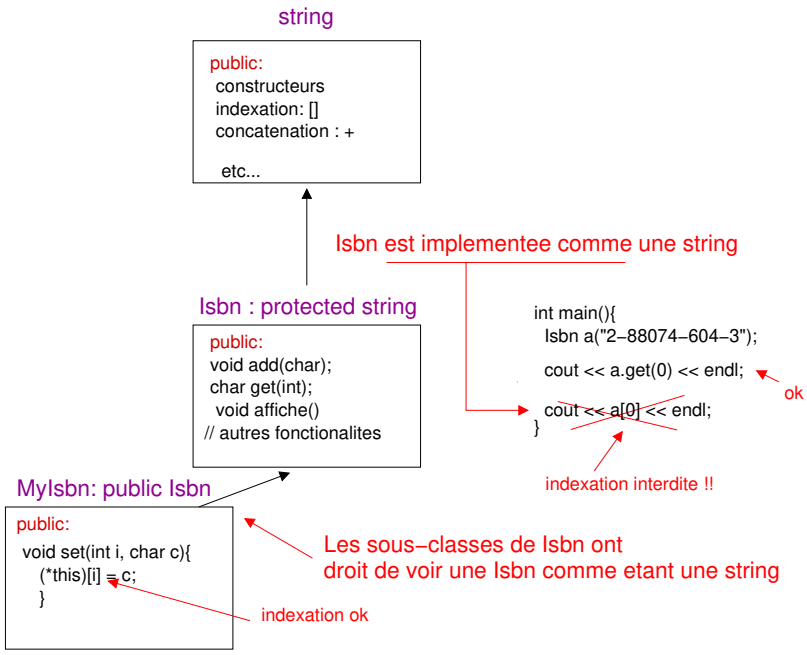
Droit d'héritage public



Droit d'héritage privé



Droit d'héritage protégé



☞ Relation **EST-IMPLEMENTÉ-COMME**

Restriction des accès lors de l'héritage (2)

Récapitulatif des changements de niveaux d'accès aux membres hérités, en fonction du niveau initial et du type d'héritage :

		accès initial		
		`public`	`protected`	`private`
héritage	**`public`**	`public`	`protected`	pas d'accès
	`protected`	`protected`	`protected`	pas d'accès
	`private`	`private`	`private`	pas d'accès

Le type d'héritage constitue une *limite supérieure à la visibilité.*

Pour préparer le prochain cours

- Vidéos et quiz du MOOC semaine 5 (première partie) :
  - Polymorphisme et résolution dynamique des liens [10 :58]
  - Polymorphisme : méthodes virtuelles [17 :25]
  - Masquage, substitution et surcharge [19 :11]
  - Classes abstraites [13 :55]

  ☞ La partie sur les collections hétérogènes sera abordée la semaine suivante avec un complément plus long en cours

- Le prochain cours :
  - de 14h15 à 15h (résumé et quelques approfondissements)