

Support MOOC

Concepts  
centraux

Etude de cas

Compléments

# Programmation Orientée Objet (C++) : Héritage

Jamila Sam

Laboratoire d'Intelligence Artificielle  
Faculté I&C

# Vidéos, transparents et quiz

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

## ☞ Semaine 4

Objet	
Traitements	Données
Encapsulation et Abstraction Classes Héritage simple/multiple Polymorphisme Classes abstraites/virtuelles Résolution des collisions de noms	Attributs Appels aux constructeurs des attributs (hérités) Statiques

# 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



# 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

string

public:  
constructeurs  
indexation: []  
concaténation : +  
etc...

MaChaine: public string

public:  
void add(char);  
char get(int);  
void affiche()  
// plein d'autres nouvelles  
// fonctionalités

MaChaine est (aussi) une string

```
int main(){  
    MaChaine a;  
    a.add('!');  
    cout << a[0] << endl;  
}
```

indexation ok

# Droit d'héritage privé

string

```
public:  
constructeurs  
indexation: []  
concatenation : +  
  
etc...
```

isbn : private string

```
public:  
void add(char);  
char get(int);  
void affiche()  
// autres fonctionalites
```

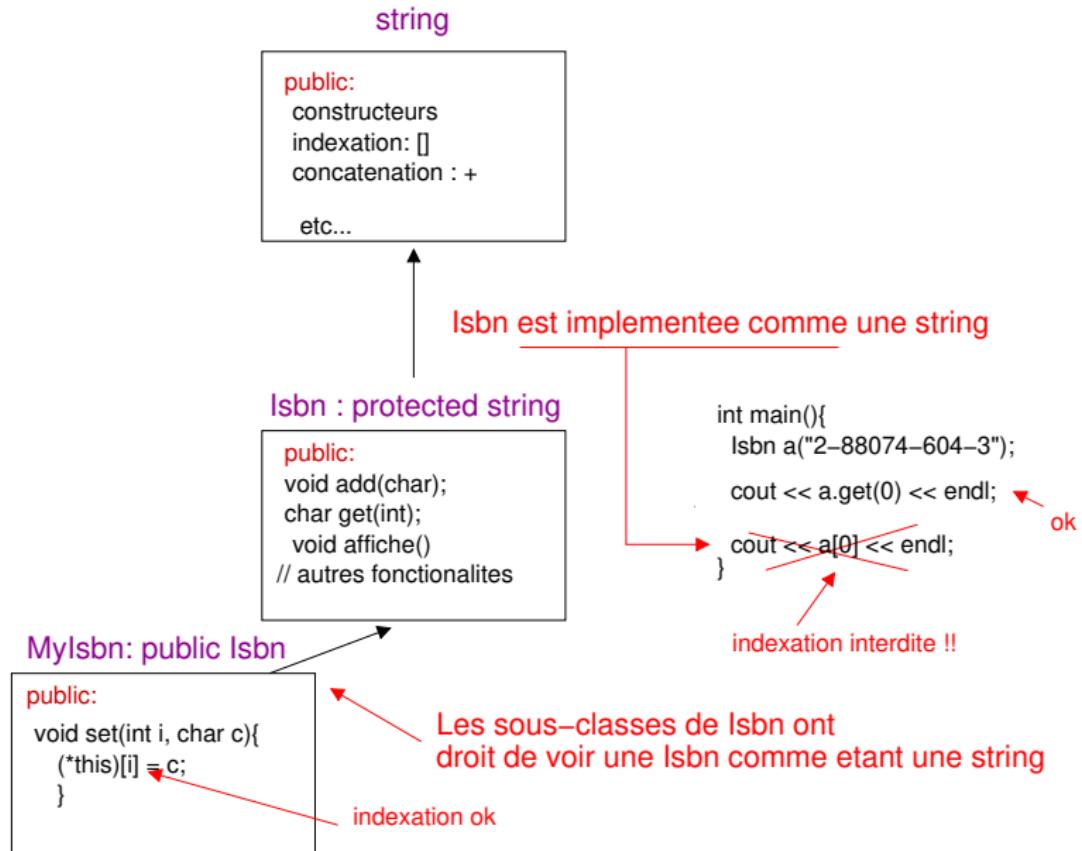
isbn est implementee comme une string

```
int main(){  
isbn a("2-88074-604-3");  
  
cout << a.get(0) << endl;  
}  
cout << a[0] << endl;
```

ok

indexation interdite !!

# Droit d'héritage protégé





# 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	<b>public</b>	public	protected	pas d'accès
	<b>protected</b>	protected	protected	pas d'accès
	<b>private</b>	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)