

# Programmation Orientée Objet : Héritage multiple

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle  
Faculté I&C

## Organisation du travail (semestre)

	MOOC	déc.	cours 1 h Jeudi 8-9	exercices 2 h Jeudi 9-11
1 22.02.24		0	Intro + compil. séparée	
2 29.02.24	1. Intro POO	0	Intro POO	
3 07.03.24	2. Constructeurs/Des	0	Constructeurs	
4 14.03.24	3. Surcharge des opé	0	Surcharge	
5 21.03.24	4. Héritage	0	Héritage	
6 28.03.24	5. Polymorphisme	0	Polymorphisme 1	
- 11.04.24		-	vacances Pâques	
7 04.04.24		1	Polymorphisme 2 / Collections hétérogènes	
8 18.04.24		-		Série notée
9 25.04.24	6. Héritage multiple	2	Héritage multiple	
10 02.05.24	(7. Etude de cas)	-	Templates	
12 16.05.24		-	(Ascension)	
11 09.05.24		-	Structure de données abstraites ; Bibliothèques	
13 23.05.24	(7. Etude de cas)	-	Bibliothèques (fin) + Révisions	
14 30.05.24		-		Examen

## Objectifs de la leçon d'aujourd'hui

- ▶ Concepts fondamentaux
- ▶ Étude de cas

## Concepts fondamentaux

- ▶ Buts, syntaxe (aucune difficulté)
- ▶ Ordre d'appel des constructeurs/destructeurs
  - ↳ ordre de déclaration *d'héritage*
- ▶ Sens (= sémantique) de l'héritage multiple ?
  - ▶ diagramme en losange
  - ▶ héritage et classes *virtuel(les)*
  - ▶ appel du constructeur de la classe virtuelle

# Etude de cas (simples)

Que faut il corriger pour que le code suivant compile :

```

class A                { public: A(int x) : a(x) {}
                      private: int a;          };

class B : public virtual A { public: B() : A(0) {} };

class C : public virtual A { public: C() : A(1) {} };

class D : public B, public C { };

int main()
{
  D d1;
  return 0;
}

```

# Cas numéro 2

Le code suivant compile-t-il ?

```

#include <iostream>
using namespace std;

class A
{ public: void f() const { cout << "A "; } };

class B : public virtual A
{ };

class C : public virtual A
{ };

class D : public B, public C
{ };

int main()
{
  D d1;
  d1.f();
  return 0;
}

```

# Cas numéro 3

Le code suivant compile-t-il ?

```

#include <iostream>
using namespace std;

class A
{ public: void f() const { cout << "A "; } };

class B : public virtual A
{ public: void f() const { cout << "B "; } };

class C : public virtual A
{ public: void f() const { cout << "C "; } };

class D : public B, public C
{ };

int main()
{
  D d1;
  d1.f();
  return 0;
}

```

# Cas numéro 4

Le code suivant compile-t-il ?

```

#include <iostream>
using namespace std;

class A
{ public: virtual void f() const { cout << "A "; } };

class B : public virtual A
{ public: void f() const { cout << "B "; } };

class C : public virtual A
{ public: void f() const { cout << "C "; } };

class D : public B, public C
{ };

int main()
{
  D d1;
  d1.f();
  return 0;
}

```

Objectifs

Concepts

Etude de cas

Cas 1

Cas 2

Cas 3

Cas 4

## Dernière question

Quelle différence entre les cas 3 et 4 ?

Que change le *virtual* ?

Donnez un exemple *illustratif*.