Programmation Orientée Objet : Surcharge des opérateurs

Jean-Cédric Chappelier

Laboratoire d'Intelligence Artificielle
Faculté I&C

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Organisation du travail (semestre)

		MOOC	déc.	cours 1 h Jeudi 8-9	exercices 2 h Jeudi 9-11
1	20.02.25		0	Intro + compil. séparée	
2	27.02.25	1. Intro POO	0	Intro POO	
3	06.03.25	2. Constructeurs/Des	0	Constructeurs	
4	13.03.25	3. Surcharge des opé	0	Surcharge	
5	20.03.25	4. Héritage	0	Héritage	
6	27.03.25	5. Polymorphisme	0	Polymorphisme 1	
7	03.04.25		1	Polymorphisme 2 / Collections hétérogènes	
8	10.04.25	6. Héritage multiple	1	Héritage multiple	
9	17.04.25		-	-	Midtem
-	24.04.25		-	vacances Pâques	
10	01.05.25	(7. Etude de cas)	-	Templates	
11	08.05.25		-	Structure de données abstraites ; Bibliothèques	
12	15.05.25	(7. Etude de cas)	-	Bibliothèques (fin) + Révisions	
13	22.05.25		-	-	Examen
14	29.05.25		-	(Ascension)	

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Objectifs de la leçon d'aujourd'hui

- Concepts fondamentaux
- Étude de cas

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Concepts fondamentaux

- à quoi sert la surchage des opérateurs :
  - pourquoi l'utiliseriez-*VOUS* ?
  - à quel niveau voulez-*VOUS* le faire ?
- surchage interne / surcharge externe
- (Attention aux copies !)
  (moins grave en C++11, le compilateur peut vous aider)

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Etude de cas

Comment afficher nos nombres complexes ?

Et finalement les nombres complexes, n'est-ce pas (surtout) pour faire des calculs ?.....

Décortiquons entièrement, pas à pas, la ligne suivante :

```
cout << 5.5 * ( Complexe(1.1, 2.2) * Complexe(3.3, 4.4) )
     << endl;
```

Et, si on a le temps, aussi celles-ci :

```
Complexe z1(1.1, 2.2);
Complexe z2(3.3, 4.4);
Complexe z3( z1 *= z2 );
```

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Décodage

Que signifie

```
cout << 5.5 * ( Complexe(1.1, 2.2) * Complexe(3.3, 4.4) )
     << endl;
```

Essayons de le réécrire en lignes d'une seule expression :

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Opérateur d'affichage

```
cout << z4;
```

☞ `cout.operator<<(z4);` ou `operator<<(cout, z4);` ?

☞ `void operator<<(ostream&, Complexe const&);`

```
cout << z4 << endl;
```

☞ `operator<<(operator<<(cout, z4), endl);`

☞ `ostream& operator<<(ostream&, Complexe const&);`

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Multiplication entre complexes

```
z1 * z2
```

☞ surcharge interne ou externe ?

Deux principes (parfois contradictoires) :

1. préférez la surcharge externe si un nouvel objet est créé ; sinon la surchage interne ;
2. utilisez la surcharge interne si vous *devez* accéder aux parties privées ;

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Multiplication entre complexes

ICI :

1. est-ce qu'un nouveau complexe est créé ?
   ☞ oui, `(z1*z2)` est un nouveau complexe
   Donc de ce point de vue : clairement une **surcharge externe**
2. est-ce qu'on peut faire la multiplication sans d'accéder aux parties privées ?
   ☞ cela dépend en général de l'encapsulation, mais dans ce cas précis pas trop car ici la bonne façon d'écrire cette multiplication serait justement de l'adapter à la représentation interne : utiliser la formule avec des coordonnées cartésiennes si le nombre complexe est représenté en cartésiennes et utiliser des coordonnées polaires si le nombre complexe est représenté en polaires.

   Donc de ce point de vue : plutôt une **surcharge interne**

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Multiplication entre complexes

On peut changer la seconde conclusion précédente en offrant dans la partie *publique* une méthode que la multiplication externe pourrait utiliser : l'opérateur `*=`

La *bonne* façon de faire consiste donc à :

1. définir en interne `operator*=`
2. définir en externe `operator*` qui utilise `operator*=`

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam

Multiplication par un `double` ?

Est-ce que l'on peut (déjà) écrire : `z4 = 5.5 * z3;` ?

☞ Attention ! Il y a une subtilité !

©EPFL 2025
Jean-Cédric Chappelier
& Jamila Sam